计算机科学2002Vol. 29№. 6

弱点数据库的设计及其实现*)

The Design and Application of the Vulnerability Database

杨 凡 蒋建春 卿斯汉

(中国科学院信息安全技术工程研究中心 北京100080)

Abstract This paper introduces the structure of the database and its implementation. It also introduces the application of the database in practice.

Keywords Threat, Vulnerability, Vulnerability database, Vulnerability scanner

从计算机网络诞生的那天起,其规模就一直呈指数规律增长(见图1)。短短几十年,无论是计算机网络提供的服务、服务的提供商、使用的目的还是使用的对象都发生了巨大的变化,网络已日益成为现代社会的一个关键部分。计算机网络不仅遍布全球,而且事实上已经开始影响人类的各个方面。网络系统已经广泛应用于工业、商业、政府和国防部门。主要的经济部门,包括能源、交通、电讯、制造、金融服务、医疗和教育都依赖于大量的运行在本地、本国和全球范围的网络。社会对网络的依赖和信任日益加深使得网络安全越来越受到人们的关注和重视。

通过分析几种主要的安全威胁(包括外部黑客的攻击、内部人员作案以及拒绝服务攻击),我们发现所有这些安全威胁之所以能够成功都在于:它们利用了系统中存在的某种弱点。例如:外部黑客的攻击主要利用了系统提供的网络服务中的弱点;内部人员作案则利用了系统内部服务及其配置上的弱点;而拒绝服务攻击主要是利用资源分配上的弱点。

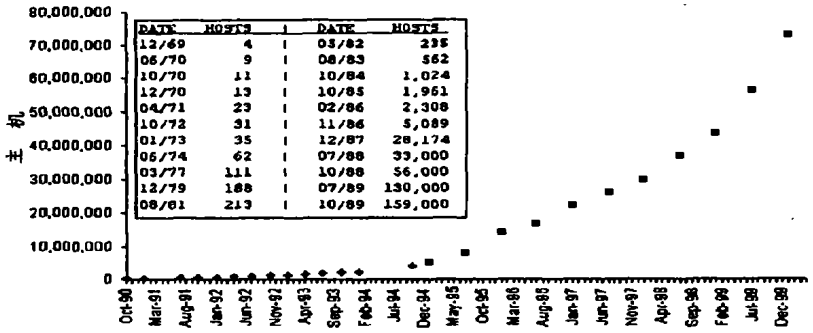

图1 主机数目增长示意图

在一个安全事故可能导致一场灾难的工程领域内(如飞机制造、核能利用等),研究者早已认识到:积累有关安全事件的信息并加以分析对于提高系统的可靠性,对于寻求更合适、更可靠的设计方法都是非常关键的,尤其是在系统非常复杂的时候。同样,如果我们能构造一个弱点数据库,尽可能多地收集和存储对系统安全造成威胁的弱点的有关信息,这对于我们的网络安全工作将是大有帮助的。

1. 建立弱点数据库的意义

建立弱点数据库具有以下几点意义:

1)避免漏洞重复出现。我们在对大量的攻击实例分析中发现,相当多的攻击都是利用缓冲区溢出漏洞进行的。导致缓冲区溢出的原因很简单:程序没有仔细地检查用户输入的参数和缓冲区的大小是否相符。既然造成漏洞的原因很简单,为什么还会重复地出现呢?很重要的一个原因就是开发人员并不清楚漏洞产生的原因,以致在开发过程中也没有引起足够的重视。因此积累、分析历史上的各种脆弱性信息,可以帮助开发人员从以前的错误中吸取教训,帮助其设计出更安全的软件,提供更有效的指导原则。

2)弱点信息种类繁多,攻击方式也是手法多变。新的弱点、新的攻击方法更是层出不穷,数据库是整理这些复杂信息的合适方式。新出现的弱点、攻击方法可以及时地输入库中保存起来,以供进一步的分析。

3)目前主要的安全工具,例如弱点扫描系统、入侵检测系统等都需要大量的弱点信息作为支撑。弱点数据库可以为这些系统提供信息来源。可以说,这些系统的有效性在很大程度上就依赖于底层的弱点数据库的规模与能力。

另外,如果弱点数据库的信息足够详尽,以至包括详尽的脆弱性检测、修补程序以及相应的攻击程序,我们还可能在这种数据库的基础上进行系统攻防实验,就象军事学上的实战演习一样,从而在系统的攻防实践中来检验各种安全技术、安全措施的有效程度。

2. 建库的指导原则

我们确定了以下几条指导原则,作为我们设计弱点数据库的出发点:

*)中国科学院软件所青年创新基金资助课题(cxZK5606)。国家重点基础研究发展规划资助项目(G1999035810)国家自然科学基金助项目(60083007)。

·建立弱点数据库的目的是为了存储各种弱点信息，所以我们首先力求数据结构的完整性，力求从各个方面全面地反映系统弱点的特性。

·随着对弱点认识的发展，有可能出现新的弱点属性，所以建库时应该考虑数据结构的可扩展性；如果以后发现某一属性对于弱点的研究很重要，应该可以很容易地把这种属性加入数据库中。

·尽可能全面地分析各种系统弱点，力求通过详细的分析抽取各方面的特征；由于某些限制，有些信息一时不能获得，相关的数据字段将保留为空(NULL)，在获得这些信息之后，可以重新加入。

·在各种有关弱点研究的文献中，一般会有一些系统弱点的实例分析，我们将引用他们的分析结果；但是这些文献中所举的例子中，很大一部分都是比较陈旧的系统(如 SOLARIS2.3、2.4，LINUX2.1等)中的弱点，这些系统基本已经淘汰，所以无从验证；但是这些数据还是有保存价值，从这些弱点信息、软件错误中我们可以吸取教训，防止以后再犯。

·在建立好数据库后，要考虑如何尽可能广泛地收集各种弱点信息，这是扩充数据库的关键，我们希望发挥更多人的力量，鼓励用户上载他们对各种系统脆弱点的分析结果，共同建设好弱点数据库。在具体实现过程中，我们将提供用户上载弱点信息的功能。

以上五条原则是我们建库的指导原则，下面一节将讨论弱点数据库的设计，这些原则将得到进一步的体现。

3. 弱点数据库的设计

我们通过对大量的弱点分析发现，要真正了解一个弱点的特性，除了了解其本身所具备的特征(例如：产生的原因、造成的危害)之外，还要从攻击者和防卫者的角度来了解弱点。从防卫者的角度来考虑，我们需要了解如何能检测出该漏洞以及如何修补该漏洞。从攻击者的角度考虑，我们需要了解利用该弱点进行攻击所需要的权限、攻击时在对方主机上会留下怎样的特征(入侵签名)，以及该攻击的难易程度。基于以上这些考虑，我们认为弱点数据库应包含以下字段：

首先，为了方便数据的组织，库中的每一个弱点都应该有一个标志，我们可以用一个数字再加上一个名称来标志该弱点：数字对应于该弱点在整个数据库中的位置，名称则以文字的形式概括性地描述了该弱点。表1中说明了这种结构。

表1 弱点标志

字段名称	详细描述	数据类型
Vid	弱点标志号	Smallint
Title	弱点的名称	Char(64)

·从弱点本身考虑。弱点的描述以文字的形式描述了该弱点的大致情形，可以用于说明系统中怎样的一种状态会导致系统的安全受到怎样的一种威胁。弱点的后果进一步地描述该弱点可能给系统带来的危害，这种描述应该比较具体，可以说明相关的文件或程序。除了这种具体的后果描述外，数据库还应该说明该后果所属类型，包括系统内部权限的非法盗用，系统数据完整性、保密性的受损害，正常的服务请求被拒绝。基于这一字段可以对库中的所有弱点进行统计分析，讨论各种不同类型的弱点的分布规律。基于这种分类，还可以讨论对付这些不同安全威胁的安全措施。表2说明了这三种字段的名称、描述及数据类型。

表2 弱点的描述与后果

字段名称	详细描述	数据类型
Description	有关该弱点的详细描述	BLOB
Impact	该弱点的可能后果描述	Char(128)
Impact_type	后果所属类型	Char(64)

·从防卫者角度考虑。从系统防卫的角度来看，检测系统中是否存在各种弱点是安全防卫的第一步，在检测出系统存在安全弱点之后，还应该给出修补漏洞、避免危害的方法。如果可以找到检测某种弱点的程序，可以将该程序加入数据库，基于数据库中的这一字段，就可以建立一个全新的系统弱点检测系统。修补方法这一字段描述该脆弱点的修补方法或补丁程序。在给出具体的检测程序、修补方法的基础上，还应该尽量地给出这些它们所属的类型。在这四项字段中，首先需要知道具体的检测、修补方法，在目前的情况下，我们还不可能找到(或者自己编写)所有的方法；所属的种类则需要做进一步的分析，有的补丁程序只是一些二进制的可执行代码，中间的具体细节厂商也没有透露。所以这些字段的内容有可能暂时为空，但是由于这些属性的重要性已经是很明显的，从数据结构的完整性考虑，我们将这四种属性纳入我们的弱点数据库。随着我们可获得的知识、信息的丰富，暂时为空的字段将一步一步地填充。表3说明了这四种字段的名称、描述及数据类型：

表3 弱点检测与修补

字段名称	详细描述	数据类型
Detection_script	可以检测该弱点的程序或脚本	Varchar(128)
Detection_type	该检测程序的类型	Char(32)
Fix_patch	修补方法或者补丁程序	Varchar(128)
Fix_type	修补方法所属的类型	Char(32)

·从攻击者角度考虑。讨论弱点的检测与修补是从系统防卫者的角度看问题，但系统的弱点涉及攻击者与防卫者双方，因此还有必要从攻击者的角度观察弱点的各种特性，从这一角度观察系统的弱点，我们可以进一步明确攻击者到底是怎样破坏我们的系统，他具有什么样的一些特性。在详细地分析攻击者的攻击手法之后，我们可以通过查看用户的行为轨迹来判断用户行为的企图，从而达到入侵检测的目的，这就为我们提供了一种更为主动的、更为有效的安全措施。如果能够找到相应的攻击程序，可以将这些程序组织起来，存储到数据库中。在对这些攻击手法进行分析的基础上，可以提取入侵签名。除了这些信息外，我们还可以加入另外两种信息：攻击者需要的访问权限以及攻击手法的难易程度。这两种信息从两个侧面进一步加深我们对攻击方法的认识。在系统发现攻击事件之后，可以根据这些信息大致判断攻击者的范围。在系统防卫的时候，还可以根据攻击的难易程度确定自己的防卫重点，最有效地利用自己有限的防卫资源。表4说明了这四种字段的名称、描述及数据类型。

表4 利用该弱点进行攻击的细节

字段名称	详细描述	数据类型
Intrusion_Script	利用该弱点进行攻击的程序或脚本	Varchar(128)
Intrusion_signature	入侵签名	Varchar(128)
Access_required	攻击者需要的访问权限	Char(64)
Easy_of_exploit	实现该攻击的难易程度	Char(64)

·相关信息。以上从系统弱点本身、系统防卫者、攻击者三个角度描述了弱点的多种属性，有这些信息，我们对弱点的认识已经有一定的深度。但是由于系统弱点的研究尚处于初期阶段，因此远未成熟，可能还会出现一些新的看问题的角度。我们的弱点数据库也应该能够反映这种发展、这种变化，所以在我们的数据库中还应该包括一些其它信息，目前我们设立了受影响的系统与应用程序、有关的参考文章或安全建议两个字段。基于前一字段，我们可以统计不同厂商、不同应用中的弱点状况。基于后一字段我们可以找到一些相关的文章（通常也是比较权威、比较深刻的分析文章），这些文章也是我们工作的基础。此外，我们还设立了一个"弱点信息来源"字段，表明弱点的来源，以方便用户在遇到问题时，可以直接到来源处寻找答案。目前，我们搜集信息的渠道主要有这样三种：①互联网，包括用户直接上传、新闻组、邮件列表及 E-mail；②公开发表的著作、文章；③同行之间相互交流。

表5　其它相关信息

字段名称	详细描述	数据类型
Sys_utility_impacted	受影响的系统和应用程序	TEXT
Reference_advisory	相应的参考文章或安全建议	Varchar(128)
Source	弱点信息来源	Varchar(32)

以上五个方面的信息，包括漏洞的标志、描述与后果、检测与修补、相关的攻击信息以及其它信息，共同组成了我们的弱点数据库。在设计以上这种数据结构的时候，兼顾了我们的指导原则。我们主要的着眼点是数据结构的完整性与可扩展性，所以广泛地分析了各方面的情形，并借鉴了其他人的成果。在这些信息方面，可能还有一些我们尚未完全理解的，或者很难获取信息的，这些是受到当前的整体状况影响所致，随着人们认识的进一步提高，有关信息的进一步公开，我们的数据库信息将逐步完善。

4. 弱点数据库实现模型

在分析比较了几种主流的数据库管理系统之后，我们决定采用 MYSQL 作为弱点数据库的管理系统。之所以做出这样的选择是考虑到这种数据库小巧灵活，而且因为它不提供存储过程与触发器的功能，速度比一般商用系统要快。对于我们这种研发性的应用，这两种功能也没有必要。

MYSQL 数据库管理系统可以在各种 UNIX 操作系统平台下运行，我们目前使用的是 LINUX 操作系统，它对系统配置要求比较低，能运行于多种平台，功能强大。如果有需要，也可以很方便地移植到其它操作平台。

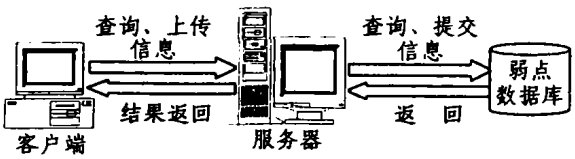

图2　人机交互

在设计人机界面的时候，我们考虑使用基于 Web 的界面，MYSQL 数据库在 LINUX 服务器上运行，用户可以在另外一台机器上通过 WWW 服务来访问数据库。客户端通过 URL 地址访问弱点数据库所在的 LINUX 服务器，然后请求访问弱点数据库。服务器在接受到用户请求后，对弱点数据库进行相关操作，并将结果信息以超文本的形式返回到用户的浏览器上，该处理过程由 PHP 程序完成。整个实现流程如图2所示。

5. 建立弱点数据库存在的一些问题

建立弱点数据库的主要目的就是收集弱点信息，并将其共享。弱点数据库与其他行业数据库一个很重要的不同在于：弱点数据库中数据的收集和共享都有其特殊性。

我们知道收集和发布弱点信息的时效性是非常重要的，如果在弱点已经造成重大损失后再来发布相关信息，意义不大。因此，弱点数据的收集很重要的一个渠道是通过网络由发现弱点的人直接将相关信息上传。这样就难免存在一些问题：上传信息的人的身份不能确认，上传信息的可信度也存在问题。为了解决这些问题，在发布弱点信息之前先要对弱点信息进行确认。而确认工作是十分困难的，因为不同的弱点存在于不同的系统或不同的软件，要实现各种检测环境，也许需要多家组织和机构进行合作。

同时，弱点数据库的共享也存在一些技术、动机和后果等方面的障碍。技术障碍包括数据的有用性和正确性、数据的格式、弱点的分析和确认。即使技术上面的障碍可以克服，我们依旧面临着动机方面的障碍。比如说发现一个弱点以及收集相关信息是一个相当复杂的过程，因此如何能与别人公平交换信息也是值得考虑的，否则用户是不愿意将信息上传与别人共享的。

另外一个重要的障碍是共享将导致的后果。弱点数据库共享所带来的后果往往是难以预料的，因为共享的信息很有可能被黑客用于攻击有价值的目标。我们将在以后的研究中进一步研究如何解决这些问题。

6. 弱点数据库在实际中的应用

在上文中，我们提到了弱点数据库中的一些字段可以被用来作为弱点扫描系统和入侵检测系统的基础。目前，我们正在开发的弱点扫描系统就用到了弱点数据库。

我们的扫描系统分为客户端和服务器端。其中客户端主要功能是提供给用户一个选择界面，用户可以选择其想要扫描的主机，想要扫描的内容，并上传到服务器端。服务器接收到用户的请求后开始扫描目标机器，并将扫描后的结果输出到客户端。结果包括系统的弱点信息、弱点的危险程度及其相应的修补建议。具体过程如图3所示。

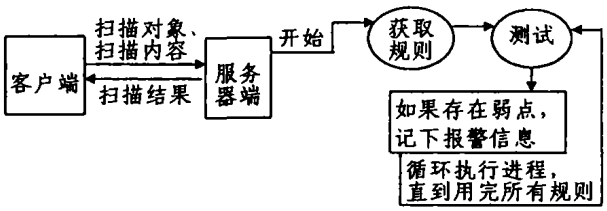

图3　弱点扫描过程

在该系统中用到的规则来自"Detection_Script"字段。而扫描结果中的弱点信息、弱点的危险程度及其相应的修补建议等内容也分别取自弱点数据库中相应的字段。由此可见，弱点数据库的规模在很大程度上决定了弱点扫描系统的能力。

同样，入侵检测系统的核心——规则集也是由弱点数据库的入侵签名字段提供的。

总结语　本文介绍了建立弱点数据库的意义并详细介绍了弱点数据库的结构及其实现。在最后，结合实例介绍了弱点

一种多级数据仓库体系结构和双通道视图更新算法

Multi-level Data Warehouse Architecture and Update Algorithm of Double Channels View

熊忠阳　黄海龙　张玉芳　欧　灵

（重庆大学计算机学院　重庆400044）

Abstract This paper puts forward a new multi-level data warehouse architecture based on WHIPS of Stanford University. First adds sub-data warehouse that has the similar structure of data warehouse. Second brings OLAP history base into multi-level data warehouse for enhancing OLAP query efficiency. Finally for ensuring the data consistency and improving query efficiency, it presents 2 channel view updating algorithm to maintain multiple views on line and parallel realize OLAP query. The improved system separates data updating and OLAP service, avoids the data inconsistency. The establish sub-data warehouse satisfies the department OLAP, increases the working efficiency, and enhances retractility and scalability.

Keywords Data warehouse, Double channel, On-Line Analyze Process, View

1　引言

联机分析处理(On Line Analyze Processing:OLAP)是决策支持系统的重要组成部分，它为企业管理人员提供了利用集成化的数据仓库进行更准确、更完整的信息查询的能力。数据仓库是支持企业或组织决策分析处理的，拥有面向主题的、集成的、相对稳定的数据集合。在数据仓库的联机维护中，只有保证数据仓库数据与数据源的一致性，才能为高层决策人员提供全面一致、有效的分析型战略数据信息。

数据仓库中一般存在多种集成粒度上的视图，视图的元组数据存储在数据仓库中，这种实例化视图一般没有自动更新功能。当数据源发生变化后，数据仓库视图也应该随之发生相应的更新，在数据更新时可能出现以下几个问题：

1)几个视图可能由同一数据源派生，此数据源的更新将带来多视图的同时更新；

2)数据源的广泛分布性，对于分布式的数据仓库存在并发更新的可能；

3)数据仓库多视图的更新和前台的OLAP查询数据之间的一致性问题。

本文着重研究了多视图的OLAP查询与数据仓库数据更新之间的数据一致性问题，对相关数据仓库结构进行了改进并设计出相应的双通道算法，使数据仓库的数据更新和OLAP服务之间的事务分离，避免数据出现不一致，同时使数据仓库的扩展性得到了提高。

2　多级数据仓库体系结构

数据仓库中的下钻(Drill-down)操作在多数情况下需要访问数据源。由于视图建立后的更新维护相对滞后于源数据，当下钻查询用到原始数据和视图数据时，则会出现查询不一致的问题[2]。为了对视图进行联机维护，及时应答用户的OLAP请求，数据源也要参与相应运算，这又会降低数据源端的联机事物处理OLTP性能。为此，在Stanford大学WHIPS系统[1]的基础上，为了解决视图更新时的数据一致性问题以及提高OLAP的效率，本文提出了一种改进的数据仓库结构。

改进之一：在数据源与数据仓库之间加入一个具有数据仓库结构的子仓库，形成多级数据仓库模式，如图1所示。

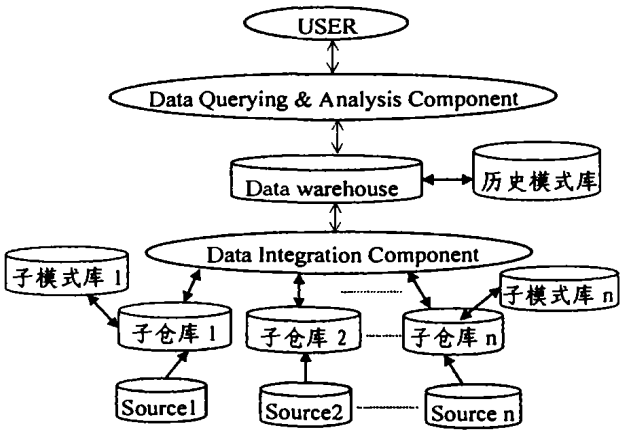

图1　多级数据仓库体系结构

子仓库位于数据源和数据仓库之间，拥有和数据仓库类似的数据结构及关系定义。子仓库向上面的数据仓库发送更新消息及更新数据，向下接收数据源的更新请求。子仓库的引入不仅简化了数据仓库的更新操作，而且简化了数据仓库的视图管理，提高了效率。引入子仓库的主要优点是：

1)保证了数据仓库和OLAP服务数据的一致性，提高了

数据库在实际中的应用。在建立弱点数据库的过程中也遇到了一些现实的问题，这需要我们在今后的工作中加以解决。

参 考 文 献

1　Taimur A, Ivan K, Eugene S. A Taxonomy of Security Faults. In: Proc. of the national computer security conf. 1996

2　Du W, Mathur A P. Categorization of software Errors that led to Security Breaches: [Techinical Report]. CS Department, Purdue University, 1998

3　Kumar S, Spafforf E. A Taxonomy of Common Computer Security Vulnerability Based on their Method of Detection: [Tech Report]. Purdue University, 1995

4　余建斌. 黑客的攻击手段及用户对策. 人民邮电出版社，1998