一种基于三级内网蜜场系统的网络蠕虫检测技术

陈志杰[1] 鲜 明[2]
(国防科学技术大学训练部信息中心 长沙 410073)[1]
(国防科学技术大学电子科学与工程学院 长沙 410073)[2]

摘 要 在针对内网的入侵行为中以网络蠕虫危害最大,传统的网络安全手段存在缺陷,蜜场技术作为监测和信息搜集手段与传统网络安全手段的本质不同。研究提出了一种基于三级内网蜜场系统的网络蠕虫检测技术,它可以有效提高蠕虫检测速度,对提高内部网络的安全,具有重要实用意义。

关键词 内网,网络蠕虫,蜜场,蜜罐
中图法分类号 TP393 **文献标识码** A

Honeyfarm-based Network Worm Detection Technique in Three-tied Intranet

CHEN Zhi-jie[1] XIAN Ming[2]
(Information Center of Exercitation Department, National University of Defense Technology, Changsha 410073, China)[1]
(School of Electronic Science & Engineering, National University of Defense Technology, Changsha 410073, China)[2]

Abstract There are many intrusion activities on the intranet in which network worms cause the greatest harm. Traditional network security means have many deficiencies, as a means of monitoring and information gathering, honey field technology is essentially different with traditional network security measures. A network worm detection method based on honeyfarm of three-tied intranet was proposed. With the ability of improving the speed of worm detection, enhancing the security of internal network, it has important practical significance.

Keywords Intranet, Worm, Honeyfarm, Honeypot

在网络日益普及的今天,随着网络安全风险系数不断增加,网络安全变得越来越重要。尤其是内部网络,由于安全制度不完善和使用人员安全意识不强或错觉,攻击者可以轻易地对这些防护较弱的内网进行攻击和入侵,如进行拒绝服务攻击、从事非授权的访问、肆意窃取和篡改重要的数据信息、安装后门监听程序以便随时获得内部重要信息、传播计算机病毒、摧毁主机等等。如果因特网中流行的蠕虫和木马病毒被植入,内网上的每一台主机都面临成为攻击目标的危险。在针对内网的入侵行为中以网络蠕虫危害为最,多样化的传播途径和复杂的应用背景使蠕虫的传播速度增高、潜伏性变强、覆盖面更广,从而造成巨大的危害。

本文研究提出了一种基于三级内网蜜场系统的网络蠕虫检测技术,它可以有效提高蠕虫检测速度,对提高内部网络的安全,具有重要实用意义。

1 背景

某大型机构的内部网络系统按照其部门职能和地域划分建成了三级结构,网络拓扑结构如图1所示。该系统三级结构内部网络采用星型拓扑结构,以分支机构服务器为核心,视频会议、网页新闻等网络应用都以分支机构服务器为主,下层是各基层单位网络;各子网相对独立,在分支机构主干汇接处形成子网边界。由于大部分服务都基于分支机构服务器展开,因此保护好分支机构服务器的安全是该三级内部网络安全的重中之重。

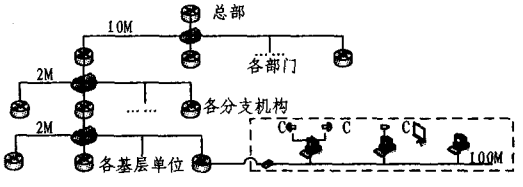

图1 某机构网络拓扑结构

2 蠕虫概述

蠕虫是一种有着巨大危害的病毒,它攻击网上的脆弱主机,并在受害主机上自我复制,继而通过扩散攻击网络上其他的脆弱主机[1]。

蠕虫利用常见服务的安全漏洞或策略缺陷,通过网络进行传播。如果有漏洞的服务被安装在大量可公开访问的主机上,蠕虫就能自动入侵这些系统,造成大范围的感染。在发现和感染脆弱系统阶段,蠕虫占用大量网络和系统资源,如果蠕虫负载具有攻击性,则会造成窃取用户敏感数据、删除宝贵资源等无法挽回的后果。蠕虫最具有威胁性的特点是传播速度

本文受国家自然科学基金(60372039)资助。
陈志杰(1960—),女,高工,主要研究方向为网络与信息系统等;鲜 明(1970—),男,研究员,主要研究方向为雷达信号处理、网络安全、电子对抗、信息战,E-mail:qwertmingx@tom.com。

很高，许多事件和研究工作[2,3]表明，一个工作良好的蠕虫能在几分钟内90%感染以上的网络弱点系统，严重扰乱网络正常秩序。这样快速的传播能力需要自动化的检测响应机制来进行有效防御。

蠕虫的攻击过程通常包括感染、传播和执行负载3个阶段。

在感染阶段，蠕虫已入侵到本地系统，它为了进行复制传播必须修改一些系统配置、运行程序，为后续阶段做准备，如生成多个线程用于快速探测新的目标、设置互斥标志防止系统被再次感染而影响传播速度等。通过这个阶段的工作，蠕虫将所入侵的系统变为一个新的感染源。这个感染源继续攻击过程以扩大感染范围。

在传播阶段，蠕虫要尽可能地发现可攻击的目标，利用漏洞获取系统权限，将自身从一个系统传播到另一个系统，从而入侵大量的目标。

蠕虫负载是指能够完成攻击意图的代码[4]。有些蠕虫包含负载，而有些蠕虫不包含负载，通过执行负载，蠕虫可以完成攻击任务，例如发动拒绝服务攻击、删除系统数据等。

2.1 蠕虫行为特征

从网络检测的角度来看，比较理想的时机是在蠕虫传播阶段，可以利用探测特性和自传播特性及早发现蠕虫活动[5]。在这一阶段，蠕虫会采用多种技术最大限度地发现可攻击目标。为尽快发现攻击目标，在一定时间内会产生大量的网络探测报文，尽管蠕虫扫描策略有多种形式，如选择性随机扫描、可路由的扫描、分解扫描、DNS扫描、完全扫描等。但在监视网络活动时可以观测到这些扫描都有以下的树型网络活动特征，如图2所示。

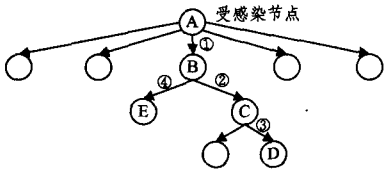

图2 蠕虫树型活动特征

(1)一对多通信。为了感染更多的系统，蠕虫在利用诸如多线程等技术的同时向多个目标系统发送探测信息，以及在发现了多个弱点系统以后进行自传播时，都会出现一对多的通信模式，如图2中的A节点所示。

(2)相似的通信模式。蠕虫是自传播程序，包含用于探测和攻击弱点服务的特定模块。当蠕虫利用同样的模块探测和攻击时，就会出现这一特征。特别是在三级网中所有的主机运行相同版本服务，即使蠕虫会变换使用不同的探测和漏洞利用模块，它们的数理也是有限的。在图2中，A使用相同的目的端口探测其他目标系统，A感染B、B感染E等，也利用相同的服务漏洞。如果一个节点使用相同的模式访问了多个节点，则这个节点就为异常节点。

(3)无效的连接访问。无效的连接访问是由于目标系统不存在或服务端口不存在所引起的。大多数的蠕虫选址策略含有随机因素，因此，蠕虫的网络活动中有相当数量的目标IP地址或服务不存在的报文。有针对性地分析这些报文就会加快蠕虫检测的速度。

(4)链路通信模式。节点只有先被感染，然后才能去感染其它节点。随着感染的扩散，会形成感染路，如图2中的链路①→②→③和①→④。

2.2 蠕虫模型分析

在蠕虫研究中，一般以简单蠕虫传播模型[6]为依据：

$$\frac{dJ(t)}{dt}=\beta J(t)[N-J(t)] \tag{1}$$

式中，$J(t)$是在t时刻被感染的主机数；N是易感染主机数；$\beta=a/N$，a是蠕虫感染率。当$t=0$时，$J(0)$是初始状态时已被感染的系统个数。这一模型能够较好地描述扫描机制一致的蠕虫，特别是在传播的初期没有人工干预或网络拥塞可以忽略的情况下，能够刻画蠕虫的传播特征。文献[6]中使用上述模型研究蠕虫检测，通过选用卡尔曼滤波算法检测异常的扫描流量来发现蠕虫。

文献[7]中利用离散模型研究蠕虫扫描特征，提出了基于受害主机数的检测算法。假定易受攻击的主机是统一地分布在一个实际分配的IPv4空间(所有潜在的受攻击主机都分布在这个空间内，设定为$T=109$)，不是整个IPv4空间(表示为$\Omega=232$)。假定N是整个三级网中易受攻击的主机数，n_i是在i段时间内整个三级网中被入侵的主机数，s是在每个时间段蠕虫的扫描率。所以在第$i+1$时间段内，空间T内的被扫描主机数应为$k_{i+1}=sn_i\frac{T}{\Omega}$。在这个空间内，一个主机被攻击的几率是$1-(1-\frac{1}{T})^{k_{i+1}}$，据此得出蠕虫传染公式：

$$n_{i+1}=n_i+[N-n_i]\left(1-(1-\frac{1}{T})^{sn_i\frac{T}{\Omega}}\right) \tag{2}$$

上述研究都是以全球互联网为参照模型，参考主机数目非常大。如果考虑三级网蠕虫爆发时的情况，特别是蠕虫在传播选址时优先选用本地网络，则三级网会很快被感染。设置因特网$N=360000$，$T=232$，$s=120/s$，$n_0=10$；三级网$N=100$，$T=216$，$s=120/s$，$n_0=1$，则模型中蠕虫传播时间如图3所示。

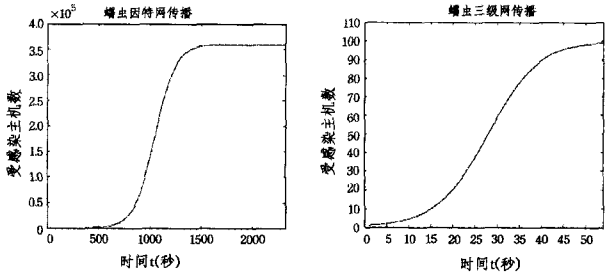

图3 因特网传播与三级网传播时间对比

图3中在同样的传播速度下，蠕虫需要2322秒的时间感染整个网络的易感染主机，但是感染一个典型的三级网易感染主机只需要不到60秒。通常对蠕虫的检测方法要以牺牲一部分网络为代价，而三级网中各子网同等重要，因此在三级网内并不完全适用。针对这种情况，本文通过使用基于虚拟漂移的蜜场系统来解决这种不适应问题。

3 基于虚拟漂移的蜜场系统

3.1 蜜场的概念

为了在大型的分布式网络中方便地部署和维护一些蜜罐，对各个子网的安全威胁进行收集，提出了蜜场的概念。即

所有的蜜罐均部署在蜜场中，而在各个内部子网中设置一系列的重定向器(Redirector)，若检测到当前的网络数据流是黑客攻击所发起时，通过重定向器将这些流量重定向到蜜场中的蜜罐节点中，并由蜜场中部署的一系列数据捕获和数据分析工具对攻击行为进行收集和分析[8]。典型的蜜场结构如图4所示。

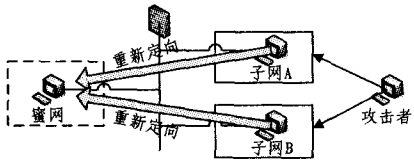

图4 典型的蜜场结构

蜜场模型的优越性在于其集中性，使得其部署变得较为简单，即蜜场可以作为安全操作中心(SOC: Security Operations Center)的一个组成部分[9]，由安全专业研究和管理人员进行部署和维护。蜜场模型的集中性也使得蜜罐的维护和更新、规范化管理及数据分析都变得较为简单。此外，将蜜罐集中部署在蜜场中还减少了各个子网内的安全风险，并有利于对引入的安全风险进行控制。

3.2 漂移技术

漂移技术是最近新出现的一种动态的、主动的网络路由技术，它实现了应用服务在不同结点之间的漂移。这种多样化的动态漂移，不仅能在活跃服务器发生故障时实现由主结点到备份结点的漂移，也可以在正常的响应客户方请求的过程中根据安全策略，将请求调度到不同的服务器上。

漂移技术与通常的网络地址转换技术(Network Address Translation)最核心的区别在于：网络地址转换技术允许一个整体机构以一个公用 IP 地址出现在 Internet 上。它是一种把内部私有网络地址(IP 地址)翻译成合法网络 IP 地址的技术。而通常内部私有网络地址所做的 NAT 都是一一对应的关系，即一个私有 IP 地址对应一次连接(某次连接的始终只对应建立连接的唯一私有 IP 地址)；而漂移技术改变了这种对应关系，采用动态的映射关系，在一次网络通信的过程中可以改变连接所对应的私有 IP 地址对象。

漂移技术有3个特点[10]：

服务的分离：漂移应该能对指定的协议或服务，指定端口，包括对来访 IP 进行分离；例如将对 HTTP/TELNET/FTP 的访问分离到不同的结点上，不同的结点可能提供不同的服务、也可能提供同样的服务；而针对来访 IP 的分离，使得漂移技术可以用于主动路由和攻击导向。

策略的调整：漂移技术不同于以往的 NAT 技术或者静态路由技术还在于，它具有很强的动态性和自适应性，它能通过手动或自动的形式，修改网络数据路由规则，调度服务请求；因此漂移技术结合入侵检测技术时，能在系统检测到入侵行为时，主动调整漂移策略实现数据的智能引导，从而达到入侵防御的能力。

调度的粒度：漂移可以是以客户方的 IP 地址或 URL 为单位进行，也可以达到单个 TCP 报文的细粒度，也就是在已建立好的 TCP 连接内实现动态漂移，为此，每个备份服务器都会保存有当前活跃服务器的每一条 TCP 连接的相关信息，以便在发生连接漂移后能够直接从需要的 TCP 顺序号处恢复。

利用网络数据漂移技术可以将网络攻击数据引导到指定的目标，不仅转移了遭受的攻击同时也迷惑了攻击者，在把安全风险转嫁的同时不影响正常数据的通信传输。

4 蜜场系统应用

4.1 框架

为了监控三级网蠕虫活动情况，本文建立了基于虚拟漂移技术的蜜场系统，使用蜜罐系统的协议状态分析引擎检测蠕虫，利用虚拟漂移技术加速检测过程。整个蜜场的部署以分支机构网管中心为核心，配置日志服务器和控制主机，由其网络管理员完成对整个蜜场的控制；在分支机构和各基层单位子网内配置一个探测主机节点，用来检测各子网内的异常和攻击行为。分支机构级蜜场一般有 20 个左右的探测主机节点，200 台左右的整体主机空间。

三级网蜜场整体框架如图5所示。

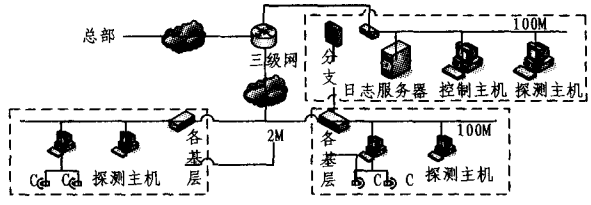

图5 蜜场框架

4.2 蠕虫检测

为了监控三级网的蠕虫活动情况，在分支机构和各基层单位子网内配置探测节点，各蜜罐探测节点之间可以相互共享捕获数据信息，通过虚拟漂移技术将针对本子网的攻击行为引入蜜罐内，使用协议状态分析引擎的蠕虫检测规则，实时分析本地子网的蠕虫攻击行为。

当蠕虫病毒根据一定的扫描策略扫描存在漏洞主机的地址空间时，蜜罐节点通过分析捕获的网络蠕虫扫描攻击数据，采用协议状态分析引擎来判断是否有蠕虫攻击。三级网蜜场系统的协议状态分析引擎通过分析蠕虫病毒传播过程的协议状态信息，对蠕虫病毒造成的网络异常和攻击进行有效的检测。通过使用虚拟漂移技术，蜜场系统能够在蠕虫传播的早期向本地管理员发出预警信息。此外蜜场系统在发现蠕虫攻击行为之后，通过通知管理员切断蠕虫所在子网的连接，阻断网络蠕虫的攻击。下面以“冲击波病毒”为例，介绍三级网蜜场系统使用协议状态分析引擎检测蠕虫病毒的过程。

冲击波蠕虫病毒主要针对 Windows 2000 和 Windows XP 主机中所运行的 DCOM RPC 程序漏洞，利用 TCP 135 端口来发起攻击。一旦计算机感染了该病毒，它会按照一定的规则或随机产生 IP 地址，并向使用这些 IP 地址的其它主机发送大量攻击数据包，以试图感染没有及时更新系统补丁的主机。

冲击波病毒的攻击数据包是一个向 TCP 135 端口发起的 TCP 连接请求。大量的攻击数据包用来消耗被攻击的主机的 CPU 和内存资源，最终导致其堆栈溢出而被感染。这种攻击方式是典型的 TCP Syn Flooding 攻击。

TCP Syn Flooding 攻击的原理是在短时间内，攻击者发送大量 SYN 报文建立 TCP 连接，在服务器发送应答后，客户

端不发出确认，服务器会维持每个连接直到超时，这样会使服务器端的 TCP 资源迅速枯竭，导致正常的连接不能进入。结合协议状态分析模型，可知冲击波蠕虫病毒攻击发生时将有大量的 TCP 连接处于 q_2 状态，指定计数常量 n_1 和时间间隔常量 i_1，则统计攻击序列集合 $C(q_2,n_1,i_1)=\{s|count(s,q_2)\geqslant n_1, interval(s,q_2)\leqslant i_1\}$，当 $S\in C(q_2,n_1,i_1)$ 时，$Attack(s)$ 为真，即对于同一 TCP 服务器连接的状态序列中，在时间间隔 i_1 内，有大于等于 n_1 个 q_2 状态存在时，认为发生了 TCP Syn Flooding 攻击。根据 TCP Syn Flooding 攻击的原理结合行为模式转换算法，本文编制了攻击检测函数。

检测 TCP 建立连接是否正常的伪代码如下：

```
int Check TCPConnect(receivedTCPPacket){
    Remove(s,si);
    while(TRUE){
        switch(STATEi)
        case q0:
            if(SYN)
                STATEi=q1;Si=Si+(q1,t1);
        case q1:
            if(SYN&&ack)
                STATEi=q2;s=Join(s,si);
            if(count(s,q2) >= n1)
                CheckTCPSynFlooding(s,q2);
            else
                return FAIL;
        case q2:
            if(ack){
                STATEi=q3;s=Remove(s,si);
                return SUCCESS;
            }
            else
                return FAIL;
    }
    i++;
}
```

TCP Syn Flooding 攻击检测算法的伪代码如下：

```
int CheckTCPSynFlooding(s,q){
    if(interval(s,q)>=i1){
        Alert();
        return ATTACK;
    }
    else
        return NORMAL;
}
```

函数 1 用来判断 TCP 建立连接是否正常，函数 2 用来判断是否有 TCP Syn Flooding 攻击。其中，s_i 表示服务器当前存在的第 i 个 TCP 连接的状态，s 表示所有连接状态合并作为一个状态序列。当收到客户端发出的建立 TCP 连接的 SYN 包时，便跟踪记录此连接的状态，直到成功完成或超时。如果多个连接中的 q_2 状态达到 n_1，则计算接收这些数据包的时间间隔；如小于 i_1，则输出 TCP Syn Flooding 攻击的报警信息。

由此可见，三级网蜜场系统运用协议状态分析引擎可以方便地检测到蠕虫病毒的攻击，及时向管理员报警，阻断蠕虫病毒在三级网内的传播。

4.3 三级网蠕虫模型

本文对文献[7]中的离散模型式(2)进行改进，使其适应三级网结构和虚拟漂移蜜场特点，并检验虚拟漂移技术对蜜场系统检测蠕虫速度的提升。假定有一个大小为 D 的三级网蜜场。开始时易受攻击的主机数为 u_0。在不失去一般性的情况下，假定 $u_0=D\alpha$，α 为易受攻击主机传染率。在 i 段时间内，本地被入侵主机数为 v_i，$v_0=0$ 表示最初在蜜场内没有主机被入侵。第一个蜜罐节点发现诱导入蠕虫数据的时间为 t_1。因此由式(2)推导得到公式：

$$v_{i+1}=u_0\left(1-(1-\frac{1}{T})^{\Sigma_{j=0}^{i}sn_j\frac{T}{n}}\right),\quad (i+1)<(t_1+t_r) \tag{3}$$

式中，t_r 表示虚拟漂移模块将蠕虫数据引入蜜场的时间。在这个反应时间 t_r 后，假定新的易受攻击的主机数变为 $u_1=D\gamma=D\alpha+A_\gamma$。$A_\gamma$ 为蠕虫传播数据经过数据诱导的蜜罐节点数。现在得到公式：

$$v_{i+1}=u_1\left(1-(1-\frac{1}{T})^{\Sigma_{j=t_1+t_r}^{i}sn_j\frac{T}{n}}\right),\quad (i+1)\geqslant(t_1+t_r) \tag{4}$$

根据式(4)，可以方便地计算出蜜罐节点检测出蠕虫时整个网络主机的被感染率。表 1 是使用不同的 α 和 D 时，整个网络的被感染率。

表 1 网络感染率计算结果

α \ D	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}
10%	3.9102	1.9538	0.9776	0.4890	0.2446	0.1222	0.0612
25%	1.5595	0.7805	0.3900	0.1954	0.0979	0.0490	0.0246
50%	0.7786	0.3890	0.1949	0.0976	0.0489	0.0246	0.0123
75%	0.5171	0.2586	0.1295	0.0650	0.0326	0.0164	0.0082
100%	0.3871	0.1939	0.0971	0.0486	0.0245	0.0123	0.0062

对应式(4)的网络感染率仿真结果如图 6 所示。从图 6 中可以看出 α 和 D 是影响整个公式的主要原因。

在 $D=2^{15}$ 即网络规模为因特网的情况下，无论 α 取值多大，网络感染率几乎为零时，就发现蠕虫。而在三级网中，D 的大小一般为 2^8 左右，此时 α 为决定整个蜜场系统检测蠕虫效率的主要因素。在蜜罐探测节点对蠕虫数据进行虚拟漂移之后，在三级网感染率很低的情况下，很快地得到足够受感染主机。因为通过对蠕虫数据在蜜罐节点虚拟漂移之后，蜜罐节点容易被蠕虫病毒感染攻击，增加了易受攻击的主机数，α 接近 100%。在 $D=2^8$ 和 $\alpha=100\%$ 的三级网蜜场系统中，当 0.5%的三级网主机被感染时，蜜场系统就发现了蠕虫。因此在整个三级网蠕虫检测过程中，虚拟漂移成为整个系统的关系因素。

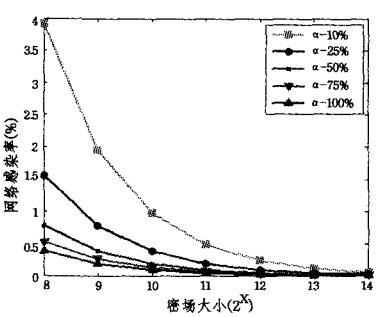

图 6 网络感染率仿真结果

通过以上的分析，蜜场系统提供了一个有效地检测三级网内蠕虫的方法，即通过虚拟漂移技术，提高蜜罐节点的蠕虫感染率，及时准确地发现三级网内的蠕虫。

结束语 近年来，蜜场技术被广泛应用于网络安全领域，它不仅弥补了传统网络安全手段的不足，而且具有自己独特的优点和适应性。本文利用蜜场系统给出了一个有效地检测三级网内蠕虫的方法，在大规模网络中进行蠕虫样本捕获，可以在第一时间捕获流行的蠕虫病毒，从而提高网络的安全运行效率。

参 考 文 献

[1] Berk V H, Gray R S, Bakos G. Using Sensor Networks and Data Fusion for Early Detection of Active Worms[C]// IEEE INFOCOM. 2003, 3

[2] Moore D, Paxson V, Savage S, et al. Inside the Slammer Worm [J]. IEEE Security & Privacy, 2003, 1(4): 33-39

[3] Staniford S, Moore D, Paxson V, et al. The Top Speed of Flash Worms[C]// Proc. of the 2004 ACM Workshop on Rapid Malcode. Washington: ACM Press, 2004: 33-42

[4] Berk V, Bakos G, Morris R. Designing a Framework for Active Worm Detection on Global Networks[C]// Cole J L, Wolthusen S D, eds. Proc. of the IEEE Int'l Workshop on Information Assurance. Darmstadt: IEEE Computer Society, 2003: 13-23

[5] 张新宇，卿斯汉，李琦，等. 一种基于本地网络的蠕虫协同检测方法[J]. 软件学报，2007

[6] Zou C C, Gao L, Gong W, et al. Monitoring and Early Warning for Internet Worms[C]// Jajodia S, ed. Proc. of the 10th ACM Conf. on Computer and Communication Security. Washington: ACM Press, 2003: 13-23

[7] Dagon D, Qin Xin-zhou, Gu Guo-fie, et al. HoneyStat: Local Worm Detection Using Honeypots[EB/OL]. Georgia Institute of Technology

[8] 柳亚鑫，吴智发，诸葛建伟. 基于 Vmware 的第三代虚拟 Honeynet 部署以及攻击实例分析[OL]. http://www.icst.pku.edu.cn/honeynetweb/ honeynetcn/, 2005-08

[9] 诸葛建伟. 蜜罐及蜜网技术简介[EB/OL]. http://www.icst.pku.edu.cn

[10] 陈长松. 基于智能漂移技术部署的隐形蜜罐系统[J]. 信息网络安全技术研究，2006(5)

(上接第 21 页)

智慧校园能把分布在大量廉价 PC 上的内存、存储和计算能力集中起来成为一个虚拟的资源池，使资源利用率提高，运用原有软硬件资源实现大规模计算与存储，实现超级计算机的性能。并通过校园网为师生提供服务，实现海量计算、大规模数据处理等功能，加快科研项目的完成速度，提高科研质量。

智慧校园可以渗透到校园的每一个角落，从实验室、图书馆、教室到防盗、门禁以及校车、停车位等。智慧校园不仅能采集智能物品、传感器和激发器的数据，而且可以实现物品的相互连网，使人们可以从任何地方通过多种方式，如通过邮件、短信、即时通信系统及有线和无线电话等，对其进行控制。

结束语 本文介绍了基于云计算及物联网构建智慧校园，利用云计算、虚拟化、SOA 构建校园云，结合物联网、RFID 构建智慧校园。利用虚拟化技术将高校硬件资源整合形成资源池，实现资源动态分配、动态负载均衡，提高资源利用率。同时，利用资源监控、负载管理、冗余备份、动态部署、快速部署、资源调度、多租户技术、海量数据处理、大规模分布式存储、数据管理技术构建校园云。结合物联网技术，将物理基础设施和 IT 基础设施普遍互联，实时监控，将获得的大量数据利用校园云处理，真正实现智慧校园。但是，也存在一些问题有待解决，如隐私安全等问题。相信随着云计算、物联网技术的不断成熟，在不久的将来，智慧校园的构建无疑会成为一个热点。

参 考 文 献

[1] 王鹏. 走近云计算[M]. 北京：人民邮电出版社，2009

[2] Miller M. 云计算[M]. 北京：机械工业出版社，2009

[3] 王庆波，金萍，何乐，等. 虚拟化与云计算[M]. 北京：电子工业出版社，2010

[4] 英特尔开源软件技术中心，复旦大学并行处理研究所. 系统虚拟化：原理与实现[M]. 北京：清华大学出版社，2009

[5] 鲁松. 计算机虚拟化技术及应用[M]. 北京：机械工业出版社，2008

[6] 周洪波. 物联网：技术、应用、标准和商业模式[M]. 北京：电子工业出版社，2010

[7] 邹生，何新华. 物流信息化与物联网建设[M]. 北京：电子工业出版社，2010

[8] IBM 商业价值研究所. 智慧地球[M]. 北京：东方出版社，2009

[9] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters[J]. 2004

[10] Ghemawat S, Gobioff H, Leung S-T. The Google File System [J]. 2003

[11] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters[C]// OSDI'04: Sixth Symposium on Operating System Design and Implementation. San Francisco, CA, December 2004

[12] Ghemawat S, Gobioff H, Leung S-T. The Google File System [C]// 19th ACM Symposium on Operating Systems Principles. Lake George, NY, October 2003

[13] Chang F, Dean J, Ghemawat S, et al. Bigtable: A Distributed Storage System for Structured Data[C]// OSDI'06: Seventh Symposium on Operating System Design and Implementation. Seattle, WA, November, 2006