软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(10):2251−2265 [doi: 10.13328/j.cnki.jos.004682]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

创建软件定义网络中的进程级纵深防御体系结构*

崔竞松[1,2], 郭 迟[3], 陈 龙[4], 张雅娜[1], Dijiang HUANG[5]

[1](武汉大学 计算机学院,湖北 武汉 430072)

[2](软件工程国家重点实验室(武汉大学),湖北 武汉 430072)

[3](武汉大学 卫星定位导航技术研究中心,湖北 武汉 430079)

[4](中山大学(珠海校区) 移动信息工程学院,广东 珠海 519000)

[5](School of Computing Informatics and Decision Systems Engineering, Arizona State University, AZ, USA)

通信作者: 郭迟, E-mail: guochi@whu.edu.cn

摘 要: 云计算因其资源的弹性和可拓展性,在为用户提供各项服务时,相对于传统方式占据了先机.在用户考虑是否转向云计算时,一个极其重要的安全风险是:攻击者可以通过共享的云资源对云用户发起针对虚拟机的高效攻击.虚拟机作为云服务的基本资源,攻击者在攻击或者租用了某虚拟机之后,通过在其中部署恶意软件,并针对云内其他虚拟机发起更大范围的攻击行为,如分布式拒绝服务型攻击.为防止此种情况的发生,提出基于软件定义网络的纵深防御系统,以及时检测可疑虚拟机并控制其发出的流量,抑制来自该虚拟机的攻击行为并减轻因攻击所受到的影响.该系统以完全无代理的非侵入方式检测虚拟机状态,且基于软件定义网络,对同主机内虚拟机间或云主机间的网络流量进行进程级的监控.实验结果表明了该系统的有效性.

关键词: 无代理技术;内网防火墙;虚拟机纵深防御;网络虚拟化;软件定义网络

中图法分类号: TP393

中文引用格式: 崔竞松,郭迟,陈龙,张雅娜,Huang DJ.创建软件定义网络中的进程级纵深防御体系结构.软件学报,2014,25(10):2251−2265. http://www.jos.org.cn/1000-9825/4682.htm

英文引用格式: Cui JS, Guo C, Chen L, Zhang YN, Huang DJ. Establishing process-level defense-in-depth framework for software defined networks. Ruan Jian Xue Bao/Journal of Software, 2014,25(10):2251−2265 (in Chinese). http://www.jos.org.cn/1000-9825/4682.htm

Establishing Process-Level Defense-in-Depth Framework for Software Defined Networks

CUI Jing-Song[1,2], GUO Chi[3], CHEN Long[4], ZHANG Ya-Na[1], Dijiang HUANG[5]

[1](Computer School, Wuhan University, Wuhan 430072, China)

[2](State Key Laboratory of Software Engineering (Wuhan University), Wuhan 430072, China)

[3](Global Navigation Satellite System Research Center, Wuhan University, Wuhan 430079, China)

[4](School of Mobile Information Engineering, Sun Yet-Sen University (Zhuhai Campus), Zhuhai 519000, China)

[5](School of Computing Informatics and Decision Systems Engineering, Arizona State University, AZ, USA)

Corresponding author: GUO Chi, E-mail: guochi@whu.edu.cn

Abstract: Cloud computing is gaining momentum against traditional method in providing users various services with greater flexibility and scalability. Before switching to cloud computing, users must take into account the security of cloud as an extremely important factor. That is because in the cloud environment, attackers can initiate efficient attacks to cloud users through the shared cloud resources such as virtual machines. Since virtual machines (VM) are basic resources of cloud service, by compromising or renting several virtual machines,

* 基金项目: 国家高技术研究发展计划(863)(2013AA12A206, 2013AA12A204); 国家自然科学基金(41104010, 91120002); 高等学校学科创新引智计划(B07037)

收稿时间: 2014-02-28; 修改时间: 2014-07-07; 定稿时间: 2014-07-31

attackers may deploy malicious software into those machines and launch a wider range of attacks to other virtual machines such as distributed denial of service (DDoS). To tackle this issue, this paper proposes a defense in depth system based on software defined networking to be able to detect suspicious virtual machines and monitor the flow they issued in time, and inhibit the aggressive behavior from the suspected virtual machines to mitigate the attack consequences. The system detects the virtual machines' running state in a completely non-intrusive and agent-free way, and monitors network traffic between virtual machines on the same host or between cloud hosts at process level based on software defined networking. Experimental results demonstrate the effectiveness of the system.

Key words: agent-free; inside network firewall; virtual machines' defense in depth; network virtualization; software defined networking

云计算在一个新的基础设施环境下,推动着企业迈向更加高效和灵活的工作模式.然而,安全问题仍是一个关键问题,阻碍了人们全面转向云的步伐.虚拟机(virtual machine,简称 VM)作为云计算基础设施中基本的计算资源,成为了恶意软件极具吸引力的攻击目标.云用户被授予特权,可以通过网络访问其 VM 并安装任意种类任意版本的操作系统(operate system,简称 OS)及应用,而这些 OS 和应用本身可能带有漏洞,会给系统带来风险.这些带有漏洞的 VM 极易被攻击者攻陷,并成为攻击者攻击云系统内其他 VM 的跳板,被攻击者远程控制发动分布式拒绝服务攻击(distributed denied of service,简称 DDoS)等极具破坏力的网络攻击,导致多台 VM 瘫痪.因此,为防止脆弱的 VM 被利用作为攻击跳板,对云内网中虚拟机间的网络行为应加以关注并及时隔离攻击源以保护其他 VM 不受攻击.

虽然云管理员可以利用特权修补 VM 的漏洞、在 VM 安装个人防火墙或基于主机的入侵检测系统(host-based intrusion detection system,简称 HIDS)来保护其免受攻击.然而,这种做法可能会中断用户正在运行的服务,甚至因为应用程序版本依赖的原因而导致服务功能障碍.再者,这类基于代理的方式,由于其代理是可见的,攻击者可以伪造检测结果,禁用代理.最近有些方案提出将监控代理放置于 VM 外,并使用虚拟机自省(virtual machine introspection,简称 VMI)技术获取 VM 状态信息.此类方案可有效克服上述方法的弊端.但尽管如此,因虚拟机自省技术普遍运用在主机级检测上,通过现有 VMI 技术获取的主机信息,包括内存信息和磁盘信息等,仍然很难直接运用在网络攻击行为的检测上.

针对于网络攻击的检测和防护,传统防护方法通常会根据网络拓扑情况在边界或旁路部署安全设备,如基于网络的入侵检测系统(network-based intrusion detection system,简称 NIDS)和网络防火墙.这类方法的部署与网络结构紧密耦合,存在诸多不适应性.同时,对于同一主机中 VM 间通信的情况不具备检测功能.作为改变现有网络格局的软件定义网络(software defined networking,简称 SDN)技术,由于其灵活性,也被应用在了网络安全防护领域.然而,现有的软件定义安全技术对于攻击行为的检测和控制粒度仅能细到流(flow)这一级别,因而无法通过定位进程级攻击源的方式彻底隔离恶意进程、杜绝攻击的再次发生.

鉴于现有 VMI 技术和 SDN 技术难以解决本文所讨论的问题,我们提出了一个虚拟化网络纵深防御系统,整合位于主机层的 VMI 技术和位于网络层的 SDN 技术的优势,通过 VMI 获取信息,辅助 SDN 定位发起攻击的进程,并对其发出的流量进行检测、控制、隔离,实现更细粒度的进程级攻击防御.本系统可以实时检测到 VM 发起的与其他 VM 的新建连接,反向追溯发起连接的 VM 和具体进程并予以检测,继而根据检测结果控制该连接的数据包流量.通过 VMI 的进程监控和 SDN 控制,可以检测恶意进程并重定向可疑流量到一个基于网络的入侵检测引擎中进行深层数据包检测以及消除恶意流量.

本文旨在提出一种类似内网防火墙的虚拟化网络纵深防御系统,目的是防止脆弱的 VM 成为攻击者攻击其他 VM 的跳板、隔离攻击源、抑制攻击行为的发生范围,并减轻攻击产生的后果.其优势在于可以及时捕获网络流量并控制流量最终的目的地(原目的地址、入侵检测系统或是直接丢弃),实现进程级的网络攻击防御,而不是改良现有的入侵检测算法.其中,系统的纵深防御是指:

- 除开云主机与外部网络的通信流量,也可监控同一云主机内部的虚拟机间的网络流量;
- 可在完全无代理的状态下主动感知并抑制恶意行为;
- 具备进程级的检测功能和网络控制功能,不仅可以检测具体进程的状态,对网络流量的控制也可细致到进程,因而不影响虚拟机内其他的网络服务,最大限度地保证了服务质量;

- 对虚拟内网中的流量可进行重定向,为网络控制提供了除告警、丢包外的其他选择.

1 相关工作

现有很多着重于检测被感染 VM 并保护云中其他 VM 不受其攻击的技术,如防火墙和入侵检测系统(intrusion detection system,简称 IDS).在网络层,传统的网络防火墙和基于网络的入侵检测系统(network-based intrusion detection system,简称 NIDS)建立在“被监控网络内-防火墙外”模式下.此种模式不适用于被感染 VM 和攻击目标处于同一云主机内部的情况,因为网络防火墙和 NIDS 搜集的必须是进出被监控网络的信息.在主机层,个人防火墙、HIDS[1,2]、防病毒和防间谍软件系统等,运行在它们所保护的 VM 内,使得这些工具很容易被窃取到系统特权的攻击者破坏.故这些传统网络安全技术并不能很好地解决本文所需着重解决的问题:及时检测有害 VM,并在其试图与其他 VM 进行连接时予以阻止,防止攻击的发生.

目前,部分研究在虚拟化技术的基础上进行的网络分析检测工作,可以检测网络通信过程中的恶意行为,保障 VM 的安全.

Yassin 等人[3]提出了一种基于云计算的框架,称作“基于云的入侵检测服务(cloud-based intrusion detection service,简称 CBIDS)”.CBIDS 能够克服传统入侵检测技术在部署方式上的不足.同时,该框架可以嗅探到网络内任意地方的网络流量,可疑数据包会传递到 CBIDS 中进行深度检测,并产生相应的警告.然而,CBIDS 仅能检测网络流量并提出警告,却无法对攻击行为做出干预,本身不能阻止攻击的发生.除此之外,CBIDS 的检测对象仅为数据包,无法追溯该数据包所属的 VM 和进程,因此不能满足本文应用场景中提出的隔离恶意进程网络行为的要求.Laureano 等人[4]提出的一种网络入侵检测方案,从被监控 VM 外部监控,检测并阻塞来自 VM 内部服务的攻击.该方案包括了入侵检测机制和响应机制,保证既能检测出攻击行为,也能阻止其发生.此方案中,检测的是主机内 VM 与外界(主机以外)进行网络通信过程中的流量,而这些网络通信行为由安装在主机内的防火墙软件进行管理.并且,响应机制中通过该防火墙软件阻塞 VM 的端口,并断开其与外界的通信.因此,对于同一主机内 VM 与 VM 间的网络通信并未得到关注,即,该方案无法应用在同主机中 VM 间的网络行为检测中.Gupta 等人[5]提出了一种基于配置文件的网络入侵检测和防御系统,该系统为每一个 VM 创建属于它的配置文件,该配置文件描述了此 VM 的所有网络行为,这些行为将会被用作判断是否被攻击的依据.该系统可以检测来自 VM(内部实体,如恶意的云用户)和外部实体(外部攻击者)的入侵.然而,系统中的网络流量通过虚拟机的 IP 地址过滤,即,对流量的识别是 VM 级的,因此,对于进程级检测的细粒度要求,该系统无法满足.SDN[6−8],即软件定义网络,解耦网络控制功能与转发功能,使得网络控制部分可被直接编程.这种灵活性使得在其上部署网络攻击检测能实现更灵活的安全防护,因此,现有部分网络攻击分析检测研究也转向了 SDN.Shin 等人[9]提出了一个基于 OpenFlow 的安全应用程序开发框架——FRESCO.FRESCO 提供了支持 OpenFlow 的检测控制组件的模块化组成方式.利用 RESCO,安全组件可以整合基于 DPI 的传统网络安全应用发出的告警信息,进而将其作为 FRESCO 检测部件的输入参数或触发 FRESCO 响应部件生成新的流规则.然而,在 FRESCO 中,一个操作(检测或响应)是指对于网络数据包(或称为流)的处理,这些由 FRESCO 提供的操作源于 NOX OpenFlow 控制器所能支持的操作.因此,对于进程级的检测,FRESCO 同样无法支持.Jafarian 等人[10]使用 OpenFlow 开发了一个移动目标防御(moving target defense,简称 MTD)框架,可以透明地改变机器的 IP 地址,且此变化具有难预测性和较快的速度.同时,它也保持了结构的完整性和最小的操作开销.此框架原本的设计并不针对于云环境,但经过若干改造,仍能在云中使用.若是应用在云中,改变的将不再是物理机器的 IP 地址,而是每台 VM 的独立 IP 地址.虽然也能通过动态配置保护 VM 的安全,但是:首先,对于恶意云用户这一情况,它会立即失效;再者,该框架的防护级别也属于 VM 级,因此做不到进程级的防御.Shin 等人[11]提出了 CloudWatcher 框架,为大型、动态的云网络提供监控服务.CloudWatcher 会自动改变网络数据包走向,以供预装的网络安全设备进行检查.CloudWatcher 以一个应用程序的形式运行在网络操作系统上(如 NOX,Beacon),可用来控制 SDN 环境下的网络路由器或交换机.因此,此框架的作用类似于 NIDS,并提供一定的控制功能.同样,它也具备普通 NIDS 的缺陷:无法检测同一主机内 VM 间的流量;检测粒度为流,无法达到进程级.

综上所述,现有的网络分析检测方式由于其各自的缺陷,均无法直接应用在本文所需解决的问题上,而所有方式对比本文所采用方法的共同弱点在于,无法提供进程级的细粒度检测.上述方式与本文工作的详细对比见表 1.

Table 1 Comparison with other researches

表 1 与其他工作的对比

项目名	监测粒度	能否监控同一云主机内的流量	能否对攻击行为进行干预
CBIDS	flow 级	√	×
Marcos Laureano 方案	flow 级	×	√
Sanchika Gupta 方案	VM 级	√	√
FRESCO	flow 级	√	√
MTD	VM 级	×	√
CloudWatcher	flow 级	×	√

对于本系统所采用的 VMI[12]技术,即虚拟机自省技术,是从 VM 外部分析 VM 内部状态的一种方法.基于 VMI 的 VM 保护方案将安全保护工具和被监控 VM 予以隔离,因此无需在被监控 VM 中安装任何代理.这一特点使得该方案抵抗攻击的能力大为增强,不受 VM 的影响.虽然现有的很多研究[13−17]采用了此种方法,但普遍运用在了主机级检测方面,没有涉及网络攻击研究,故这里不再详述.

2 攻击场景与模型

2.1 攻击场景

在本框架的应用场景中,云用户能够通过 Internet 访问其虚拟机,并具有在 VM 内安装任意操作系统或应用的权限.云管理者在确定某 VM 存在安全威胁之前,不对其运行作任何干预.攻击者可能来自云外部,将一台已被控制的 VM 作为跳板,攻击云内其他 VM;或是本身即为控制着若干 VM 的云用户,可直接攻击其他 VM.

攻击者通过给目标 VM 发送恶意网址或文件,达到攻击的目的.若 VM 用户用带有漏洞的应用程序打开这个网址或文件,文件或网站里的 shellcode 会触发执行,并下载病毒程序到本地.该病毒程序会自动执行,并窃得该 VM 的控制权.无论病毒程序以何种方式执行,它都会试图与攻击者连接以接受进一步的命令.攻击者下达的命令会控制病毒程序获取当前局域网的拓扑,扫描其他 VM 或主机的漏洞,攻击然后控制其他机器,窃取这些被攻击机器里的敏感信息,供攻击者发起 DDoS 攻击.

如图 1 所示,攻击者在通过路过式下载(drive-by download)或恶意的邮件附件等方式成功攻击一台 VM 后,获得其控制权,接下来便可以该 VM 为起点,去发动更多的攻击.本研究的目标即为准确、及时地发现已被控制的可疑 VM,并阻止它们发动更多针对云里其他 VM 的攻击.

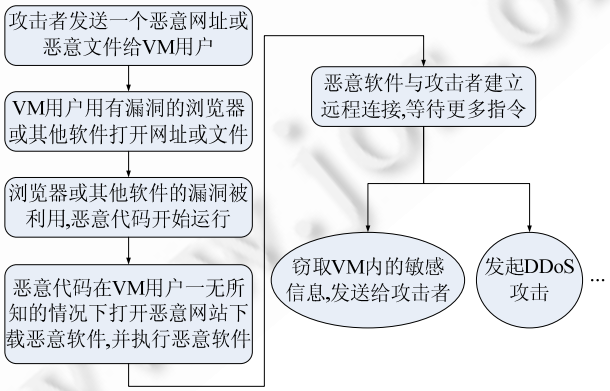


Fig.1 Attack scenario

图 1 攻击场景

2.2 “终止-检测-转发”模型

当某 VM 发起一个面向其他 VM 的连接时,在 OVS 和 OFS 的帮助下,可以及时获取该连接的数据包流量.为判断此连接的发起者是否可信,本文提出一个“终止-检测-转发”模型来阻止攻击行为的发生.

在该模型中,终止、检测、转发分别代表 VM 处于 3 种安全状态下本系统对其采取的操作.其中,终止代表判定VM或其进程为恶意后采取的隔离操作,此操作将阻塞从该VM或进程发出的所有流量;检测代表发现VM或进程可疑后采取的深度检测操作,此操作将该VM或进程发出的流量重定向到NIDS处进行深度数据包检测;转发则代表 VM 一切正常,所有流量正常转发到原目的地.

图 2 简要描述了“终止-检测-转发”模型在系统处于不同状态时的行为,值得一提的是,本系统暂时只提供网络层面的控制,不关注恶意进程的清除.

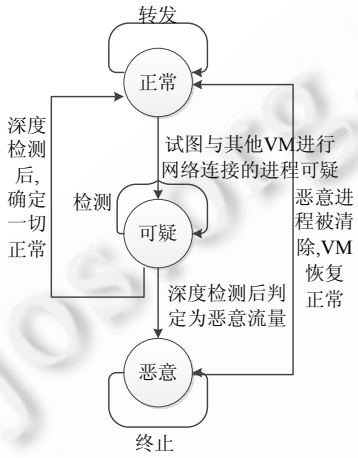


Fig.2 “Suspend-Check-Forward” model

图 2 “终止-检测-转发”模型

该模型集成了 VMI 技术和 SDN 的可编程特性,包括两个层次——VM 级和进程级.

- VM 级:本系统通过定位发出连接请求的 VM 并检测其安全性,若判定为恶意,则终止该流量并阻塞从此 VM 发出的所有流量,将其彻底隔离;若 VM 为可疑,却无法确认为恶意,则重定向此 VM 发出的流量到 NIDS 处进行深度检测;若能确定为正常,则让流量继续转发到原目的地;
- 进程级:与 VM 类似,确定发起连接的具体 VM 中的具体进程,并进行检测.若判定为恶意进程,则终止其与其他 VM 进程的通信行为;若可疑,则重定向它发出的流量到 NIDS 处检查;若正常,则按正常程序进行转发.

3 系统设计与实现

3.1 总体框架

本系统旨在基于软件定义网络技术,建立一个云服务器集群中的内网防火墙.服务器的虚拟化环境由特权域(privileged domains,简称 PD),如 XEN 的 dom0,以及非特权域(unprivileged domains,简称 UPD),如 VM 等构成.云主机之间和主机内虚拟机之间分别由可编程的网络交换机 OpenFlow Switches(OFS)和 Open vSwitches (OVS)连接,OVS 转发主机内 VM 间的网络数据包,而 OFS 负责主机间的网络通信.本系统中,控制中心包括攻击分析器(attack analyzer,简称 AA)和网络控制器(network controller,简称 NC),其中,攻击分析器是系统中制定策略的核心部分,由它决定流量最后的处理为“终止-检测-转发”模型中的哪一部分;网络控制器则负责控制网络交换机的路由.本地部件包括特权域里的进程检测器(process inspector,简称 PI)、VMI,负责对 VM 及其中进程的

检测工作,以及非特权域里的 NIDS,负责确认重定向到此处的流量是否安全.本系统的总体框架如图 3 所示.

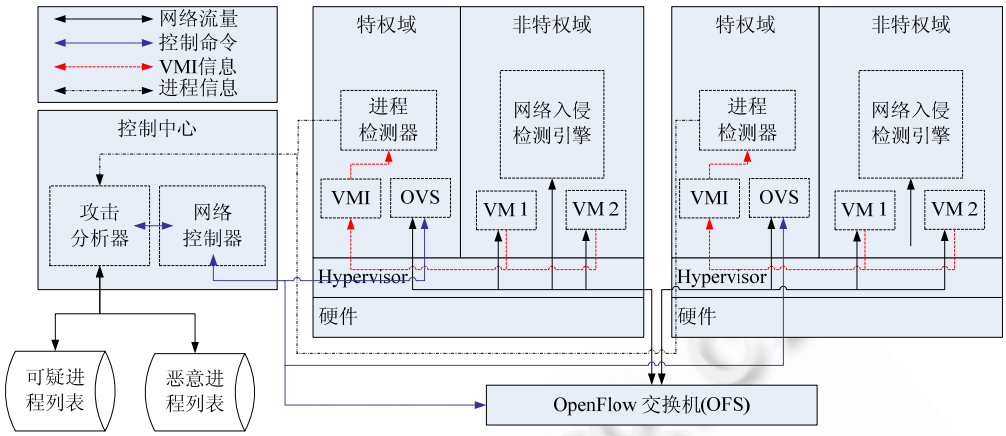


Fig.3 Architecture of the system

图 3 系统总体框架图

本文中,仅以 XEN 虚拟化平台为例进行说明.在云主机内,所有网络流量都由 OVS 进行处理和转发,并且每个 OVS 中都存在流表,指示应该如何转发流量.而在云主机之间,由 OFS 根据内部的流表进行处理和转发.由于 OFS 和 OVS 的数据转发功能和路由控制功能相互分离,故在本系统中,OFS 和 OVS 仅完成数据转发这一功能,而路由控制则由 NC 来完成.一旦一个网络连接请求被创建,该请求的流量就会到达 OVS 并等候处理.由于这是一个新的连接,故 OVS 内不存在相应的流表项.此时,OVS 会将流量发送给控制器 NC,向 NC 询问处理方式.由于 NC 在主机外部,与该主机以 OFS 相连接,故这一过程需要经过 OFS.成功发送到 NC 后,NC 记录下流量的信息,并继续转发给 AA,由 AA 调用 PI 和 VMI 模块获取发起请求的 VM 及内部进程的信息,并做出决策,如转发到目的地、重定向到 NIDS 处进行深层检查或是直接丢包.AA 将命令传递给 NC,由 NC 控制 OVS/OFS 具体执行流量处理操作.同时,将该操作写进流表.这种处理方式可以在恶意 VM 试图与其他 VM 建立连接时及时阻断连接,保证其他 VM 不会受到威胁.

3.2 系统结构设计

本节详细介绍本系统中各重要部件的工作原理及其实现方式.

3.2.1 攻击分析器 AA

AA 是系统中的信息处理和决策制定中心.其工作流程如图 4 所示.

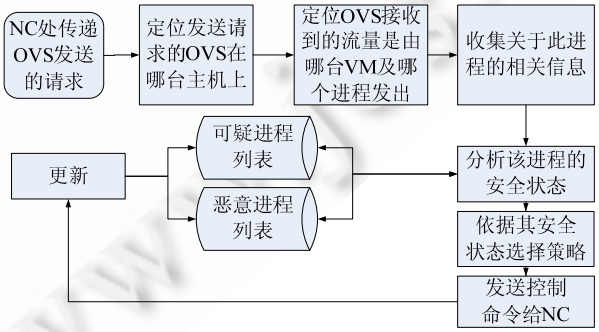


Fig.4 Workflow of AA

图 4 AA 工作流程图

当 NC 接收到由 OVS 发送过来的流量时,会记录下该 OVS 的 datapath ID 以及发送数据包的 VM 对应在 OVS 上的虚拟网卡号 ofport 和具体发包的进程所使用的端口号(source port),然后将其转发给 AA.AA 通过这些信息,并调用 VMI 模块,能够确定该请求来源于哪台 VM 里的哪个进程.定位发起连接请求的 VM 和进程的过程如下:

1) OFS 根据 datapath ID,确定 OVS 所属的云主机;
2) OVS 根据数据包来源的 ofport,确定该数据包来源于何 VM;
3) 若只检测 VM 的安全状态,则直接返回该 VM 的 ID 给 AA;
4) 若还需了解具体发送数据包的进程,则调用 PI 和 VMI 部件,列出该 VM 内的所有进程(包括隐藏进程)以及进程所使用的网络端口;
5) 将之前记录的 source port 与步骤 4)中的所有进程的网络端口号一一比对,找到符合的进程号 pid,再将 VM 的 ID 和进程的 pid 返回给 AA.

然后,AA 可通过 PI 模块获取该 VM 以及该进程的安全信息并进行分析,进而在先验知识(可疑进程列表、恶意进程列表)的帮助下做出决策.如 PI 模块获知发出网络连接的进程为隐藏进程,或在可疑进程列表中出现,则有理由相信该进程发出的数据包需要得到更深层的检查;若出现在恶意进程列表中,则可直接丢弃它发出的包;若以上情况均未发生,则相信此进程是正常的,其数据包可以转发.可疑进程列表和恶意进程列表是可以动态更新的,因为 AA 每次得到 PI 的反馈后,若确认某进程可疑或恶意,便会更新这两个列表.值得一提的是,AA 作为一个信息处理和决策模块,它本身可以接收不同安全组件的信息,包括 NIDS 的警告信息,或是其他查毒软件发送来的报告,进而根据所获得的信息做出相应决策.下面仅以隐藏进程检测的结果以及可疑进程列表、恶意进程列表作为其中一个说明范例:

- 可疑进程列表(suspect process list,简称 SPL):该列表列举了拥有漏洞数据库中漏洞的进程,此列表中的进程都是可疑的,因为它的漏洞很容易被攻击者利用,故需要得到更多的关注;
- 恶意进程列表(malicious process list,简称 MPL):该列表列举出的进程是已被确定为恶意的进程,故此列表中的进程应被立即禁止与其他 VM 的通信.

与“终止-检测-转发”模型相对应,AA 做出的决策包括两个层次,分别是 VM 级以及进程级.

- 在 VM 级,具体措施发生在网络第 2 层——数据链路层.由于 OVS 通过虚拟网桥将两台 VM 的网卡连接起来,并且不同网桥上的 VM 在数据链路层是相互隔离的,故只需禁用某 VM 的虚拟网卡所对应的 OVS 上的端口,即可将该 VM 隔离;也可修改流表项将从该 VM 对应的 OVS 端口出来的所有流量重定向到 NIDS 处进行检查.该决策尽可能地杜绝了攻击行为的发生,但同时也影响了 VM 内正常服务的运行,可能会给用户带来不便;
- 在进程级,通过 PI 模块获得进程的网络信息后,通知 NC 控制 OVS,每当接收到来自某 VM 内某可疑进程发出的流量,则将其丢弃或是重定向到 NIDS.进程级的决策具备更细的控制粒度,能够在有效遏制恶意行为的同时保证 VM 的其他网络服务不受影响.

总而言之,AA 的策略包括:

1) VMIsp:即 VM 级的检测.此策略将一个可疑的 VM 发出的所有流量导入到一个 in-line 模式下的 NIDS 处进行检查;
2) VMIso:即 VM 级的隔离.此策略将一个被怀疑的 VM 的所有网络流量阻塞掉,使得该 VM 与其他 VM 相互隔离;
3) PrIsp:即进程级的检测.此策略仅将特定 VM 内的特定进程发出的流量重定向到 NIDS 处进行检查;
4) PrIso:即进程级的隔离.此策略阻止某可疑 VM 内的可疑进程与其他 VM 的进程进行网络通信.该方法仅停止某一进程的网络服务,并不影响同 VM 内的其他进程.

3.2.2 网络控制器 NC

NC 是本系统能够利用 OpenFlow 协议构建可编程网络的关键部件.OVS 作为直接处理 VM 之间流量的交

换机,由 NC 控制.本系统中,NC 从 OVS 处获得流量的信息并转发给 AA,再接收从 AA 处传递回来的处理命令.然后,NC 根据此命令创建一个流表项插入到 OVS 的流表中,从而操控流量的去向.

图 5 说明了 NC 从 OVS 接收到连接请求直到创建流表项以操控流量的工作过程.

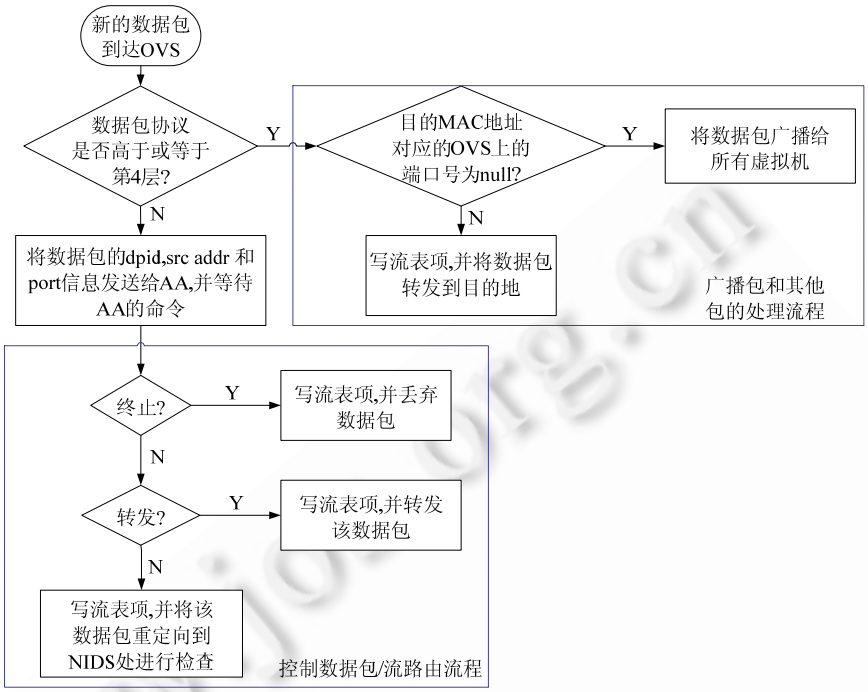


Fig.5 Workflow of NC

图 5 NC 工作流程图

为将流量重定向到 NIDS 中进行深层数据包检测,NIDS 的两个虚拟网卡会被添加到被监控网络所属的 OVS 上,并设置成混杂模式用来嗅探网络流量.同时,配置 NIDS,其中一个虚拟网卡仅作为接收流量的入口(inport),另一个网卡仅作为正常流量的出口(outport).在这种状态下,广播流量易引起自循环,因为广播流量会发送到每个 VM 中,当此流量进入到 NIDS 入口时,NIDS 会判断是否正常:若为正常,便让其通过出口又再回到 OVS 上重新进行广播.因此,需要特别处理这种情况,避免引起自循环,耗费网络资源.防止自循环的伪算法如下:

算法 1. 自循环避免算法.

输入:数据包包头信息,NIDS 的 inport,NIDS 的 outport;

```
1.  if (数据包从 OVS 其中一个 ofport 发出)
        //如果数据包不是从 OVS 的 ofport 发出,则为非法数据包,应丢弃
2.    if (数据包中目的 MAC 地址对应 OVS 的 ofport==null)
3.      //代表这是一个广播包
4.      if (数据包源端口==NIDS 的 outport && 数据包目的端口==NIDS 的 inport)
            //如果数据包由 NIDS 发出,目的地也是 NIDS,则应丢弃该包,防止自循环发生
5.              丢包
6.      else
7.              将数据包转发到 OVS 的每一个端口
8.    end if
9.  else
```

10. 丢包

11. **end if**

NC 的工作就是根据 AA 的决策处理流量的转发路由.对于协议低于第 4 层的数据包,NC 会根据 AA 传回来的命令:终止、重定向或正常转发,在 OVS 中创建流表项并填充转发路由.若是终止命令,则目的地填 null;若为重定向命令,则为 NIDS 在 OVS 上的 outport 号;若为正常转发,则为原目的地址对应的 OVS 上的端口(ofport)号.若数据包协议高于第 4 层或是一个广播包,则 NC 仅检查该包的目的地址是否为空(null):若为 null,则将其广播到所有 VM 中;若不为 null,则按照其目的地址进行转发.

3.2.3 进程检测器 PI 和 VMI

PI 模块主要用于在 VMI 模块的协助下检测云主机内每台 VM 内部的进程状况,获知当前 VM 以及其中进程的安全状态.为保证检测工具自身的安全和检测结果的完整性,本系统将 PI 模块置于被监控 VM 外的 Domain0 中,与 VMI 模块协同合作.

PI 模块内包含了两个子模块:守护程序(daemon)和语义翻译模块(semantic translation module).如图 6 所示:守护程序主要用来接收来自 AA 的命令(如检测隐藏进程、判断发包的是何进程或进程的网络连接状态等)并予以执行,然后将结果返回给 AA;而语义翻译模块的存在是为了解决语义鸿沟(semantic gap)的问题.在虚拟化平台中,虚拟机监视器(virtual machine monitor,简称 VMM),也称为管理程序(hypervisor),仅能读到 VM 内部的虚拟硬件信息.这些硬件信息是大量的不可读的原数据(raw data),不具备客户操作系统内部的数据结构,使得这些原数据缺少有意义的语义信息,因而也很难通过这些原数据直接获知当前系统的运行状态.这种状况通常称为语义鸿沟问题.

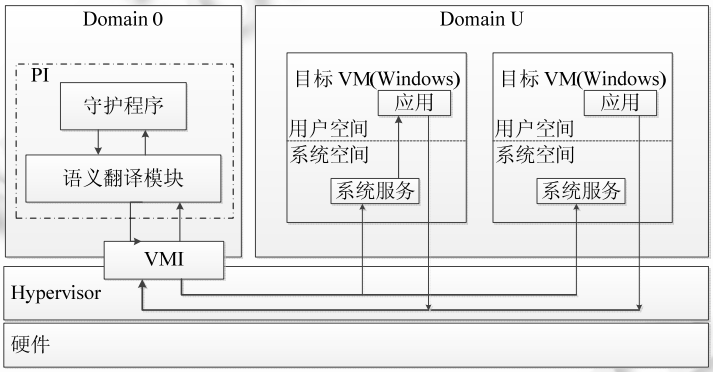


Fig.6 Structure of PI module and VMI module

图 6 PI 模块和 VMI 模块结构图

由于 PI 模块放置在被监控 VM 外部,故需要 VMI 模块帮助从外部获取 VM 内的信息.VMI 模块负责通过两层地址转换,将 VM 应用程序的虚拟地址空间转换到主机的物理地址空间,进而可以直接在主机中读到 VM 内的信息.其中,第 1 层地址转换是将客户机(即 VM)虚拟地址(guest virtual address,简称 GVA)映射到客户机物理地址(guest physical address,简称 GPA),第 2 层地址转换是将 GPA 翻译成主机物理地址(host physical address,简称 HPA).通过在 VM 外部读 HPA 地址中的内容,即可获知相应的 VM 内 GVA 的内存信息.完成地址翻译后,VMI 模块会将读到的内容返回给 PI 进行检测.

本系统中,隐藏进程检测以及进程的网络状态是非常重要的部分,对于判断 VM 及特定进程的安全性有着重要意义.下面以操作系统为 Windows 的 VM 为例,分析隐藏进程检测和网络状态检测.

在隐藏进程检测过程中,PI 首先利用 VMI 模块寻找 VM 内的活动进程链.在此之前,需要找到 Windows 加载的内核模块链表表头 PsLoadedModuleList,然后遍历内核模块链表,找出 ntoskrnl.exe.找到操作系统内核文件 ntoskrnl.exe 以及其加载到内存中的基地址(BaseAddress)后,通过分析 ntoskrnl.exe 的 PE(portable executable)文

件,在其中的.edata 节的导出地址表中得到 PsInitialSystemProcess 的相对虚拟地址(relative virtual address,简称 RVA).PsInitialSystemProcess 是指向系统 system 进程的 EPROCESS 结构的指针,其在内存中的虚拟地址 VA 可由 VA=RVA+BaseAddress 计算得出.在 Windows 中,活动进程的 EPROCESS 结构会相互连接成一个双向循环链表,故得到 PsInitialSystemProcess 地址并寻找到 system 的 EPROCESS 后,即可确定活动进程链.

由于隐藏进程会通过各种方式隐藏自身,如采用直接操作内核对象(direct kernel object manipulation,简称 DKOM)的方式,则该进程的 EPROCESS 将不会出现在上述的活动进程链中.由于 Windows 的调度单元是线程,故该进程即使不在活动进程链中,也不会影响 OS 的调度和进程的运行.但若将进程从线程链表上摘除,则无法获得 CPU 时间,进而无法运行,故遍历线程调度链即可发现从活动进程链上脱链的所有进程.经过比对在用户态获取的进程(如采用 ZwQuerySystemInformation 函数枚举进程)和内核态的活动进程链以及线程调度链,可检测出隐藏进程.

关于获取进程的 socket 信息和网络连接状态,PI 需要定位分别包含 socket 信息和网络连接信息的 _ADDRESS_OBJECT 和_TCPT_OBJECT 两条链表.在此之前,需先找到内核模块 tcpip.sys,因为 tcpip.sys 的 PE (portable and executable)结构体中的 AddrObjTable 和 TCBTable 分别指向了两条链表的基地址.除了使用中的 socket 信息和活跃的网络连接信息外,PI 也能获取该 socket 和连接所属的进程.

3.2.4 网络入侵检测系统 NIDS

NIDS 用于对无法确认是否为恶意的可疑流量进行更彻底的深层数据包检测.本系统的工作在于对网络流量的捕获以及路由控制,而不在于改良现有的流量检测算法,因此直接利用了开源工具 snort[18,19],将其运行在 in-line 模式下,实行数据包检测.在本系统中,NIDS 被放置在非特权域 UPD 中,这样做的好处是:

1) 部署动态化和自适应化:NIDS 作为一个虚拟应用部署,重配置并动态迁移到任意网段中会变得相对容易,且无需复杂的网络配置过程.NIDS 在 in-line 模式下工作,检测并过滤恶意流量,但无需将其配置成一个 in-line 的网关;
2) 减轻特权域 PD 的负载:由于特权域 PD 需要管理和配置内存和硬件资源的访问,若 PD 的负载过大,则会严重影响 VM 访问内存和硬件资源,故应尽可能地减轻 PD 的负载.NIDS 的主要作用是对重定向到此处的流量进行深度数据包检测(deep packet inspection,简称 DPI).这一过程十分耗费资源,故不适合放在 PD 内.

NIDS 自身的安全也需要考虑在内.尽管 NIDS 处于 UPD 内,但是基于以下假设,仍然可以认为它是安全的:

1) 假设每台云主机的管理器 Hypervisor 是受保护的,每个 VM 都能完全地相互隔离.恶意的 VM 不能通过 Hypervisor 攻击 NIDS;
2) 尽管 NIDS 处于 UPD 中,能够从 PD 中被访问,但只有云管理员才能在其中维护软件的安装行为并运行管理任务.假设云管理员是可信的,那么 NIDS 仅能由云管理员从 PD 安全地访问,未授权者无法访问;
3) NIDS 对于网络内的其他 VM 是不可见的,仅有 NC 可以通过 OVS 和 OFS 的控制通道去控制和主导 NIDS 的行为.而控制通道是与 OVS、OFS 传输网络流量的数据通道相互分离的,故控制通道的安全性值得信任.

4 实验测试与分析

进行实验测试工作的云主机硬件配置为:8 个 Intel(R) Xeon(R) 2.33GHz 的 CPU,32768MB 的内存.主机中有 8 台 VM 在运行,其中 1 台 VM 中运行着 snort,其余 7 台均为普通云用户所使用的 VM.实验部分将按照功能测试和性能测试两方面完成.

4.1 系统功能测试

本节利用 Xcap 工具模拟拒绝服务攻击(denial of service,简称 DoS)中的 SYN 洪泛[20]攻击(SYN flood).利用 TCP/IP 协议栈的漏洞,给攻击目标发送大量伪造的 TCP 连接请求,从而耗尽对方的资源(CPU 满负荷或内存不

足).在同一主机内,设置一台 VM 为攻击方,OS 为 Microsoft Windows XP SP3,另一台 VM 作为目标服务器,OS 为 Microsoft Windows Server 2003 Standard SP1,且未安装补丁.将 NIDS 即 snort 的两个网卡设置为混杂模式,并留一个网卡作为管理接口.各部分网络配置如图 7 所示.

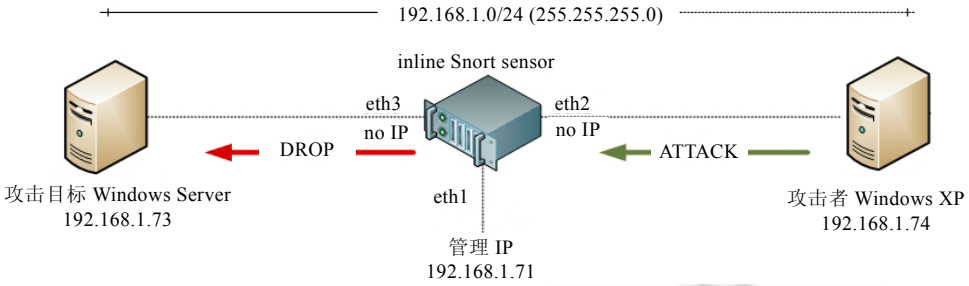


Fig.7 Configuration of test network

图 7 实验网络配置

在本系统中,攻击方发送的 TCP 连接请求将不会直接到达被攻击方,而是首先到达网络交换机(OVS,OFS)中.OVS 和 OFS 将该请求的流量转送到 NC 处加以检查.AA 会调用 PI 模块和 VMI 模块,追溯该流量的来源以进行检测判断.由于本攻击实为模拟的,发送请求的 Xcap 工具是一个正常程序,为使其代表恶意程序,本实验用 FU rootkit 将其隐藏.PI 模块检测时,会发现此进程为隐藏进程,同时将结果交由 AA 处理.隐藏进程的检测结果如图 8 所示.

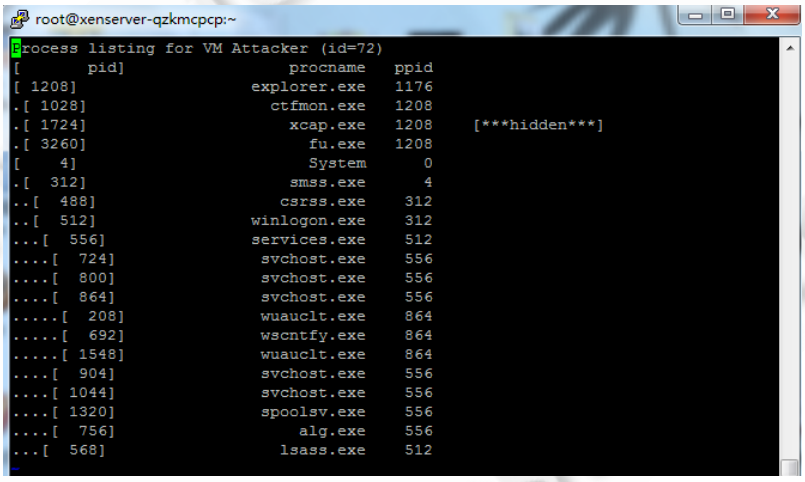


Fig.8 Result of hidden process detection

图 8 隐藏进程检测结果

此时,AA 判定该进程为可疑,指挥 NC 将其发出的连接请求流量重定向到 NIDS 处进行深层检测.此时,在 NIDS 所在系统中能够嗅探到此流量,如图 9 所示.由于攻击方的 Xcap 工具会连续发包,所有包经上述处理均会到达 snort 处,直至 snort 判定这类数据包已构成了 syn flood 攻击.

在 NIDS 中事先做过处理,添加一条关于虚假 TCP 连接请求的规则.规则如下:

```
drop tcp any any>192.168.1.74 any ( \
    msg: “TCP sync flood”;
    flow: established, to_server; \
    detection_filter: track by_src, count 10, seconds 60;
```

```
        sid:1000001;
        rev:1;)
rate_filter \
        gen_id 135, sig_id 1, \
        track by_dst, \
        count 10, seconds 60, \
        new_action drop, timeout 6000, \
        apply_to 192.168.1.74
```

故当流量定向到 NIDS 处时,NIDS 会根据规则将其直接丢弃并发出警告.然后,AA 根据 NIDS 的警告确定攻击方的 Xcap 程序是恶意程序,将其加入恶意程序列表 MPL,并告知 NC 向 OVS 中写入流表项,阻塞该进程发出的所有数据包.此时,被攻击方不会收到任何来自攻击方发来的连接请求,保证了它的安全.OVS 对应的流表项如图 10 所示(图中第 1 条规则).图中,in_port=4 代表攻击者发送的数据包,而 actions=out_put:14 代表将数据包转发到端口号为 14 的 VM,即为 NIDS.

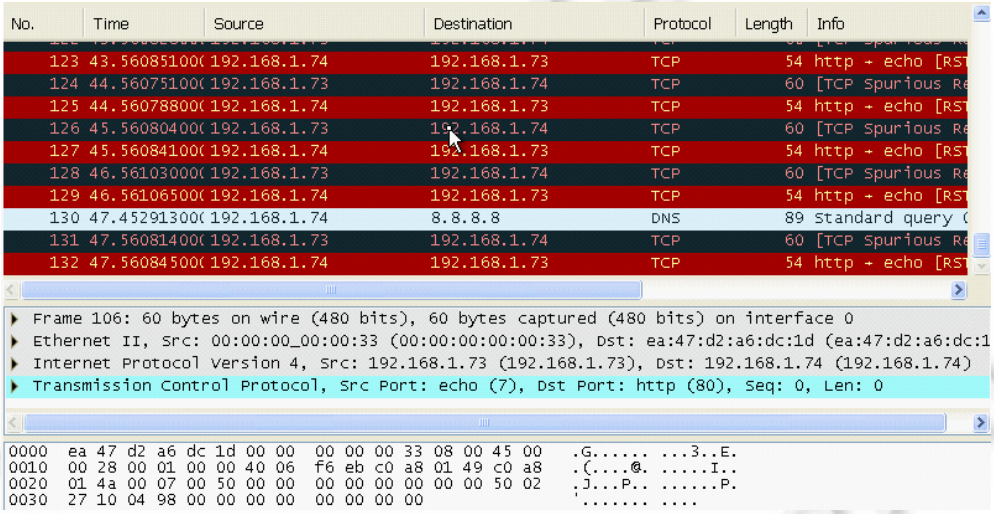


Fig.9 Flow captured in NIDS

图 9 NIDS 处捕获到的流量

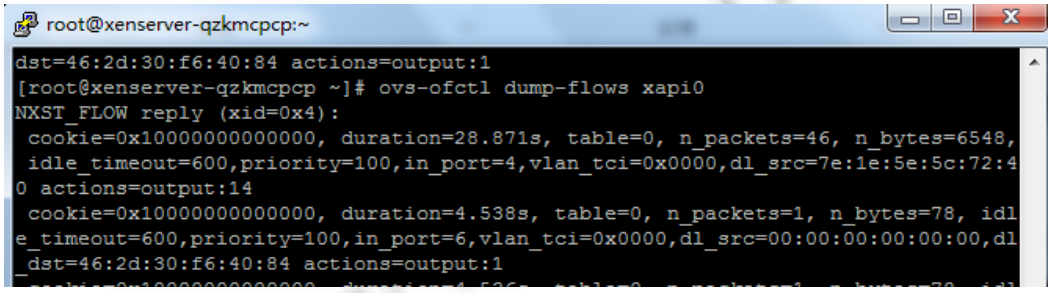


Fig.10 OVS's flow table

图 10 OVS 流表

4.2 系统性能测试

本节将测试云用户以不同的发送速率向外(仍在云内网中)发送 TCP 包时,运行本系统所带来的丢包率以

及不运行本系统情况下的丢包率.同时,还将对比运行本系统的 CPU 负载和不运行本系统的 CPU 负载.

首先测试以不同速度发送包长为 1 514Bytes 的 TCP 包时,运行本系统和不运行本系统时的丢包率,结果如图 11 所示.

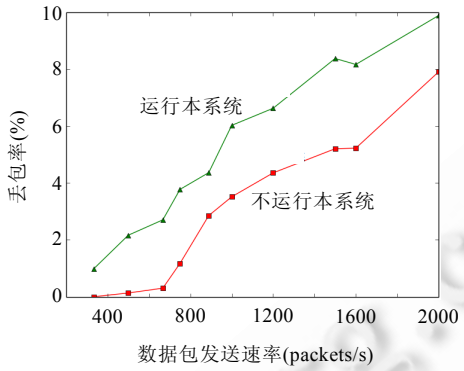


Fig.11 Comparison of drop rate between situations of running our system and without our system

图 11 运行本系统与不运行情况下丢包率的对比

图 11 显示:随着发包速率的增大,两种情况下的丢包率均在逐步增长.由于本系统中 OVS 转发包之前还需对其发送方进行检查等操作,因此在运行本系统时,丢包率会比另一种情况下的丢包率略高.然而实验结果也表明:两种情况下的丢包率比较接近,最坏情况下,本系统所带来的额外的网络负载为 3.17%.

然后对比运行本系统和不运行本系统两种情况下的 CPU 负载状况,对比分为 7 组,分别为 1~7 台 VM 同时向外(仍在云内网中)发送 TCP 包,从开始发包到持续 1 分钟时系统的 CPU 平均占用率.测试发包 1 分钟内的平均 CPU 占用率的原因是:VM 刚开始发包时,OVS 需要与控制器连接,判断当前数据包的走向.此时的系统负载是最大的,而本系统中 PI 需要追溯发送源并进行检测,因此与不运行本系统相比负载更大;1 分钟之后,OVS 的流表项已经由控制器填好,可以直接转发数据包,因此运行本系统的负载情况与不运行本系统的负载情况此时应该一致,故不予考虑.实验结果如图 12 所示.

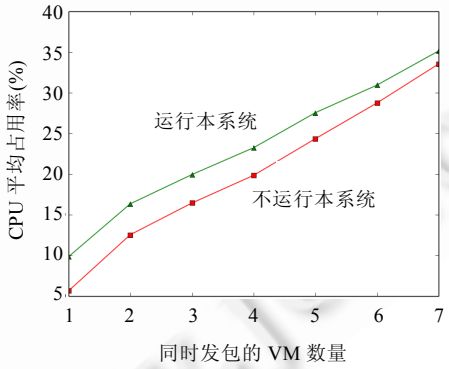


Fig.12 Comparison of CPU load between situations of running our system and without our system

图 12 运行本系统与不运行情况下的 CPU 负载对比

图 12 表明:随着发送数据包的并发数的增加,两种情况下的 CPU 负载会增长,而运行本系统的情况下负载会相对更大.然而随着并发数的增长,运行本系统的 CPU 负载会逐渐靠近不运行情况下的 CPU 负载.出现此种现象的原因是:对于本系统,PI 对数据包来源的检测是串行的,与发送并发数无关,故它所消耗的 CPU 负载相对比较稳定.因此,当并发数变大时,PI 所占用的 CPU 比例会逐渐变小,显示出来的结果便是两种情况的 CPU 负载

越来越接近.

5 结束语

本文提出了一种具有内网防火墙特点的虚拟化网络纵深防御系统,融合了软件定义网络以及虚拟机自省技术等,能够及时捕获可疑或恶意流量,反向定位发起流量的 VM 和进程,利用隐藏进程检测和进程网络信息获取等安全技术确定流量的安全性,运用网络的可编程性对特定进程发起的流量进行控制,在保证网络服务质量的同时,有效阻止了恶意行为的扩散,保护了系统安全.

References:

[1] Vokorokos L, Baláž A. Host-Based intrusion detection system. In: Proc. of the 14th Int'l Conf. on Intelligent Engineering Systems. Piscataway: IEEE, 2010. 32−36.

[2] Kothari S, Parmar H, Das E, Panda N, Ahmed A, Marchang J. Host based intrusion detection system. In: Proc. of the Int'l Conf. on Mechanical Engineering and Technology. London, New York: ASME Press, 2011. 182−185. http://ebooks.asmedigitalcollection.asme.org/content.aspx?bookid=484&sectionid=38791823&resultClick=1

[3] Yassin W, Udzir NI, Muda Z, Abdullah A, Abdullah MT. A cloud-based intrusion detection service framework. In: Proc. of the Int'l Conf. on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec). Kuala Lumpur: IEEE, 2012. 213−218. [doi: 10.1109/CyberSec.2012.6246098]

[4] Laureano M, Maziero C, Jamhour E. Intrusion detection in virtual machine environments. In: Proc. of the 30th Euromicro Conf. Washington: IEEE, 2004. 520−525. [doi: 10.1109/EURMIC.2004.1333416]

[5] Gupta S, Kumar P, Abraham A. A profile based network intrusion detection and prevention system for securing cloud environment. Int'l Journal of Distributed Sensor Networks, 2013,2013:131−142. [doi: 10.1155/2013/364575]

[6] ONF Market Education Committee. Software-Defined networking: The new norm for networks. ONF White Paper, 2012.

[7] Pettit J, Gross J, Pfaff B, Casado M. Virtual switching in an era of advanced edges. In: Proc. of the 2nd Workshop on Data Center—Converged and Virtual Ethernet Switching (DC-CAVES). 2010. http://openvswitch.org/papers/dccaves2010.pdf

[8] McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008,38(2):69−74. [doi: 10.1145/1355734.1355746]

[9] Shin S, Porras P, Yegneswaran V, Fong M, Gu GF, Tyson M. FRESCO: Modular composable security services for software-defined networks. In: Proc. of the Network and Distributed System Security Symp. in 2013. Washington: Internet Society, 2013. 314−329. http://www.csl.sri.com/users/vinod/papers/fresco.pdf

[10] Jafarian JH, Al-Shaer E, Duan Q. Openflow random host mutation: Transparent moving target defense using software defined networking. In: Proc. of the 1st Workshop on Hot Topics in Software Defined Networks. Helsinki: ACM Press, 2012. 127−132. [doi: 10.1145/2342441.2342467]

[11] Shin SW, Gu GF. CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?). In: Proc. of the 2012 20th IEEE Int'l Conf. on Network Protocols. Austin: IEEE, 2012. 1−6. [doi: 10.1109/ICNP.2012.6459946]

[12] Garfinkel T, Rosenblum M. A virtual machine introspection based architecture for intrusion detection. In: Proc. of the Network and Distributed Systems Security Symp. Washington: Internet Society, 2003. 191−206. http://www.isoc.org/isoc/conferences/ndss/03/proceedings/papers/13.pdf

[13] Jiang XX, Wang XY, Xu DY. Stealthy malware detection through VMM-based "out-of-the-box" semantic view reconstruction. In: Proc. of the 14th ACM Conf. on Computer and Communications Security. Alexandria: ACM Press, 2007. 128−138. [doi: 10.1145/1315245.1315262]

[14] Jones ST, Arpaci-Dusseau AC, Arpaci-Dusseau RH. Antfarm: Tracking processes in a virtual machine environment. In: Proc. of the USENIX Annual Technical Conf. New York, ACM, 2006. 1−14. http://dl.acm.org/citation.cfm?id=1267360

[15] Jones ST, Arpaci-Dusseau AC, Arpaci-Dusseau RH. VMM-Based hidden process detection and identification using Lycosid. In: Proc. of the 4th ACM SIGPLAN/SIGOPS Int'l Conf. on Virtual Execution Environments. Seattle: ACM Press, 2008. 91−100. [doi: 10.1145/1346256.1346269]

[16] Lombardi F, Di Pietro R. KvmSec: A security extension for Linux kernel virtual machines. In: Proc. of the 2009 ACM Symp. on Applied Computing. Honolulu: ACM Press, 2009. 2029−2034. [doi: 10.1145/1529282.1529733]

[17] Dai YH, Qi Y, Ren JB, Wang XG, Shi Y, Xuan Y. Trochilidae: Light weight and high performance virtual machine monitor for many-core. Journal of Frontiers of Computer Science and Technology, 2012,6(12):1126−1135.

[18] Roesch M, Green C. Snort users manual. Snort Release, 2012,1(3):10−28.

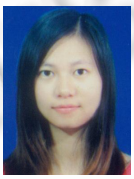
[19] Wuu LC, Hung CH, Chen SF. Building intrusion pattern miner for Snort network intrusion detection system. Journal of Systems and Software, 2007,80(10):1699−1715. [doi: 10.1016/j.jss.2006.12.546]

[20] Lau F, Rubin SH, Smith MH, Trajković L. Distributed denial of service attacks. In: Proc. of the IEEE Int'l Conf. on Systems, Man, and Cybernetics in 2000. Nashville: IEEE, 2000. 2275−2280. [doi: 10.1109/ICSMC.2000.886455]

崔竞松(1975－),男,湖北武汉人,博士,副教授,CCF 会员,主要研究领域为云计算,位置服务,信息安全.
E-mail: jscui@whu.edu.cn

郭迟(1983－),男,博士,讲师,CCF 会员,主要研究领域为位置服务,云计算.
E-mail: guochi@whu.edu.cn

陈龙(1985－),男,博士,讲师,主要研究领域为 3S 集成,云计算.
E-mail: chenl46@mail.sysu.edu.cn



张雅娜(1990－),女,硕士生,主要研究领域为信息安全.
E-mail: 1021998325@qq.com

Dijiang HUANG(1971－),男,博士,教授,博士生导师,主要研究领域为网络安全系统,认证系统,网络攻击分析.
E-mail: dijiang@asu.edu