安全+
SECURITY
2013/12 总第 023
技术版 ▶▶ 与安全人士分享技术心得 Share technique experience with security professionals

★ 本期焦点

浅谈网络虚拟化安全

DDoS放大攻击原理及防护方法

一种有效的CSRF漏洞检测技术

绿盟科技DDoS威胁态势报告

NSFOCUS

本期看点 HEADLINES

主办：绿盟科技
策划：绿盟内刊编委会
地址：北京市海淀区北洼路4号益泰大厦三层
邮编：100089
电话：(010)6843 8880-8667
传真：(010)6872 8708
网址：www.nsfocus.com

Nsmagazine@nsfocus.com

2013/12 总第 023

安全+ SECURITY+

目录 CONTENTS

道魔之争，下一代安全帮您重获主动

子墨子解带为城，以牒为械，公输盘九设攻城之机变，子墨子九拒之。公输盘之攻械尽，子墨子之守圉有余。公输盘诎，而曰“吾知所以距子矣，吾不言。”子墨子亦曰“吾知子之所以距我，吾不言。”

不管您关注与否，注入、跨站、假冒、大规模拒绝服务、网络钓鱼、定向梭镖、梭镖社工复合攻击、水坑攻击、多水坑战术、USB 摆渡、BIOS 和固件 Rootkit、高频声波联网突破物理隔离……这些令专业人士都目不暇接的各种网络攻击入侵时时占据专业和大众媒体的头版，不断给大家的心理带来震撼。

另一方面，反病毒、高级恶意软件防护、下一代防火墙、下一代入侵检测和入侵防御、WAF、动态和静态应用安全测试、蜜网、取证分析以及其他各种对抗高级持续威胁 APT 的技术和产品，也不断涌现出来。

九攻九距，胜负几何？

在绿盟科技研究院发布的下一代安全研究报告中认为，威胁汹汹，多为利来。当攻击收益远高于成本，此威胁必扶摇而上。防护的目的在于消减和控制风险，亦有成本和潜在损失之比。 防护成本过高，则组织宁可接受或转移风险。只要防护成本明显低于潜在损失，组织的理性选择必是部署防护措施，消除潜在损失。

将攻击和防护放在一个图像里看，应该承认在高级持续威胁的范畴里，威胁方相对于防护方处于暂时成本优势区域。 攻方优势的其中一个原因是因为威胁方主动，防护方被动，即威胁方可选择单点突入，防护方则需要全线布防，防护设备和技术在协同方面的不足也限制了防护效率和响应时效性，不能及时在有效部位使用针对性的“守围”来对抗“攻械”。

有效消除“攻方成本优势”的重要途径包括威胁感知（了解敌情、知己知彼）、提高协同能力（见招拆招，九攻九距）、闭环运营（及时评价防护效果，优化防护措施）等。

在下一代威胁面前，安全防护体系注定要进入一个变革的年代。道魔之争，下一代安全帮您重获主动。

浅谈网络虚拟化安全

战略研究部　刘文懋

关键字：网络虚拟化　安全　SDN　OpenFlow

摘要：本文简要介绍了网络虚拟化的发展，分析了网络虚拟化方面的一些安全问题，并给出了一些可行的措施。

经过多年的技术准备和商业模式探索，云计算已然进入快速发展的阶段。当前有不少大型数据中心和企业 IT 系统可为云计算提供设施级别的强大支撑平台，然而业务快速增加给数据中心和大型企业的复杂网络管理带来了极大的挑战。网络虚拟化技术和软件定义网络（SDN，Software Defined Networking）技术以软件可编程的方式管理虚拟和实体网络资源，在很大程度上解决了上述问题，近年来得到了极大的关注。本文主要介绍了面向数据中心和企业级应用的网络虚拟化技术，并探讨网络虚拟化对于传统安全的挑战，以及可能的安全解决技术。

1. 网络虚拟化技术简介

在虚拟化应用中，大规模应用对计算和存储的天然需求使得计算虚拟化和存储虚拟化技术较为成熟，与之相比，网络虚拟化的相关技术还在开发阶段，远没有达到成熟的程度。不过随着应用规模不断变大和业务快速变化，实体和虚拟网络的融合、快速管理和可扩展性将成为巨大的挑战，网络虚拟化已受到越来越多的关注，预计其发展会进入快车道。

网络虚拟化将物理和虚拟网络资源整合成一个可管理的虚拟网络，特别 SDN 的出现给网络管理带来了颠覆性的变化：可通过编程的方式迁移虚拟机、动态组网，大大加快变更网络拓扑的进度。ONF 指出 SDN 具备三个核心特征 [1]: 控制平面和数据平面相互分离、智能和状态在逻辑上集中以及底层网络基础设施从应用中抽象出来。SDN 是层次化的集中控制架构，如图 1 所示：网络控制器处于 SDN

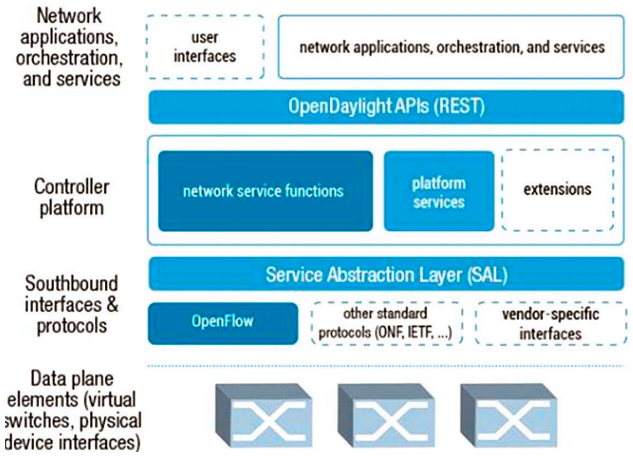

图 1 SDN 的应用实例

架构的中心，北向与应用连接，进行业务交互；南向连接底层的交换机，下发路由控制命令。功能上，网络中心的控制器掌握网络拓扑、数据转发的策略，而分布在各处的网络设备接受控制器的命令，执行数据转发和路由的行为。操作上，网络管理员可以随时更新网络拓扑、调整网络规模，或快速部署安全策略，如果操作失败或存在问题可方便地进行回滚，恢复到原来的状态，可见 SDN 确实提供了快捷方便的网络管理途径。

需要说明的是，网络虚拟化、SDN 和 OpenFlow 三者不是等价的。网络虚拟化可以不使用 SDN 技术，SDN 也不一定要用于虚拟化环境，但 SDN 的控制数据平面分离非常适合网络虚拟化的场景。同样地，OpenFlow 是一种可实现 SDN 场景的控制器 - 设备协议，但非惟一的 SDN 实现方式，还有 NVGRE 等技术。本文在讨论网络虚拟化时都是基于 SDN 的，所采用的协议标准是 OpenFlow。

1.1 主流的网络虚拟化解决方案

通过软件重构网络是大势所趋，不同厂商在 SDN 中根据自身情况制定各自的策略，总体来说，SDN 的研发领域可能会出现三足鼎立的局面：OpenFlow 标准的相关开源系、Cisco 主导的 Cisco One 系和 Vmware 主导的 SDN 系，参与三个体系的厂商关系如图 2 所示。

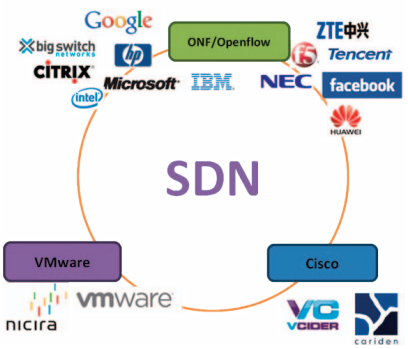

图 2 SDN 厂商阵营图

1.1.1 OpenFlow/ONF

当前参与量最大的开放 SDN 阵营是由 90 多家公司组成的非盈利组织 ONF（Open Networking Foundation），其成员不乏 IBM、Intel、Google 等重量级的公司，还有 BigSwitch、Citrix 等专注虚拟化的公司，此外还有国内的企业如腾讯、华为和中兴等。ONF 最重要的成果是 OpenFlow，OpenFlow 是网络控制器和交换机之间通信的协议，控制器通过统一的路由策略下发基于流的数据交换命令，实现了上层软件管理和更新路由表、底层交换机执行转发策略的软硬件分离模式。OpenFlow 协议处理数据的过程如图 3 所示，交换机在遇到未知模式的数据包时，会向控制器上传相关特征，控制器检查自己的策略库，生成数据交换的模式，然后将这些模式以控制信息的形式下发到对应的交换机，从而让底层的交换机可以根据全局的路由策略执行相应的数据交换。

由于 OpenFlow 定义了控制器和交换机间的通信协议和安全通道，并规定交换机应遵循的规则，交换机只执行数据交换的功能，所以各厂商容易开发出支持 OpenFlow 的交换机，如 Nicira 的虚拟交换机 Openvswitch 和 NEC、IBM 等支持 OpenFlow 的实体交换机。对应地，控制器负责整个 SDN 中的网络拓扑管理、数据包路由决策、QoS 管理和安全控制等复杂的功能，所以控制器是整个网络的核心，当前比较著名的控制器有 BigSwitch 的 FloodLight、Nicira 的 NOX 和 NEC 的

ProgrammableFlow Controller 等。

OpenFlow 只定义了控制器 - 交换机的南向通信标准，没有给出应用 - 控制器的北向接口标准，所以相关厂商在针对控制器开发自己的网络应用时，需要考虑不同控制器的应用接口。

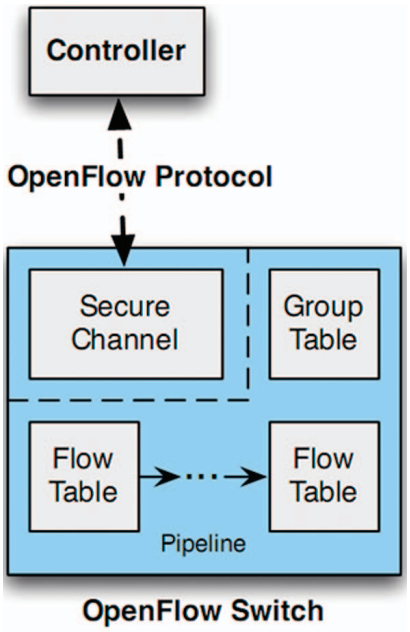

图 3 基于 OpenFlow 协议的数据交换

1.1.2 Cisco ONE

2012 年 7 月，Cisco 公司推出了自己的 SDN 计划“思科开放式网络环境（Cisco ONE，Open Network Environment）”。Cisco ONE 在实现网络环境的可编程性的同时，有更大的战略规划：它将 SDN 定位于五个目标市场，即学术和研究、企业、服务提供商、云服务提供商和数据中心，并认为 OpenFlow 只是其中的学术和研究市场。Cisco ONE 是对 OpenFlow 功能的扩展，包括 Cisco ONEPK 开发套件和 Nexus 1000V 等虚拟覆盖网等技术。其中 Nexus 1000V 包含了控制器 Virtual Supervisor Module (VSM) 和交换机 Virtual Ethernet Module (VEM)，实现了完整的 SDN 功能。

1.1.3 VMware/Nicira

老牌虚拟化巨头 VMware 在网络虚拟化方向也有大量投入。早在做研发服务器虚拟化时，VMware 就使用虚拟网桥和 NAT 的方式建立了宿主机和虚拟机之间的虚拟网络。在最新的企业级虚拟化产品 vSphere 中，网络虚拟化解决方案要有两方面：管理虚拟网络中的数据和流量的 Network I/O Control，以及集中式管理、监控和可视化虚拟网络的 Distributed Switch。

此外，VMware 于 2012 年收购了 Nicira Networks 公司，后者主导开发了虚拟交换机项目 Openvswitch。可预见 Vmware 将 Nicira 的网络控制器等产品集成到 vSphere 后，会在 SDN 方向进一步发力，保持其在虚拟化市场中领先的地位。

2. 网络虚拟化面临的安全挑战

虚拟化给数据中心和企业网络带来了新的问题和挑战。一方面，传统的安全产品和安全解决方案无法解决在虚拟化后出现新的网络安全问题；另一方面，网络虚拟化自身也面临一些安全问题。

网络在虚拟化后主要面临的问题有：

- 物理安全设备存在观测死角

虚拟机与外界存在数据交换，在虚拟化环境中的数据流有两类，即跨物理主机的 VM 数据流和同一物理主机内部的 VM 数据流。前者一般通过隧道或 VLAN 等模式进行传输，现有的 IDS/IPS 等安全设备需要在所有的传输路径上进行监控，后者只经过物理主机中的虚拟交换机，无法被实体的安全设备监控到，成为整个安全系统的死角。攻击者可以在内部虚拟网络中发动任何攻击，而不会被安全设备所察觉。如图 4 中

攻击者在虚拟机 VM1 中攻击 VM2，数据流量没有经过物理交换机，也不会传输到防火墙和 IDS。所以虚拟化改变了数据的流向，增大了物理设备不可见的区域，增加了整个虚拟化网络的安全管理难度。

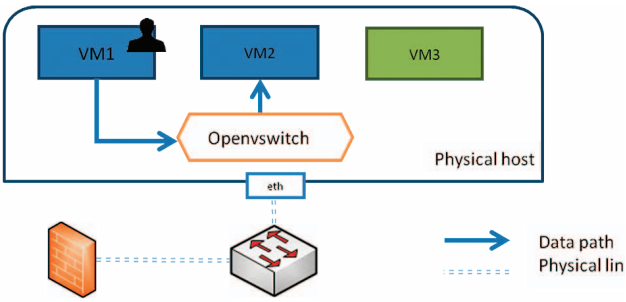

图 4 物理安全设备无法观测到内部虚拟网络的数据交互

■ 虚拟网络的数据流难以理解

虽然安全设备无法获得物理主机内部的 VM 间的数据包，但可以获取跨物理主机间 VM 的数据流。尽管如此，传统的安全设备还是不能理解这些数据流，也就无法应用正确的安全策略。例如在图 5 中，租户 X 和租户 Y 分别在两台物理主机上租用了一台虚拟机，当租户 X 从 VM1 向 VM3 发数据包时，防火墙能接收到物理主机 1 到物理主机 2 的数据包，但不知道到底是租户 X 还是租户 Y 的程序在发送数据包，也不知道是哪两台 VM 在通信。此外，很多虚拟机之间的数据包是经过 GRE 隧道传输的，所以传统的网络安全设施可能不能解析这些封装后的数据流。

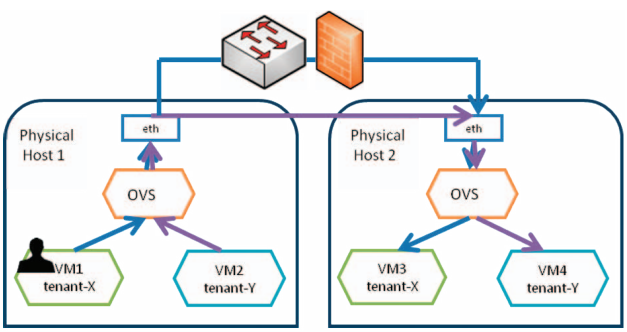

图 5 物理安全设备不能理解跨物理主机间数据流

■ 安全策略难以迁移

虚拟化解决方案的重要优点是弹性和快速，例如当 VM 从一台物理主机无缝快速地迁移到另一台物理主机时，或当增加或删除 VM 时，网络虚拟化管理工具可快速调整网络拓扑，在旧物理网络中删除 VM 的网络资源（地址、路由策略等），并在新的物理网络中分配 VM 的网络资源。相应地，安全解决方案也应将原网络设备和安全设备的安全控制（ACL 和 QoS）跟随迁移，然而现有安全产品缺乏对安全策略迁移的支持，导致安全边界不能适应虚拟网络的变化。

■ 网络流量不可见

在传统网络中，所有数据包经由交换或路由设备，这些设备可以感知并学习当前环境的数据流量，可以针对目前的网络状况动态调整路由策略。但基于 OpenFlow 的 SDN 架构中的网络控制器只会收到底层设备发来的部分数据包，并不了解控制域中大部分直接被转发的数据流具体内容。

■ 控制器的单点失效

除了传统网络升级到 SDN 后网络层的新问题外，SDN 本身也

会存在漏洞，特别是复杂的 SDN 的控制器。数据平面和控制平面的分离主要是由控制器实现的，所以控制器就成为网络虚拟化的最重要的设施。

然而控制器需要应对各种动态的网络拓扑，解析各种类型的数据包，接收上层应用的信息，并控制底层网络设备的行为，所以功能实现将会非常复杂，也就可能存在不少漏洞。当攻击者攻破控制器，就可以向所有的网络设施发送指令，很容易瘫痪整个网络；或将某些数据流重定向到恶意 VM，造成敏感信息的泄露。

■ 多应用不一致策略导致绕过控制器

控制器控制整个网络的拓扑，处理几百甚至上千个应用的路由策略，每个应用的路由路径可能不同，那么如果这些不同应用产生的路由项之间存在不一致，就可能会出现非法路径。Porras 等人提出图 6 的攻击场景 [2]，系统根据安全策略应禁止主机 10.0.0.2 与主机 10.0.0.4 通信，但如果控制器中有三项看似合法的不同应用需要的路由策略，那么当数据包从 10.0.0.2 传输到 10.0.0.3，会在交换机上被替换掉源和目的地址，成为从 10.0.0.1 传输到 10.0.0.4 的数据包，最终被允许转发，而这原本应该是被禁止的。可见控制器中的路由项如果不一致，攻击者就有可趁之机，可以绕过控制器实施攻击。

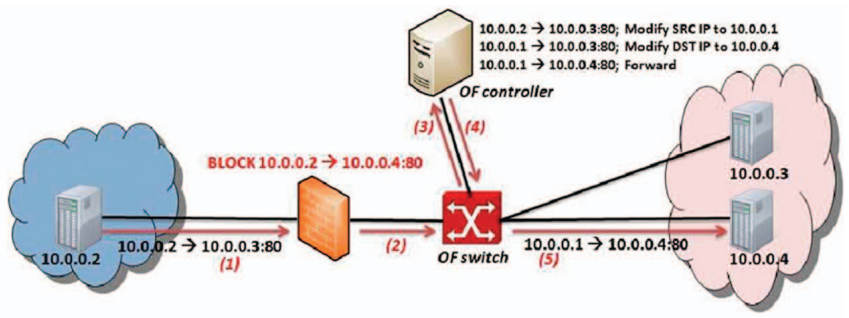

图 6 控制器上三条合法策略组形成一条非法路径

■ 控制信息难验证

除了攻击控制器外，控制器和网络设备间的通信也可能存在安全问题。虽然 OpenFlow 协议规定两者通信可使用加密的通道，但如何保证交互双方可信是一个问题。

一方面，攻击者假冒网络控制器发送恶意控制命令，即可改变网络拓扑，破坏安全策略，或修改数据流绕过防火墙。另一方面，攻击者也可以通过控制某些网络设施，向控制器发送伪造的数据包，影响控制器对网络流量的判断。

如果不解决网络虚拟化后产生的安全威胁，就可能会破坏整个网络的可用性和可靠性，造成租户的隐私泄露，并给攻击者后续攻击内部 VM 提供条件。

3. 网络虚拟化的安全对策

本章主要针对虚拟化环境中出现的安全问题，提出一些可行的安全对策。首先是确保新增的网络设施的安全，然后检测虚拟资源的安全性，最后设计自动高效的安全控制 - 网络控制联动机制。

3.1 保护 SDN 网络的虚拟资源

传统的网络安全认为被攻击目标有应用、服务器和实体网络，SDN 场景还增加了网络控制器和虚拟网络设备，所以网络虚拟化安全的第一步就是要保证这些新资源的安全。

3.1.1 设计安全可靠的网络控制器

网络控制器是 SDN 网络的中心，也是其前置安全保证，所以保障网络虚拟化必须设计一个高可信、安全和健壮的网设计控制器。

首先，控制器需要加入审计机制，检查访问控制器的用户，保证是合法可信的，避免恶意攻击者发动各类攻击，并记录原始日志，做到定时或事后检测异常行为。

其次，保证控制器和交换设备的通信安全，如 OpenFlow 协议就要求两者通信必须存在一个加密通道，需防止中间人攻击。

最后，针对内部攻击或管理员不正确的配置，安全产品可实时或定时检测控制器的路由规则是否兼容并满足安全需求。如针对前面提到的“控制器不一致策略”问题，Porras 等针对控制器 Nox 设计了一个第三方插件 FortNox，可实时检测规则是否冲突。

3.1.2 保证虚拟网络设备安全

在大二层交换网络中，虚拟网络设备主要是指支持 OpenFlow 的虚拟交换机。如果交换机出现异常，会造成网络拓扑变化，甚至会影响控制器的正常工作。

为保护虚拟交换机，需要及时更新 Hypervisor 软件，防止攻击者逃逸虚拟机后利用被攻破的虚拟交换机发送恶意或异常的消息。

此外，应设计网络设备配置的一致性检查，避免发生网络风暴，减少控制器端收到的非必要 PacketIn 数据包请求。

3.1.3 保证控制器和网络设备的通信安全

当前 OpenFlow 协议中规定控制器和交换机之间的通信使用 TCP 或 TCP/TLS 协议，如果使用不加密的 TCP 方式，攻击者很容易伪造交换机的消息，扰乱控制器所获知的网络拓扑。但如果使用认证的加密方式，在大规模网络中控制器容易遭受拒绝服务攻击。所以设计一个轻量级可认证的通信方式，保证控制器收到的消息的秘密性、完整性和可用性，将是一个重要的研究课题。

3.2 推出支持虚拟化的安全产品

针对传统安全产品对内部网络不可见的缺点，安全厂商需推出支持虚拟化的安全产品，这些安全产品以软件的形式存在，并兼容主流的虚拟化解决方案，可监控内部虚拟网络中的数据流。一般而言，支持虚拟化的安全产品通常有虚拟机形态和 Agent 形态：前者可以不对网络拓扑和计算节点进行任何改动，非常方便，但缺点是安全产品容易遭到被感染的虚拟机的攻击，配置也比较复杂；后者可感知虚拟机和网络变化并应用策略，而且产品部署在 VM 不可见的 Hypervisor 层面，在很大程度上减少了对安全产品的攻击，但需要修改物理主机的系统。可预见虚拟机形态和 Agent 形态会相互配合，与安全控制器协同，完成较完备的安全功能。

3.3 设计软件定义的安全解决方案

要想达到真正的软件定义安全（SDSec, Software-Defined Security），就需要在保护现有和新增设备以及内部虚拟网络的基础上，深刻理解 SDN 的工作模式，提出松耦合但与之匹配的安全架构，设计网络控制器和安全控制器联动的安全机制，建立基于环境的数

据传输决策模型。

SDN 网络与传统网络的最大不同是可编程化（programmable），整个网络的数据流和拓扑都在控制器的指令下快速变化，那么安全产品必须理解这种变化，并能程序化地快速自动调整底层设备策略。对于虚拟化场景下的主体，Vmware 公司首席安全和网络架构师 Rob Randell 认为可分成三级 [3]：虚拟数据中心 vDC、虚拟应用 vApp 和虚拟机 VM（如图 7 所示），然后将各类安全应用根据功能组成策略模板，最后多个策略模板可以组成若干个安全组。安全组间能组成复杂的策略，应对不同级别的主体。那么对于一个虚拟化应用而言，不管其 VM 分布在何处，也不管其数据流的路径是什么，总能建立安全组的结构，使之能处理该应用的工作流（workload），实现安全管理的功能。

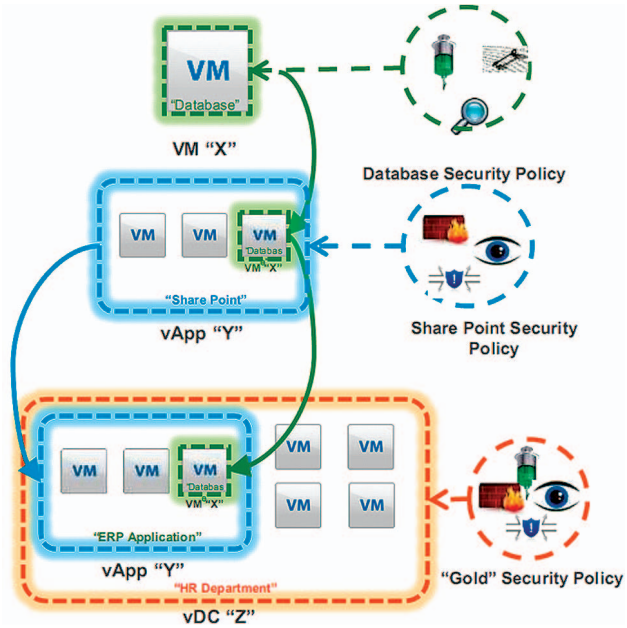

图 7 虚拟化场景下安全策略的分解

4. 结论

SDN 和 OpenFlow 将网络的逻辑视图与物理视图分离，使得网络配置更加灵活，硬件设备成本大大下降，是计算机网络的又一次革命。SDN 带来了巨大便利的同时，也带来了大量尚未解决的安全问题。本文在分析了 SDN 的安全挑战后，给出了一些安全对策。

虽然网络虚拟化为数据中心和企业的网络安全带来了巨大的挑战，但也给网络安全防护提供了新思路和新方法，如 SDN 集中式控制提高产品防护能力、为快速重定向可疑的 DDOS 流量、快速自动地构建攻防实验网络，并降低安全产品的成本。所以研究虚拟环境下的安全问题，研发支持 SDN 的安全产品对保护整个虚拟化网络是必要且有益的。

【1】F5 Networks：SDN 与 ADN 竞争还是合作?

【2】Phillip Porras, Seungwon Shin, Vinod Yegneswaran, Martin Fong, Mabry Tyson, Guofei Gu. A Security Enforcement Kernel for Openflow Networks. Proceedings of the ACM Sigcomm Workshop on Hot Topics in Software-Defined Networking (HotSDN), August 2012.

【3】Rob Randell, How the software defined datacenter is turning security on its head, RSA conference, 2013

新一代漏洞管理产品应该具备的特性

产品推广部　张旭　产品管理中心　尹航

关键字：漏洞管理产品　扫描能力　部署能力　展现能力　管理能力

摘要：漏洞管理产品已经广泛应用于信息安全领域，产品发展改进不大。但随着网络安全态势的变化，漏洞管理产品也面临着一些新的问题，需要进一步的改进与发展。本文分析了当前漏洞管理产品面临的问题，结合绿盟科技自身在漏洞管理领域的研究成果，提出了新一代漏洞管理产品应该具备的新的特性，以帮助读者了解漏洞管理产品的发展动向。

一、前言

漏洞扫描是指基于漏洞数据库，通过扫描、探测、模拟攻击等手段对指定的远程或者本地计算机系统的安全脆弱性进行检测，发现可利用的漏洞的一种安全检测行为。

漏洞扫描技术是一类重要的网络安全检测技术。它能够有效发现系统存在的安全漏洞，提高网络的安全性。通过对目标系统的扫描，系统管理员能及时发现目标系统的安全漏洞，客观评估当前网络安全情况。同时，系统管理员能根据扫描的结果来修补系统安全漏洞，在黑客攻击前进行防范，有效避免黑客攻击行为，做到防患于未然。

目前，漏洞扫描的概念和技术已经相对成熟，已经被安全行业普遍接受。各种类型的漏洞扫描系统也已经被广泛使用。目前使用最为广泛的漏洞扫描系统是网络漏洞扫描系统，网络漏洞扫描产品通过向目标系统发送具有一定特征的网络数据，进而分析目标系统的响应特征来判断目标系统是否存在特定的漏洞。由于网络漏洞扫描技术基于网络通信协议，因此一台网络漏洞扫描系统可以同时对多台目标设备进行漏洞扫描。相较于逐台设备进行人工检查的传统安全检查方法，使用网络漏洞扫描系统可以大幅度降低人工成本和时间成本。同时，使用网络漏洞扫描系统还可以降低对安全检查人员的能力素质要求，减少安全检查难度。由于具有如此大的优

势，因此网络漏洞扫描系统被广泛应用于政府、金融、能源、运营商等行业。

二、漏洞管理产品遇到的问题

漏洞管理产品经过多年的应用，似乎已经发展到了极致，近些年已经少有厂商对产品进行改进。但是，在实际安全管理工作中，安全管理者越来越多地遇到各种问题，究其原因，网络环境在变化，漏洞管理产品也需要不断发展。

首先，传统的漏洞概念只是安全脆弱性的一个方面，漏洞管理产品仅能发现系统漏洞的存在，而广义的漏洞概念还应该包含 IT 系统的安全配置以及建立在继承系统之上的应用层面漏洞，这些方面组成完整系统脆弱性。许多安全管理者已经开始重视全面的系统脆弱性管理，面对这样的需求，传统的漏洞管理产品已经无法满足。

其次，IT 系统的结构也悄然地发生变化，最大的变化体现在网络规模的扩张，这也随之带来管理方面的更多要求。跨区域的大型企业或部门更加希望对庞大的信息系统安全做统一管理，对安全风险全局把握，统一防御，漏洞管理产品在这种要求下需要进行变革。

网络规模化对漏洞管理产品的另一个挑战是，平铺模式的漏洞报告面对大量的漏洞数据已经变得鸡肋，如何为安全管理者提供更直观的风险感受，如何能够直接指出风险集中的重点区域，以及提供有效的修补建议，是漏洞管理产品需要改变的地方。

另外，信息安全是一个时间上相对的安全，从漏洞发现到修补是保证安全的关键时期，而独立的漏洞管理产品对于缩短这个时间显得无能为力。目前主流的处理方式是，安全产商在漏洞披露后，快速响应，发布防护类产品的新规则，安全管理者接到安全厂商的通报后进行规则升级。但这个过程缺乏自动化措施，难免造成拖延和疏忽，需要安全厂商进一步思考和解决。

除了上述问题以外，下一代互联网 IPv6 网络的应用和虚拟化技术的应用，都在改变现有的网络环境，漏洞管理产品要能适应这些环境变化，才能及时为管理者提供有效的防护手段。

三、国际厂商对此进行的探索

对于上文所指出的问题，一些国际优秀厂商已经在不断地探索，并且在产品中进行了实践。

部分国际安全厂商的漏洞管理产品实现了集系统漏洞扫描、配置检查、Web 应用扫描于一身，能够对 IT 系统的脆弱性进行全面检查，适合于日常安全运维管理工作。但是对脆弱性的整体分析和报告，需要依赖于其集中管理平台的组建，对于监督检查和一些小规模网络并不适合。

对于大规模网络的统一管理，一些国际厂商已经有各自漏洞管理产品的集中管理平台。这些集中管理平台能够较好地支持多台分布式部署漏洞管理产品的集中管理，统一扫描任务分配和集中分析报告。不过这种方案对于一些细节的网络拓扑，比如复杂的子网环境，部署起来成本相对较高，还需要一种轻量级低成本的部署方案。

由于欧美国家对信息系统有明确的合规要求，国际厂商的漏洞管理产品，其报告更多的倾向于满足合规要求，而由使用集中管理

平台来完成信息系统整体趋势的分析和报告，比较好地解决了规模化网络下大量漏洞数据的带来的问题。

此外还有一些安全厂商在其漏洞管理产品中实现了与 IPS 的整合，实现了自动化的漏洞的发现和防护，是缩短漏洞响应周期很好的思路，值得借鉴。

四、绿盟科技对新一代漏洞管理产品的观点

绿盟科技作为国内漏洞管理产品的领导者，提出了新一代的漏洞管理产品应该具备的新的特性。同时，依靠自身在漏洞挖掘与安全加固领域的丰富经验，绿盟科技也将这些新的特性融入到自身漏洞管理产品 RSAS v6 之中。

1. 全方位的扫描能力

传统的漏洞管理产品的功能就像其名称一样，将其功能仅局限在系统漏洞的扫描发现上，但是只对系统漏洞进行扫描发现是不能全面发现系统安全隐患的。除了系统层面的漏洞以外，应用层的网络应用，例如 Web 网页等包含的系统漏洞也时刻威胁着网络安全。因此对系统漏洞与应用层漏洞的全面扫描是系统漏洞扫描不可或缺的功能。

能够做到对系统漏洞和应用层漏洞进行扫描发现的漏洞管理产品，我们也认为是不全面的。因为除了漏洞以外，系统错误的配置也是威胁系统安全的重要安全隐患。一个系统没有安全漏洞、打全安全补丁，但是错误地设置了短小简单的密码或错误地设置了访客权限等等都会对系统安全造成威胁。而这种威胁的可怕之处在于，传统漏洞扫描系统对此是不会发出报警的，反而会告诉系统管理员“系统非常安全”。

因此绿盟科技认为，新一代的漏洞管理产品应该具有系统漏洞、配置隐患、应用层漏洞全方位的扫描能力，才能协助安全管理者全面掌握系统安全状态。绿盟科技 RSAS v6 在原有强大漏洞扫描能力的基础上集成了安全配置核查功能，多管齐下，能够全方位地发现系统安全隐患。

2. 灵活的部署能力

随着 IT 技术的不断发展，信息系统的规模也在不断的变大，从最开始几个人用一台电脑发展到现在平均一个人几台电脑。规模扩张的另一个变化就是系统的复杂性，大规模网络多数是分批逐步建立起来的，很难做到统一规划，出于安全考虑，不同安全性质的子网之间的访问要受到限制。漏洞管理产品对目标进行扫描，要采用就近原则，面对众多的子网，漏洞扫描产品的性能不能充分利用，安全建设的成本将成为困扰，这要求漏洞管理产品能够提供灵活的部署能力。

绿盟科技认为，优秀的漏洞管理产品不但要具备大规模网络的分布式部署集中管理能力，还应具备小规模子网的轻量级部署的灵活部署能力。绿盟科技 RSAS v6 可以实现单机单链路扫描、单机多链路扫描、代理引擎扫描、大规模分布式部署扫描等多种扫描方式，灵活的部署能力带来更加灵活的适应能力。

3. 一目了然的风险态势展现能力

网络规模化后，面对大量的漏洞数据，平铺模式的漏洞报告

已经变得越来越难以阅读。海量的计算机信息、漏洞信息、修补信息等扑面而来，使得安全管理者陷入“只见树木不见森林”的境地，无法全面掌握系统当前的安全态势，无法安排下一阶段的安全修补工作。因此，大量数据的分析与管理能力也是评价漏洞管理产品的一个重要指标。

绿盟科技 RSAS 集成资产风险管理功能，可以对计算机、网络设备等资产实现集中管理，同时支持以资产和资产组为单位全面展示当前的风险状态。除了优秀的资产风险管理功能外，RSAS v6 还设置了系统仪表盘和安全告警平台，可以让用户一目了然地掌握当前的系统风险状态。

4. 从发现到防护的闭环管理能力

信息安全不是一成不变的，随着新漏洞的披露，信息安全态势会变的恶化；随着系统补丁的修补，安全产品策略完善，信息安全态势又会逐渐好转。而从系统漏洞披露到安全策略完善的这段时间里，信息系统可以用“弱不禁风”来形容，必须尽量缩短这段“危险期”的长度。系统安全不是靠某一个产品单打独斗能够保障的，因此，多种安全产品应该形成合力，共同保障信息系统的安全。

绿盟科技 RSAS v6 未来会与 NGFW、NGIPS 等网关类安全设备实现整合，RSAS 发现系统安全漏洞后立刻通知网关类安全设备调整防护策略，实现从发现到防护的快速闭环管理过程。

五、总结

作为一个以“专攻术业、成就所托”作为企业愿景的信息安全厂商，绿盟科技本身在不断地进行着实践与进步，完善自己的漏洞管理产品。除了上述功能与特性外，新产品 RSAS v6 中还加入了对 IPv6、虚拟化等技术的应用，已经开始具备新一代漏洞管理产品的雏形。相信在不久的将来，绿盟科技新一代漏洞管理产品将会成为安全管理者手中的利器。

JBoss安全概述

威胁响应中心　申军利

关键字：JBoss　攻击　指纹识别

摘要：JBoss Application Server（JBoss AS）是一款被广泛使用的基于 Java 的开源的服务平台软件，自带很多功能强大的管理应用。但是，这些应用在方便用户管理的同时，也带了很多安全上的问题。

引言

JBoss Application Server（JBoss AS）是一款被广泛使用的基于 Java 的开源的服务平台软件，它是 JBoss Enterprise Middleware Suite (JEMS) 的一个部分。JBoss AS 可以用于二次开发，例如 Java 平台应用程序的开发、企业级应用程序的开发、Web 应用程序的开发以及门户网站的开发，而且JBoss AS能够在各种不同环境下搭建。因此，它被许多组织所使用，用于开发各种传统的企业 Web 网站、商业软件、电话系统控制应用程序。

但是在使用过程中，由于使用者对 JBoss 自身携带的应用功能不甚了解，导致出现诸多安全问题。本文将归纳 JBoss 过往所出现的问题，来阐述使用者的一些错误行为，以及攻击者针对 JBoss 进行攻击的一些常用手法。

一、JBoss 简介

JBoss AS 支持商务环境中所需要的多种重要平台特性，从集成有效缓存到类似 Java Message Service 的各种 JEE 能力的实施。图 1 覆盖了 JBoss AS 中当前所使用的大部分技术。

虽然 JBoss AS 是一个平台级的软件，但是在这里我们仍将它认定为框架。这是因为 JBoss AS 在提供平台服务的同时，也提供了大量的管理应用与 API。因此，我们将 JBoss AS 定义为附带框架功能的 Java Web 服务平台。

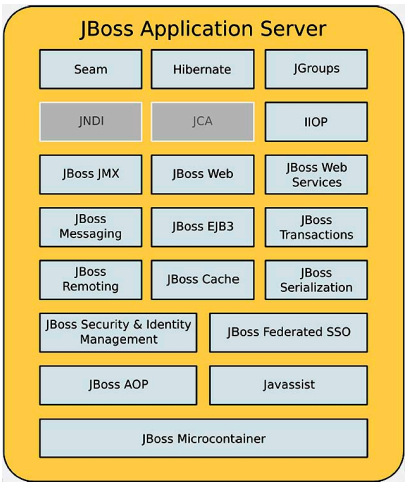

图 1 JBoss AS 架构图

这些方便开发者和使用者方便管理调用的应用（或者说是框架功能），也恰恰成为了攻击者下手的目标。强大的应用，如果没有

做到足够的权限限制，必会成为高危的漏洞。JBoss 的漏洞史刚好可以证明这句话，到目前为止每一个 JBoss 的高危漏洞，都是由于应用功能权限没有进行限制，或者默认配置没有进行修改，从而被攻击者所利用（例如 CVE-2007-1036）。

这里引用 5up3rh3i 的《我的安全世界》中的一句话——权限突破是安全漏洞评定一个核心指标。JBoss AS 的权限默认配置，导致攻击者对于其权限的突破非常容易，从而导致一个又一个的 JBoss 站点被攻陷。

二、JBoss 以往问题归纳

通过对 JBoss 以往问题的了解，我们将这些问题分成了两类，分别是配置错误和认证绕过。因为，不论攻击者使用 WAR 包发布的方法，还是使用 RMI 调用 JBoss 内置功能的方法攻击 JBoss AS，都是在获取权限之后，即对于正常功能的非法使用。最根本的问题则是，没有对这些强大的功能做足够的限制，从而导致被未授权用户所使用。下面我们来看攻击者是如何利用这两类问题对 JBoss AS 进行攻击的。

（一）配置错误

这类问题又可以根据 JBoss 远程管理的两种形式，分为直接访问管理和远程访问对外管理两类。而这两类又不会相互影响，即直接访问管理功能被限制，不会影响远程访问对外管理。同理，远程访问对外管理被限制，不会影响直接访问管理。图 2 中橘黄色模块，为 JBoss AS 提供的对外管理接口。其中 Mgmt.APP. 为远程访问对

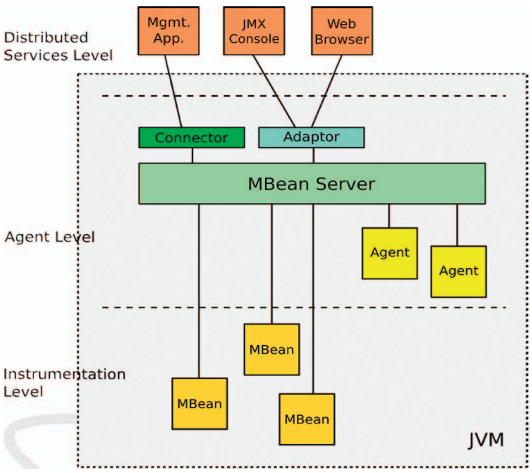

图 2 JMX 架构图

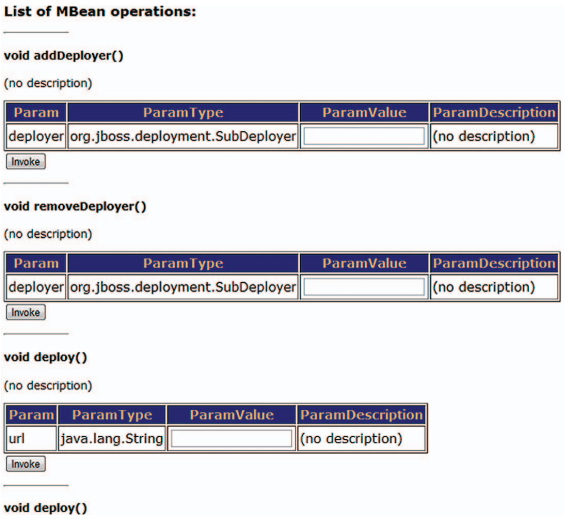

图 3 JMX-Console 管理界面

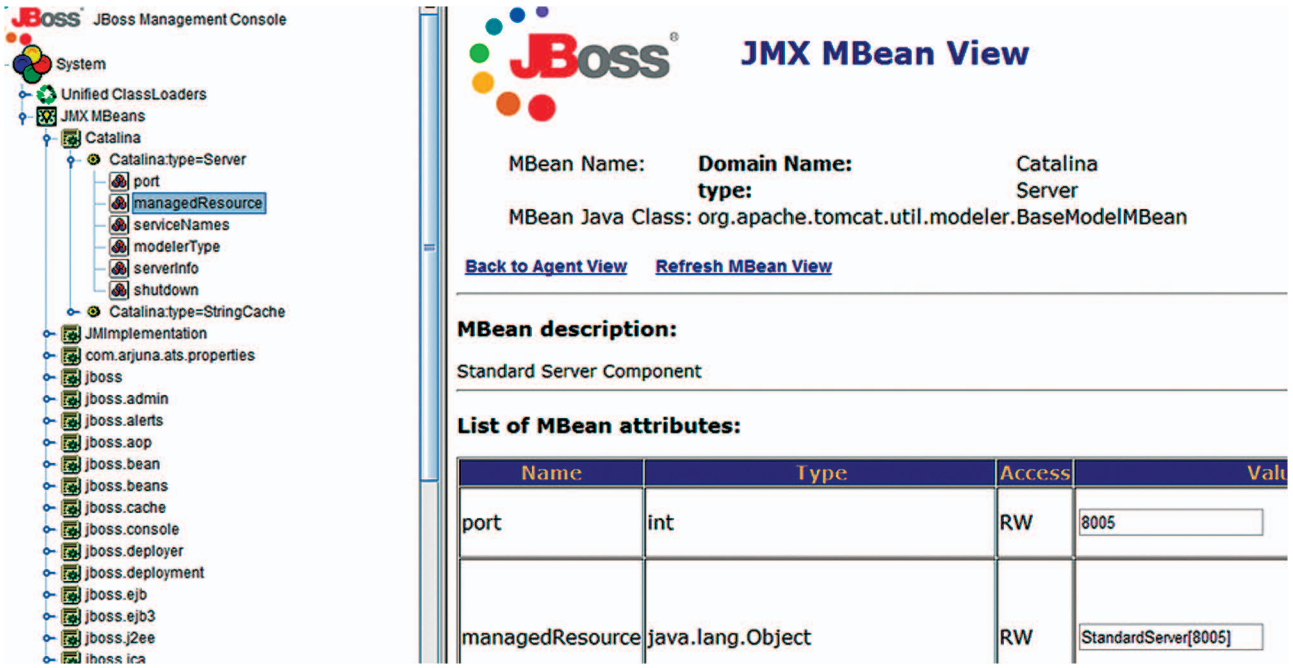

图 4 Web-Console 管理界面

外管理接口，JMX-Console 和 Web-Browser 为直接访问管理接口。

直接访问管理，通过 JBoss 提供的应用程序，可以直接在浏览器上对 JBoss 进行管理。若没有进行任何权限配置（默认情况下），将导致攻击者可以直接使用服务功能，发布 WAR 文件。图 3、图 4 分别为 JMX-Console 和 Web-Console 管理界面的部分截图。

远程访问对外管理，通过 JBoss 提供的 API 接口编写客户端，或者使用 JBoss 提供的客户端（twiddle）对远程服务进行管理。这个管理功能是独立于 HTTP 服务的 8080 端口的，默认是 1098、1099 和 4444。图 5 为通过 JBoss 在 Windows 客户端 twiddle.dat，获取远程服务的系统信息。

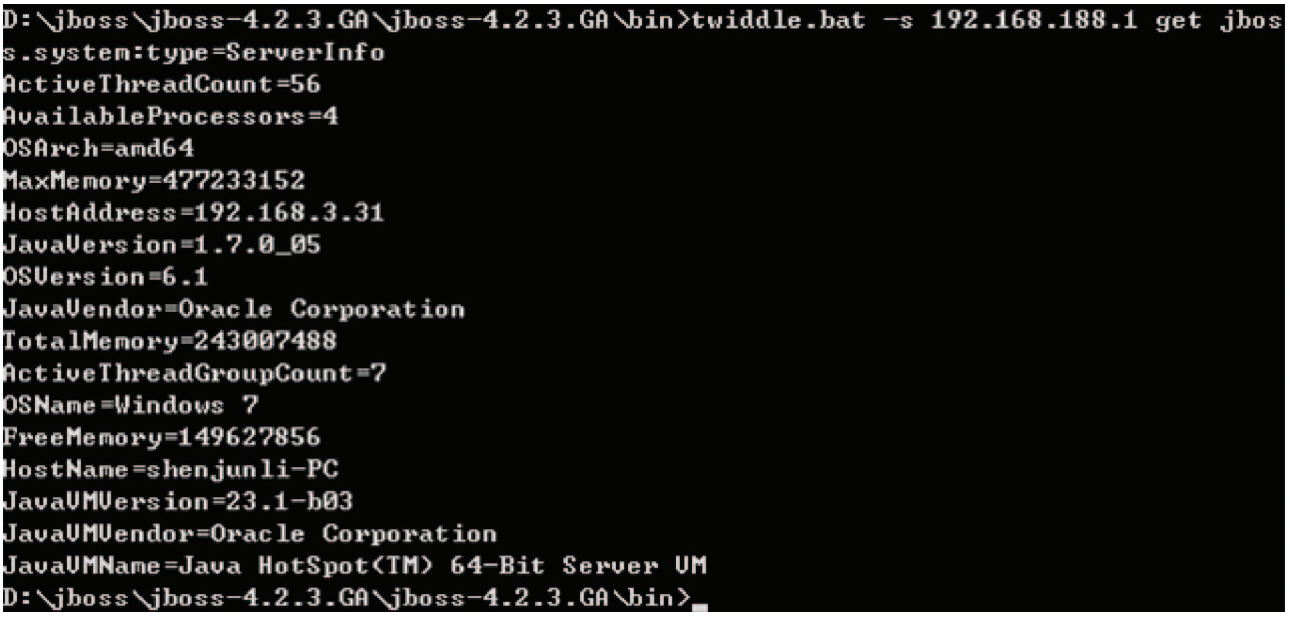

图 5 twiddle 管理工具

（二）认证绕过

这种问题也可以分为两种类型，一种是通过暴力破解认证，另一种是利用漏洞绕过认证。

暴力破解，JBoss 在对用户认证进行校验时，没有采取防止暴力猜测机制，导致攻击者可以对认证用户的用户名和密码进行暴力猜解。获取认证用户权限后，便可调用 JBoss 所提供的应用进行恶意利用。

利用漏洞，JBoss 在某些强大的应用中，“忘记”了对用户权限的检验，默认将所有可以访问到它的用户都当做了受信任用户，从而导致这些应用被滥用。JBoss 漏洞中出现过两个这样的典型漏洞，分别是 CVE-2010-0738 和 CVE-2007-1036。

CVE-2010-0738 是由于认证校验中没有对 POST 和 HEAD 方法进行检验，导致攻击者可以通过这两种方法上传并发布 WAR。

CVE-2007-1036 这个漏洞的原因是由于应用程序认为用户不可能使用这个 Severlet 接口，而未做任何认证措施，以至于这个服务完全对外开放。攻击者可以模仿 JBoss 调用这个接口的数据，构造恶意数据提交，造成代码执行攻击。网上也有很多人

对这个问题做了较为详细的描述，这里就不赘述了。

三、JBoss 攻击流程

一个叫 luca 的研究人员在 OWASP 一次峰会上的名为《From CVE-2010-0738 to the recent JBoss worm》议题中，用了一张流程图将攻击者针对JBoss攻击的流程完美地描绘了出来，如图6所示。

四、JBoss 指纹识别

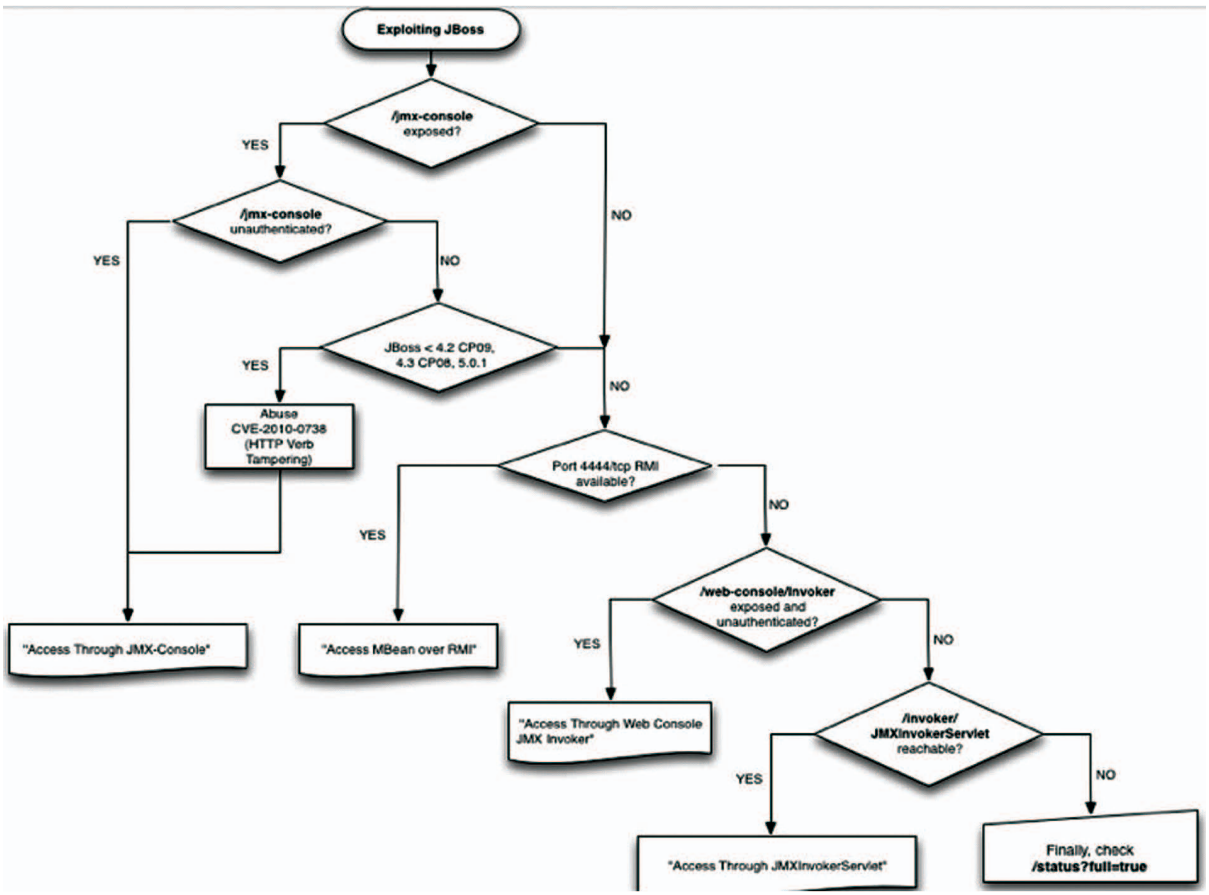

图 6 JBoss 攻击流程图

通过前三章对 JBoss AS 安全性的归纳和总结，我们从宏观的角度了解了它所存在问题的本质。本章将向各位提供我们在分析这些问题的时候，所总结的一些识别 JBoss 服务的方法。这些方法能够使你更容易地判断目标服务器是否使用的是JBoss AS 平台。不过，这些方法并不能帮助你确定目标是否存在漏洞。

首先我们介绍的是通过 Google Hacking 方法来进行判断，下面是我们所收集的关键字。你也可以根据你所要检测的目标网站，添加 site 语法，或者结合你所收集的一些其他特征。

intitle: ” JBoss Management Console – Server Information”

intitle:” application server”

inurl:” web-console”

inurl:” jmx-console”

inurl:/invoker/JMXInvoker

inurl:/status intext:JBoss ™ Application Server

也许很多网站出于安全性的考虑，删除了上述的 JBoss 特征，没有关系，我们可以通过端口扫描来检查它们的潜在特征。就像我们在第二章中提到过的 1098、1099 和 4444 端口，JBoss 还有很多默认打开的特殊端口。下面是我们收集到 JBoss 默认开放的端口和对应的服务，这些端口中只要有 2-3 个开放，就可以判定服务有 80% 以上的可能使用的是 JBoss。为什么只有 80%？因为像 8080 这样的端口，也有很多的平台也在使用，例如 tomcat、glassfish 等。当然，如果你只看到了4444、1099、1098 这样的端口，可以 99% 甚至 100% 确定它是 JBoss。

JNDI ----------> 1098,1099

RMI ----------> 4444

PooledInvoker ----------> 4445

InvocationUnMarshaller --------> 4446

AjpProtocol ----------> 8009

Http11ProtocolHTTP ----------> 8080

WebService ----------> 8083

UILServerILService ----------> 8093

除了上述的两个方法，我们也可以通过常规的获取旗标的方法，来获得目标的服务平台信息，我们经常使用的工具是 curl。不过，也有很多的管理员修改了他们网站的旗标信息，让我们看不出它是用什么搭建的。

最后，还有一个比较常用的方法，就是让服务报错。可能看到 “报错” 这两个字，你会有些大失所望，不过这确实是一个非常有效的办法，每个平台都会在报错页面中暴露出自己的一些信息。当然，这也不是百试百灵的，有些存在强迫症潜质的管理员会把报错页面重定向到自己指定的页面。

参考文献

1）《Bridging the Gap between the Enterprise and You_or_ Who’s the JBoss now?》

By Patrick Hof, Jens Liebchen

2）《From CVE-2010-0738 to the recent JBoss worm》

By luca

DDoS放大攻击原理及防护方法

安全研究部　洪海

关键字：DDoS 反射攻击　DDoS 放大攻击　DDoS 防护

摘要：DDoS 放大攻击是一种历史悠久而又威力强大的攻击技术。最早的放大拒绝服务攻击可以追溯到古老的 smurf 攻击。现代的 DDoS 放大攻击能够对被攻击目标造成极大影响，甚至拖慢局部互联网的访问速度。本文将对各种 DDoS 放大攻击的原理和 DDoS 放大攻击的防护方法进行简单的介绍。

一、概念

在介绍其具体技术之前，我们先对几个概念进行简要的说明。

（一）针对网络带宽资源的 DDoS 攻击

按照 DDoS 攻击所针对的攻击目标和所属的层次，可以将 DDoS 攻击大体分为三类，即针对网络带宽资源的 DDoS 攻击（网络层）、针对连接资源的 DDoS 攻击（传输层）以及针对计算资源的攻击（应用层）。针对网络带宽的 DDoS 攻击是最古老而常见的一种 DDoS 攻击方式。

无论是服务器的网络接口带宽，还是路由器、交换机等互联网基础设施的数据包处理能力，都是存在着事实上的上限的。当到达或通过的网络数据包数量超过了这个上限时，就会出现网络拥堵、响应缓慢的情况。针对网络带宽资源的 DDoS 攻击就是根据该原理，利用广泛分布的僵尸主机发送大量的网络数据包，占满被攻击目标的全部带宽，从而使正常的请求无法得到及时有效的响应，造成拒绝服务。

（二）DDoS 反射攻击

攻击者可以使用 Ping Flood、UDP Flood 等方式直接对被攻击目标展开针对网络带宽资源的 DDoS 攻击，但这种方式不仅低效，还很容易被查到攻击的源头。虽然攻击者可以使用伪造源 IP 地址的方式进行隐藏，但更好的方式是使用 DDoS 反射攻击技术。

DDoS 反射攻击是指利用路由器、服务器等设施对请求产生应答，从而反射攻击流量并隐藏攻击来源的一种 DDoS 技术。

DDoS 反射攻击的基本原理如图 1 所示。

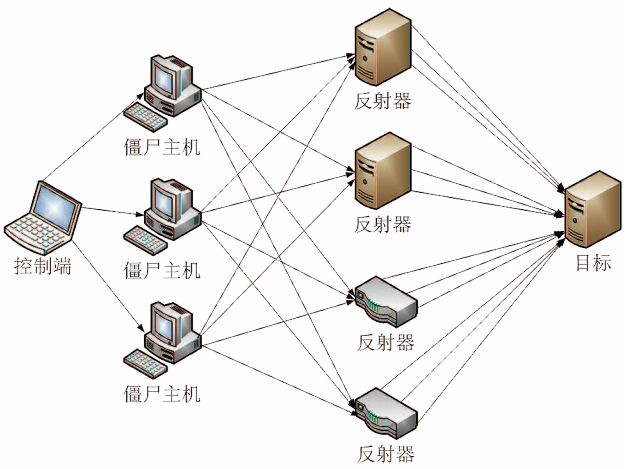

图 1 DDoS 反射攻击原理示意图

在进行 DDoS 反射攻击时，攻击者通过控制端控制大量僵尸主机发送大量的数据包。这些数据包的特别之处在于，其目的 IP 地址指向作为反射器的服务器、路由器等设施，而源 IP 地址则被伪造成被攻击目标的 IP 地址。反射器在收到数据包时，会认为该数据包是由被攻击目标所发来的请求，因此会将响应数据发送给被攻击目标。当大量的响应数据包涌向攻击目标时，就会造成拒绝服务攻击。

发动 DDoS 反射攻击需要在互联网上找到大量的反射器，对于某些种类的反射攻击，这并不难实现。例如，对于 ACK 反射攻击，只需要找到互联网上开放 TCP 端口的服务器即可，而这种服务器在互联网上的存在是非常广泛的。

相比于直接伪造源地址的 DDoS 攻击，DDoS 反射攻击由于增加了一层反射步骤，更加难以追溯攻击来源。

(三)DDoS 放大攻击

DDoS 放大攻击是 DDoS 反射攻击的一种特殊形式。简单的说，当使用的反射器对网络流量具有放大作用时，DDoS 反射攻击就变成了 DDoS 放大攻击。

针对网络带宽资源的 DDoS 攻击、DDoS 反射攻击和 DDoS 放大攻击的关系见图 2。

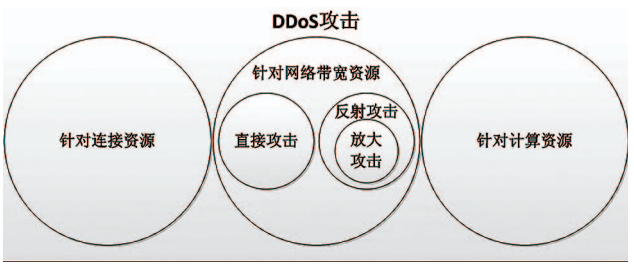

图 2 流量式攻击、反射攻击和放大攻击的关系

本文后面会对 DDoS 放大攻击的技术原理进行详细的介绍。

二、DDoS 放大攻击的技术原理

(一)DDoS 放大攻击的特点

前面已经提到，DDoS 放大攻击是一种特殊的 DDoS 反射攻击，其特殊之处在于反射器对于网络流量具有放大作用，因此我们也可以将这种反射器称为放大器。进行 DDoS 放大攻击的方式与 DDoS 反射攻击的方式也是基本一致的，不同之处在于反射器（放大器）所提供的网络服务需要满足一定条件。

首先，在反射器提供的网络服务协议中，需要存在请求和响应数据量不对称的情况，响应数据量需要大于请求数据量。响应数据量与请求数据量的比值越大，放大器的放大倍数也就越大，进行 DDoS 放大攻击的效果也就越明显。

其次，进行 DDoS 放大攻击通常会使用无需认证或握手的协议。DDoS 放大攻击需要将请求数据的源 IP 地址伪造成被攻击目标的 IP 地址，如果使用的协议需要进行认证或者握手，则该认证或握手过程没有办法完成，也就不能进行下一步的攻击。因此，绝大多数的 DDoS 放大攻击都是用基于 UDP 协议的网络服务进行攻击的。

最后，放大器使用网络服务部署的广泛性决定了该 DDoS 放大攻击的规模和严重程度。如果存在某些网络服务，不需要进行认证并且放大效果非常好，但是在互联网上部署的数量很少，那么利用该网络服务进行放大也不能打出很大的流量，达不到 DDoS 攻击的效果，这种网络服务也就不具备作为 DDoS 放大攻击放大器的价值。

以上三点就是 DDoS 放大攻击中放大器所开放的网络服务具有的特点，之后介绍的 DNS 放大攻击、SNMP 放大攻击都满足这些特点。同时我们也可以说，满足以上三个特点的网络服务协议都能够用于 DDoS 放大攻击。

(二)DNS 放大攻击

DNS 是域名系统（Domain Name System）的缩写，是因特网的一项核心服务。它作为可以将域名和 IP 地址相互映射的一个分布式数据库，能够使人更方便地访问互联网，而不用去记住能够被机器直接读取的 IP 数串。DNS 使用的 TCP 与 UDP 端口号都是 53，主要使用 UDP 协议。

通常，DNS 响应数据包会比查询数据包大，攻击者利用普通的 DNS 查询请求就能够将攻击流量放大 2 到 10 倍。但更有效的方法是使用 RFC 2671 中定义的 DNS 扩展机制 EDNS0。

在没有 EDNS0 以前，对 DNS 查询的响应数据包被限制在 512 字节以内。当需要应答的数据包超过 512 字节时，根据 DNS 服务实现的不同，可能会丢弃超过 512 字节的部分，也可能会使用 TCP 协议建立连接并重新发送。无论是哪种方式，都不利于进行 DNS 放大攻击。

在 EDNS0 中，扩展了 DNS 数据包的结构，增加了 OPT RR 字段。在 OPT RR 字段中，包含了客户端能够处理的最大 UDP 报文大小的信息。服务端在响应 DNS 请求时，解析并记录下客户端能够处理的最大 UDP 报文的大小，并根据该大小生成响应的报文。

攻击者能够利用 dig（Domain Information Groper）和 EDNS0

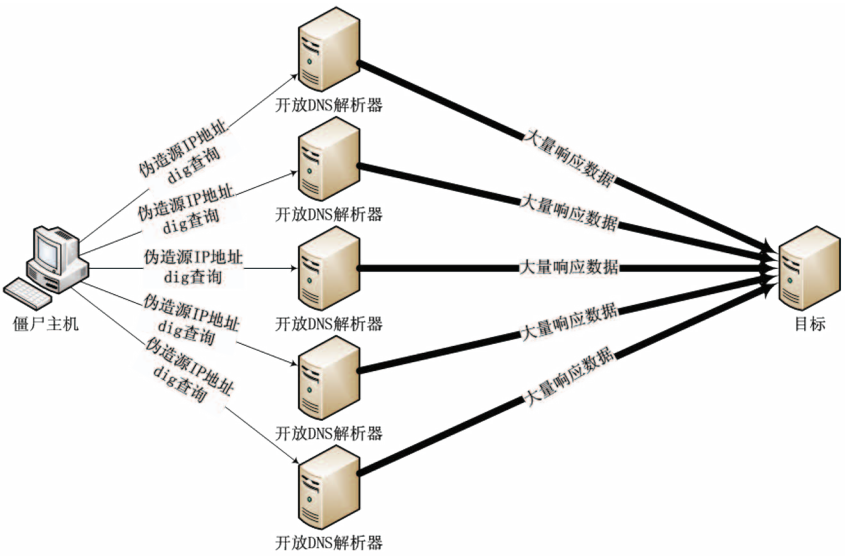

图 3 DNS 反射攻击示意图

进行高效的 DNS 放大攻击。

攻击者向广泛存在的开放 DNS 解析器发送 dig 查询，将 OPT RR 字段中的 UDP 报文大小设置为很大的值（如 4096），并将请求的源 IP 地址伪造成被攻击目标的 IP 地址。DNS 解析器收到查询请求后，会将解析的结果发送给被攻击目标。当大量的解析结果涌向目标时，就会导致目标网络拥堵和缓慢，造成拒绝服务攻击。

攻击者发送的 DNS 查询请求数据包大小一般为 60 字节左右，而查询返回结果的数据包大小通常为 3000 字节以上，因此，使用该方式进行放大攻击能够达到 50 倍以上的放大效果。极端情况下，36 字节的查询请求能够产生 3k-4k 字节的应答，也就是说，能够对攻击流量进行一百倍放大。

在 2013 年 3 月对 Spamhaus 的 DDoS 攻击中，主要就使用了 DNS 放大攻击技术，使得攻击流量达到了史无前例的 300Gbps，甚至拖慢了局部互联网的响应速度。

（三）SNMP 放大攻击

SNMP 是简单网络管理协议（Simple Network Management Protocol）的缩写，该协议是目前 UDP/IP 网络中应用最为广泛的网络管理协议，它提供了一个管理框架来监控和维护互联网设备。SNMP 协议使用 UDP 161 端口进行通信。

由于 SNMP 的效果很好，网络硬件厂商开始把 SNMP 加入到它们制造的每一台设备。今天，各种网络设备上都可以看到默认启用的 SNMP 服务，从交换机到路由器，从防火墙到网络打印机，无一例外。同时，许多厂商安装的 SNMP 都采用了默认的通信字符串（community string），这些通信字符串是程序获取设备信息和修改配置必不可少的。最常见的默认通信字符串是 public 和 private，除此之外还有许多厂商私有的默认通信字符串。几乎所有运行 SNMP 的网络设备上，都可以找到某种形式的默认通

信字符串。

在 SNMPv1 中定义的 Get 请求可以尝试一次获取多个 MIB 对象，但响应消息的大小受到设备处理能力的限制。如果设备不能返回全部请求的响应，则会返回一条错误信息。在 SNMPv2 中，添加了 GetBulk 请求，该请求会通知设备返回尽可能多的数据，这使得管理程序能够通过发送一次请求就获得大段的检索信息。

利用默认通信字符串和 GetBulk 请求，攻击者能够开展有效的 SNMP 放大攻击。

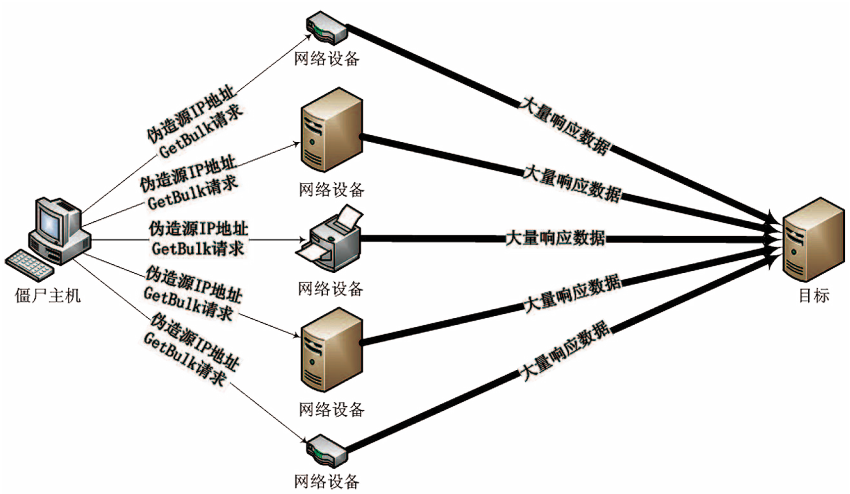

图 4 SNMP 反射攻击示意图

攻击者向广泛存在并开启了 SNMP 服务的网络设备发送 GetBulk 请求，使用默认通信字符串作为认证凭据，并将源 IP 地址伪造成被攻击目标的 IP 地址。设备收到 GetBulk 请求后，会将响应结果发送给被攻击目标。当大量的响应结果涌向目标时，就会导致目标网络拥堵和缓慢，造成拒绝服务攻击。

攻击者发送的 GetBulk 请求数据包约为 60 字节左右，而请求的响应数据能够达到 1500 字节以上，因此，使用该方式进行放大攻击能够达到 20 倍以上的放大效果。

（四）其他放大攻击

前面详细介绍了 DNS 放大攻击和 SNMP 放大攻击的原理，除了这两种协议以外，还有一些协议和网络服务可以用于 DDoS 放大攻击。

在 NTP 协议中，monlist 请求可以获取与目标 NTP 服务器进行过同步的最后 600 个客户端的 IP 地址。发送一个很小的请求包，就能获取到大量的活动 IP 地址组成的连续 UDP 包。通过伪造 IP 地址并发送 monlist 请求，可以将攻击流量放大 500 倍以上。

在 CHARGEN 协议中，每当服务器收到客户端的一个 UDP 数据包，这个数据包中的内容将被丢弃，而服务器将发送一个数据包到客户端，其中包含长度为 0 ~ 512 字节之间随机值的任意字符。利用该协议可以将攻击流量放大 2 ~ 10 倍。

需要说明的是，由于这些协议在互联网上部署的范围不够广泛，因此它们不能作为 DDoS 放大攻击的主要手段和产生攻击流量的主要部分，只能作为辅助手段增大攻击流量。

三、DDoS 放大攻击的防护方法

对于 DDoS 放大攻击，可以从三个方面进行防护。

(一) 对开放 DNS、SNMP 服务的设备进行防护

首先，应确认设备上的 DNS、SNMP 服务是否是必要的，如非必要，则应该关闭这些服务。

如果必须开放 DNS、SNMP 服务，则应加强对这些服务请求的鉴权和认证。例如，DNS 服务不应对互联网上的任意计算机都提供域名解析服务，而只应该响应该 ISP 或该网络内部的 DNS 解析请求；SNMP 要使用非默认的独特的通信字符串，并尽可能升级到 SNMPv3 版本，以提高安全性。

最后，还应该限制 DNS、SNMP 响应数据包大小的阈值，直接丢弃超大的响应数据包。

只要对开放 DNS、SNMP 服务的设备等放大器进行有效防护，就能从根源上杜绝 DDoS 放大攻击的产生。

(二) ISP 对伪造源 IP 地址的数据包进行过滤

攻击者能够发送伪造源 IP 地址的数据包，这是针对网络带宽资源的 DDoS 攻击能够产生的根本原因。通过伪造源 IP 地址，不仅能够发动 DDoS 反射攻击和 DDoS 放大攻击，还能够有效地隐藏攻击来源，降低攻击者面临的风险。如果 ISP 能够对伪造源 IP 地址的数据包进行过滤，使其不能进入到互联网中，就能够从根本上解决针对网络带宽资源的 DDoS 攻击问题。

在 RFC 2827/BCP 38 中高度建议使用入口过滤，以阻止伪造源 IP 地址的网络攻击。遗憾的是，只有少数的公司和 ISP 遵守了这些建议，但只有当所有接入互联网的设备都遵守该规则时，才能彻底阻止伪造源 IP 地址的网络攻击。

(三) 使用 Anycast 技术对攻击流量进行稀释和清洗

利用 DDoS 放大攻击技术，能够打出很大的流量。对这种规模的攻击，需要对攻击流量在多个清洗中心进行分布式清洗，将攻击流量扩散和稀释，之后在每个清洗中心进行精细的清洗。

使用 Anycast 技术进行防护是一种可行的方案。通过使用 Anycast 技术，可以有效地将攻击流量分散到不同地点的清洗中心进行清洗。在正常环境下，这种方式能够保证用户的请求数据被路由到最近的清洗中心；当发生 DDoS 攻击时，这种方式能够将攻击流量有效地稀释到防护方的网络设施中。此外，每一个清洗中心都声明了相同的 IP 地址，攻击流量不会向单一位置聚集，攻击情况从多对一转变为多对多，网络中就不会出现单点瓶颈。在攻击流量被稀释之后，清洗中心对流量进行常规的清洗和阻断就变得相对容易了。

四、总结

DDoS 放大攻击是一种针对网络带宽资源的分布式拒绝服务攻击形式，通常利用网络协议请求与响应数据的不对称、网络服务无需认证以及网络服务部署的广泛性来达到放大攻击流量的效果。

常见的 DDoS 放大攻击技术包括 DNS 放大攻击和 SNMP 放大攻击等，其他一些协议和网络服务也能够作为辅助手段增加 DDoS 放大攻击的流量。

对 DDoS 放大攻击的防护，需要从攻击源头、放大器、被攻击目标三个方面进行防护，才能达到最有效的防护效果。

基于SDN的安全解决方案

行业技术部　王卫东

关键字：SDN　DDoS　防火墙

摘要：本文分析了基于 SDN 的 DDoS 解决方案的实现原理以及基于 SDN 架构的防火墙工作机制和商业实例。

1. 前言

SDN（软件定义网络）从诞生之初就受到了全球瞩目。2012 年，Gartner 在预测未来五年内 IT 领域十大关键趋势和技术影响时，更是将 SDN 排在第二位。根据 2013 年 2 月 GigaOM 的调查报告，预计到 2018 年全球 SDN 市场份额将占到 24.5 亿美元。近年来，SDN 技术在市场上表现出势不可挡的发展态势，主要表现在：

1）**具有示范意义的商用案例不断增加**：

- Google 的 B4 网络实现了数据中心的互联。
- NTT 通过采用 SDN 技术率先现了全球多个数据中心的整合虚拟化。
- Internet2（覆盖百座美国高校）宣布基于 OpenFlow 建成首个开放 SDN 网络。
- 央视 - 索福瑞的数据中心全面采用基于 OpenFlow 的 SDN 网络。

2）**IT 巨头纷纷推出 SDN 产品**：

- HP 发布了 Virtual Application Networks SDN 控制器和 25 款支持 OpenFlow 的交换机产品。
- IBM 发布了自有 OpenFlow 控制器 PNC。
- 思科推出了开放网络环境（Open Network Environment）的解决方案。
- Juniper 推出了基于 XMPP 协议而不是 OpenFlow 的 JunosV Contrail 控制器。

3）**SDN 领域里的融资购并非常活跃**

- Oracle 于 2012 年 7 月 31 日宣布收购 Xsigo 公司。
- 英特尔和高盛对一家 SDN 公司 Big Switch Networks 进行了投资。
- 2012 年 8 月，SDN 创业公司 Plumgrid 融资 1070 万美元。
- 2012 年 12 月思科收购 SDN 初创公司 BroadHop 和 Cariden。
- VMware2012 年 7 月以 10.5 亿美元的价格收购了网络虚拟化初创企业 Nicira。

■ 2012 年 12 月 Juniper 用 1.76 亿美元现金和股票收购 SDN 创业企业 Contrail Systems。

■ Brocade 以现金的方式收购 SDN 先锋 Vyatta。

SDN 发展壮大带来网络产业格局重大调整的同时，也势必会波及到网络安全设备行业。网络侧安全产品在本质上是一种特殊用途的网络设备，SDN 技术将对跨 L2-L7 的整个协议栈产生影响，因此在网络基础架构发生变化时，甚至是发生变化之前，网络安全设备的工作机制和解决方案也会发生相应的变化。SDN 技术可以为网络安全带来一些有益的帮助，例如简化跨越地理边界的 VLAN 扩展，再例如边界模糊化的环境下安全设备（如防火墙）的部署位置难以确定的问题，SDN 技术可以将所有流量路由到一个中央防火墙以解决上述难题。事实上在目前 SDN 网络尚未大规模普及的情况下，已经出现了基于 SDN 的安全解决方案。本文主要介绍基于 SDN 的抗 DDoS 攻击的解决方案和基于 SDN 的防火墙的实现。

2. 基于 SDN 的抗 DDoS 攻击解决方案

2.1 攻击的检测

■ 攻击检测的目标

抗 DDoS 攻击解决方案通常由攻击检测和流量清洗两部分组成。攻击检测是通过对流量日志的分析计算实现的。攻击检测的目标有两个层次，一个是发现被攻击的目标地址，而更高层次的要求是将属于攻击流量的日志记录标记出来，从而抽取攻击的流量特征。

传统的抗 DDoS 攻击解决方案只要求检测出被攻击目标即可，后续对被攻目标流量的分析、找出攻击流量的工作是由流量清洗设备完成的。而在 SDN 环境中，需要知道攻击流量的特征并借此生成流表项，因此需要攻击检测方案能够发现攻击流量的特征。

■ 流量日志的选择

主流的流量日志的生成方法有两类，即 NetFlow/IPFIX 和 sFlow。NetFlow 是思科发

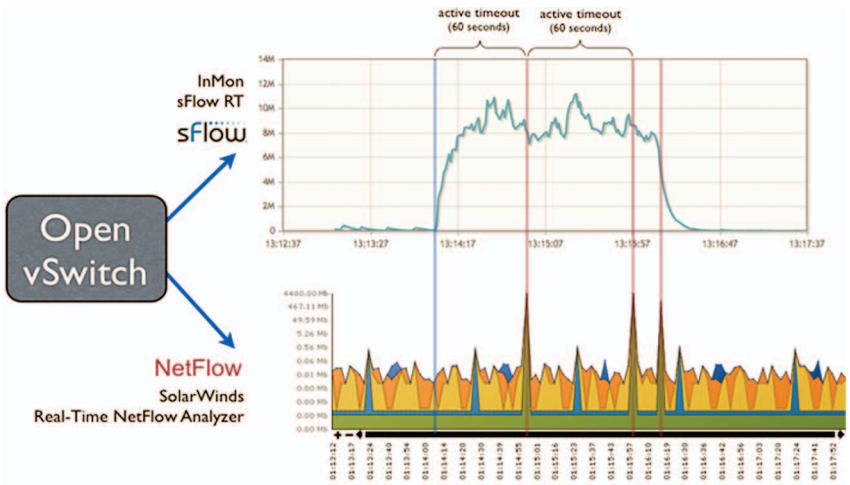

图 1 sFlow 与 NetFlow 的攻击检测延迟对比

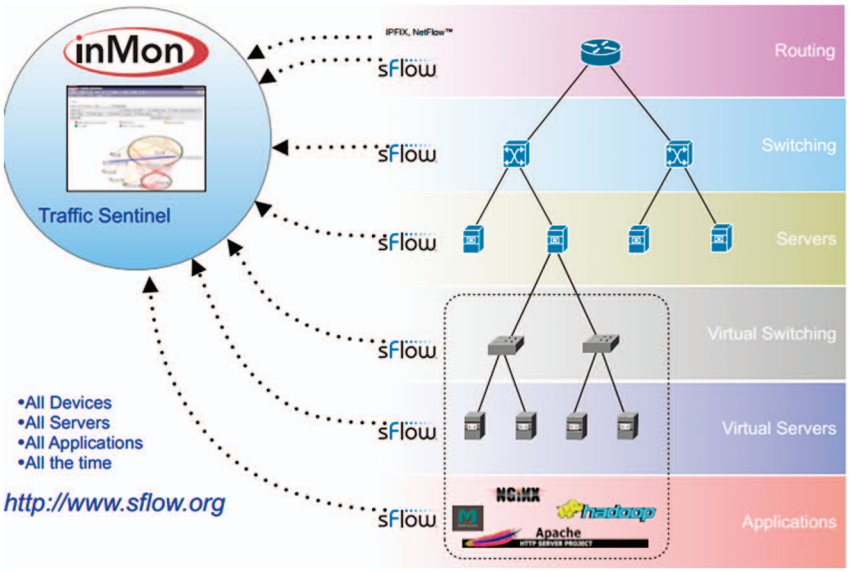

图 2 sFlow 可采集的数据

明的流量日志输入的方法，IPFIX 是基于 NetFlow 的 IETF 的标准。sFlow 是 InMon 公司发明的，由于设备部署数量的原因，没有得到广泛的应用。然而在基于 OpenFlow 的网络设备上，sFlow 获得了全部主流厂商的支持 [2]。

sFlow 在实时性（图 1）、信息丰富程度、日志生成的复杂度（图 2）等技术特性上比 NetFlow 更具有优势。sFlow 所包含的丰富信息，在客观上保证可以抽取出足够的流量特征，来达成前述的高层次检测目标要求。

■ 网络流量分析系统组成

通常用于检测 DDoS 攻击的系统称为网络流量分析系统。无论使用 NetFlow 还是 sFlow 流量日志进行分析，流量分析系统的结构都大体相似，都是由流量日志的生成部件和采集分析部件组成。流量日志的生成有两种技术方案，一种是使用网络设备内置的功能直接输出流量日志，另一种是将流量镜像到专门的流量日志生成设备。

如图 2 所示，sFlow 日志不仅可以用来描述网络流量，还可以用来描述主机的状态甚至服务器上的应用行为。

■ DDoS 攻击检测算法

由于 DDoS 攻击流量大多是由感染僵尸程序的主机发出的，因此流量特征应具有高度的相似性，且数量超乎寻常。通过对特定的指标（如 IP 地址的离散度、SYN 包比率、UDP 协议比率等）进行统计，可以发现 DDoS 攻击事件。除了利用指标统计的方法，还可以采用分类和聚类等机器学习算法，如决策树分析、k-mean 聚类、相似度分析等。

2.2 攻击流量的清洗

传统的流量清洗工作原理是将流向攻击目标的流量都牵引到专门的清洗设备上，将过滤后的清洁流量回注到原网络的路由器中。整个清洗过程涉及很多复杂的技术问题，如牵引和回注过程都涉及复杂的路由协议，清洗设备要利用复杂的过滤算法才能将攻击流量区分出来并丢弃。

在 SDN 网络中，可以用流表来实现牵

引和回注，也可以用流表项实现对攻击流量的丢弃。由于不需要修改路由，只需要添加相应的流表项即可实现清洗过程，对攻击的响应要比传统方法快一些。

2.3 基于 SDN 的 DDoS 解决方案

用 SDN 网络设备实现清洗过程，理论上可以有三种情况：1）用传统的 BGP 路由实现牵引和回注，通过流表项实现攻击流量的清洗（如图 3 所示）。2）通过流表项实现流量的牵引和回注，而清洗过程可以基于 SDN（如图 4 所示，边界路由器旁挂的

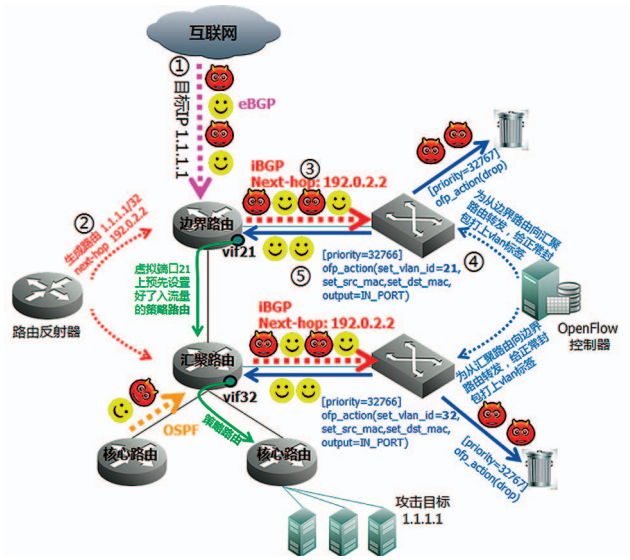

图 3 基于 BGP 的牵引，基于 SDN 的流量清洗和回注 [3]

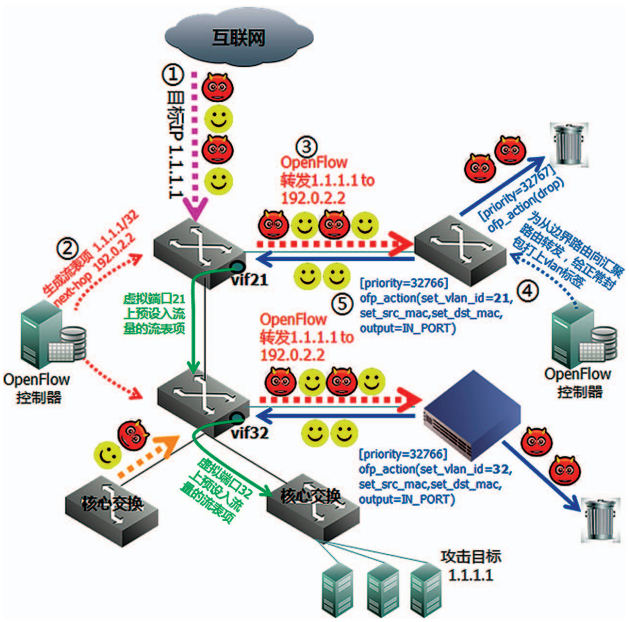

图 4 基于 SDN 的牵引回注和流量清洗

OpenFlow 交换机），也可以仍然采用传统的专用设备（如图 4 所示，汇聚路由器旁挂的专用清洗设备）。3）不牵引和回注，直接利用攻击特征生成一个用于清洗的流表项，在 SDN 网络设备上直接将攻击流量丢弃。

3. 基于 SDN 的防火墙

3.1 基于 SDN 的防火墙原理

OpenFlow 协议中规定的流表项格式与防火墙 ACL 规则非常类

似，因此很容易利用流表项构建 ACL，从而实现防火墙功能。如图 5 中的流表项，就是一个典型的 ACL，用来关闭所有 TCP 端口 22 的访问。

我们可以通过描述控制器与防火墙之间的消息交换流程来说明

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

图 5 流表项形式的防火墙 ACL

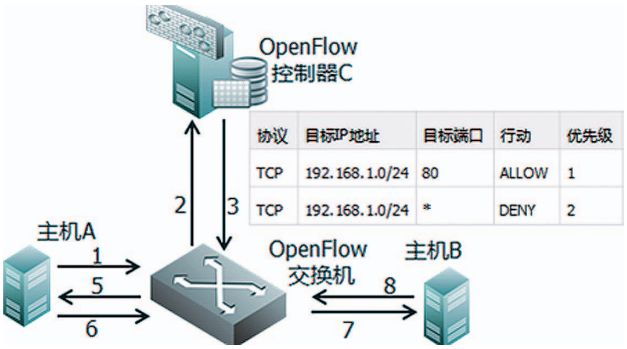

图 6 基于 OpenFlow 的防火墙与通讯主机间的消息传递

基于 OpenFlow 的防火墙的工作原理。如图 6 中，主机 A 和主机 B 之间的通讯经过 OpenFlow 交换机，通讯过程是按下列步骤进行的：

1）主机 A 向 OpenFlow 交换机发出访问主机 B 的请求，如果这个请求在 OpenFlow 交换机上无匹配的转发规则，则该请求包的拷贝会被转发到控制器 C 上，在 OpenFlow 规范中称为“packet-in”。

2）OpenFlow 交换机与控制器 C 通讯，控制器 C 上的防火墙应用模块检查访问策略。

3）控制器 C 返回响应结果。

4）规则“来自主机 A 的数据可以被发送到主机 B”安装到 OpenFlow 交换机。

5）OpenFlow 交换机响应主机 A。

6）数据从主机 A 转发到 OpenFlow 交换机。

7）数据从 OpenFlow 交换机转发到主机 B。

8）来自主机 B 的数据转发到 OpenFlow 交换机但是不会被转发到主机 A，因为没有个规则。

每个由数据流中第一个包所触发的 packet-in 将与一组防火墙规则匹配。[4] 防火墙规则按照优先级存储，并与 packet-in 的包头字段进行匹配，直到匹配到相应的规则或到达规则集的最末端。优先级最高的规则决定对流的最终的处理（允许 / 拒绝）。由此可见，在 SDN 网络中，防火墙以反应模式进行工作，控制部分已经作为一个软件应用转移到控制器中，而负责转发硬件则存在于 OpenFlow 交换机上。

目前基于 OpenFlow 的防火墙还存在两个局限 [4]：

1)调用防火墙模块 DELETE REST API 不能删除交换机中的流，规则只能在交换机流条目到它的过期时间时才能删除，这就意味着删除一个规则只能在一段时间之后才能起效，主要有连续的流量通过交换机现有的流就会一直存在。

2）通常防火墙规则支持 TCP/UDP 端口范围，但是 OpenFlow 流匹配机制不允许指定端口范围，所有这个特性没有实现。

3.2 基于 SDN 防火墙的实现

Floodlight 防火墙模块

Floodlight 开放 SDN 控制器项目中，实现了一个防火墙模块。这个模块将 ACL 部署到 OpenFlow 交换机上执行。控制器中的防火墙模块监控 packet-in 行为，packet-in 数据包头与防火墙规则集按优先级顺序进行匹配。匹配上的规则下发到 OpenFlow 交换机上。

GoGrid 的防火墙服务

GoGrid 是一家比较成功的云设施服务提供商。2013 年 4 月，GoGrid 宣布提供基于 SDN 架构的防火墙服务。客户可以利用这种低成本、高可用性的为云计算打造的解决方案来保护他们的设施，而不用在一个主机上或通过第三方设备管理防火墙。

4. 结束语

基于 SDN 的架构来实现网络安全，具有诸多的优势，主要体现在以下几个方面：

性能改善

基于 SDN 的安全解决方案，将解决超大规模云计算数据中心内部传统安全设备无法满足性能要求的问题。

效率提升

SDN 技术使安全设备的管理自动化、服务开通、网络安全管理等方面的效率都得到大幅提高。数据平面与控制平面的分离，使得网络管理者可以通过可控的软件部署相关功能，灵活地定制自己需求的服务。安全防护作为网络功能的提供也变得异常简单，比如防火墙和入侵检测等策略的部署，而不必像以前那样等待某个设备提供商在其专有设备中加入相应策略。

成本降低

SDN 的开放架构有助于打破对单一厂商的依赖，性能和效率的提升也会对降低成本起到直接的促进作用。

SDN 目前尚处于早期发展阶段，对网络安全的全部影响还无法完全知道。但是 SDN 将改变企业的网络设计和运营，并为网络安全自动化和效率改进提供机会。

参考文献

[1] "Software defined networking" http://blog.sflow.com/2012/05/software-defined-networking.html

[2] 支持 sFlow 的网络设备 http://www.sflow.org/products/network.php

[3] Tamihiro Yuzawa，“OpenFlow 1.0 Actual Use-Case: RTBH of DDoS Traffic While Keeping the Target Online” http://packetpushers.net/openflow-1-0-actual-use-case-rtbh-of-ddos-traffic-while-keeping-the- target-online/

[4] Firewall (Dev) http://docs.projectfloodlight.org/display/floodlightcontroller/Firewall+(Dev)

基于安全属性违例的工控协议异常行为分析

战略研究部　忽朝俭　李鸿培

关键字：工业控制系统　数据采集与监视控制　工控协议　安全属性

摘要：工业控制系统广泛应用于电力、油气、市政、水利、铁路、化工和制造等行业的数据采集与监视控制。本文基于对此类系统使用的典型协议中存在的安全问题及其根源的分析，提出一种针对工控协议的安全属性违例模型，并使用该模型对两种典型的工控协议中存在的异常行为进行了分析。

引言

工业控制系统（Industrial Control Systems，简称 ICS）中存在的任何缺陷均可能造成严重的社会影响和各方面的损失，因此有必要对其安全性进行研究。典型的 ICS 通常是基于网络的系统，使用特定的协议，针对这些特定的协议（下文简称为工控协议）进行安全分析，对于全面地认识 ICS 可能存在的安全问题至关重要。

一、相关工作

Sun 的报告 [1] 中对通过窃听技术截获网络数据、通过中间人攻击修改网络数据、通过拒绝服务中断网络流、通过欺骗伪造认证（报告使用 authority，应是 authentication）进行了分析，这四种攻击分别违反了保密性、完整性、可用性和认证四种安全属性。P. Huitsing[2] 和 Samuel East[3] 分别使用 [1] 中分析的 4 种安全属性对 Modbus 和 DNP3 可能遭受的攻击进行了分析。微软 MSDN 杂志 [4] 中提出了从伪装（Spoofing）、篡改（Tampering）、抵赖（Repudiation）、信息泄露（Information Disclosure）、拒绝服务（Denial of Service）和提升权限（Elevation of Privilege）6 个方面对威胁进行建模的 STRIDE 模型，STRIDE 模型对本文提出的安全属性模型具有重要的启发作用。

二、工控协议异常行为分析

（一）常用工控协议

由于各行业对 ICS 的要求不同，因此 ICS 在各行业的应用情况差异较大。根据工控网的统计 [5]：在中国，电力行业中 ICS 应用最为广泛，约占整个 ICS 市场的半壁江山，其中变电站自动化约占 ICS 市场的 40%，调度自动化约占 ICS 市场的 10%；市政行业约占 ICS 市场的 30%，包括供水、供电、供暖、供气、水处理和交通等多个具体行业；油气管线行业约占 ICS 市场的 8%；其他行业约占 ICS 市场的 12%，包括水利、铁路、化工和制造等多个具体行业，如图 1 所示。

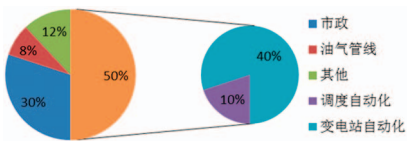

图 1 工业控制系统市场分布情况

ICS 在各行业的应用情况差异造成其在各行业的发展也不完全相同。其中，应用最为广泛的电力行业发展势头最好，发展时间最长，技术最先进，使用的协议也最多，基本上所有的典型工控协议在电力行业均有使用，例如 Modbus、Profibus、DNP3、IEC 60870-5-101/104、ICCP（即 IEC 60870-6 或者 TASE.2）和 IEC 61850 等；其他行业的 ICS 发展相对滞后，使用情况比较类似且相对电力行业的 ICS 要简单的多，采用的协议主要是 Modbus、Profibus 和 DNP3 等，如表 1 所示。

表 1 典型工控协议在 ICS 中的使用情况

行业	细分	协议
电力	变电站自动化	Modbus、Profibus、DNP3、IEC 60870-5-101/104、ICCP（IEC 60870-6, TASE.2）、IEC 61850
	调度自动化	
油气	油气管线、油气井	Modbus、Profibus、DNP3
市政	供水、供电、供暖、供气、水处理、交通	
其他	水利、铁路、化工、制造	

（二）安全问题根源

ICS 会遭遇到与传统信息技术相同的安全问题。绝大多数工控协议在设计之初，关注于效率以支持经济需求，关注于实时性以支持精确需求，关注于可靠性以支持操作需求，但是通常会为了这些需求而放弃一些并不是绝对必需的特征或功能。例如，认证（Authentication）、授权（Authorization）和不可抵赖（Non-repudiation）等需要附加开销的安全特征和功能。认证的重要性在于，保证收到的消息来自合法的用户，允许未认证用户向系统发送控制命令会造成巨大的危害；授权的重要性在于保证不同的特权操作需要由拥有不同权限的认证用户来完成，可降低误操作与内部攻击的概率；不可抵赖的重要性主要体现在事后审计和防止非法扫描与暴力破解上。

ICS 也会遭遇到很多不同于传统信息技术的安全问题，其根源在于其安全目标不同 [6]。在传统的信息技术领域，通常将保密性（Confidentiality）、完整性（Integrity）和可用性（Availability）称为安全的三种基本属性。并且通常认为保密性的优先级最高，完整性次之，

可用性最低。而 ICS 领域正好相反，一般认为可用性最重要，完整性次之，机密性最低。

ICS 传输的通常是监视控制和数据采集信息，这些原始数据多是实时数据且需放在特定背景下分析才有意义，对外部人员价值不大，因此机密性的相对重要性要低。完整性的重要性在于，错误的数据可能剧烈地改变设备的控制设置或者引起系统功能失常，从而造成潜在的危害。可用性的重要性在于，攻击者的主要目标可能就在于通过拒绝服务攻击瘫痪网络，从而造成巨大的安全危害。

（三）安全属性违例模型

要全面地认识系统可能存在的安全问题，除了要理解安全的三种基本属性，也应该考虑其他一些用于描述信息安全不同特性的属性。

基于对几种常见的工控协议中存在的安全问题及其根源的深入研究，本文提出一种针对工控协议异常行为分析的安全属性违例模型。该模型基于保密性、完整性、可用性、认证、授权和不可抵赖这六种安全属性，如图 2 所示。

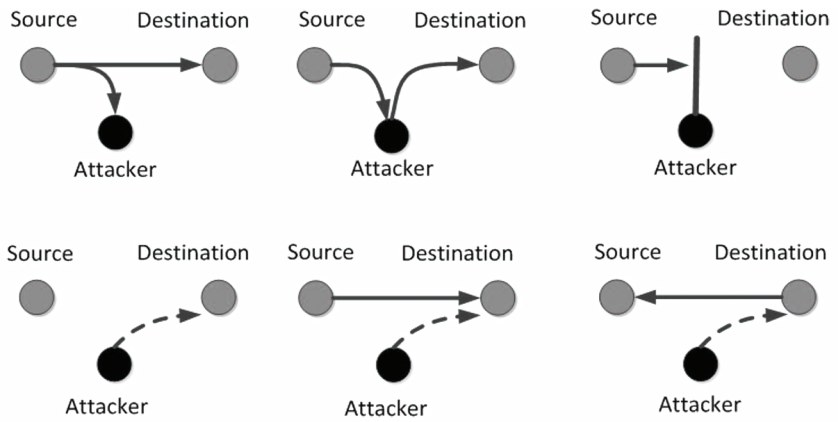

图 2 对六种安全属性的违反情况

保密性违例时，信息被窃取；完整性违例时，信息被修改；可用性违例时，信息不可达；认证违例时，非法访问；授权违例时，越权访问；不可抵赖违例时，不可追溯信息来源。

三、实例分析

基于上文提出的安全属性违例模型，对特定的工控协议数据格式和交互序列进行具体分析，可以总结并枚举出该协议中可能出现的违反特定安全属性的异常行为。基于该思路，本节对两种常见的典型工控协议 Modbus[7] 和 DNP3[8] 中可能出现的异常行为进行了分析。

（一）Modbus

Modbus 采用主 - 从结构，提供连接到不同类型总线或网络的设备之间的客户机 - 服务器通信。客户机（主站）使用不同的功能码请求服务器（从站）执行不同的操作，服务器执行功能码定义的操作并向客户机发送响应，或者在操作中检测到差错时发送异常响应。

Modbus 最初是基于串行链路的现场（总线）协议。随着以太网技术的出现、发展和互联网技术的日益进步，Modbus 也不断得到更新和扩展，例如使用以太网的 Modbus TCP 等。

虽然 Modbus 报文可能在不同的传输介质上传输，但是其应用数据单元基本保持不变。Modbus 中可能存在的异常行为如表 2 所示。

表 2 Modbus 中可能存在的异常行为

序号	异常行为描述	保密性	完整性	可用性	认证	授权	不可抵赖
1	强制从站进入只听模式（08-04）			√	√	√	√
2	复启通讯会话（08-01）			√	√	√	√
3	重置诊断信息（例如计数器或诊断寄存器等）（08-0A）			√	√	√	√
4	请求从站标志信息（43-14）	√			√	√	√
5	请求从站附加信息（17）	√			√	√	√
6	不合法报文长度（2 个字节表示长度），潜在拒绝服务攻击		√	√			
7	非 Modbus 协议运行在 TCP 的 502 端口				√		
8	从设备忙异常代码延迟（异常码 06），潜在拒绝服务攻击			√	√		
9	确认异常代码延迟（异常码 05），潜在拒绝服务攻击			√	√		
10	不正确的报文长度（最大 253），潜在拒绝服务攻击		√	√			
11	配置扫描（例如定义的点列及值）（30 秒内 5 个异常码 02）	√			√	√	√
12	可用功能码扫描（60 秒内 3 个异常码 01）				√	√	√
13	修改分隔符（08-03）			√	√	√	√
14	周期较短（实际阈值待定）的无意义命令，暴力拒绝服务			√	√		
15	广播性质的报文或一个主站向多个从站的请求				√	√	
16	包含在异常协议数据单元中的信息				√		

（二）DNP3

除了与 Modbus 工作方式类似外，DNP3 最初也基于串行链路的现场（总线）协议，后来扩展到以太网和 TCP/IP 上实现。DNP3 对数据链路层、伪传输层和应用层进行了描述。与 Modbus 不同的是，TCP/IP 上的 DNP3 并没有对串行链路上的 DNP3 做任何实质上的修改，而是将整个链路规约数据单元（LPDU）作为 TCP/IP 之上的应用层数据进行传输（也可能是基于 UDP），故 3 个层次均可能受到威胁。DNP3 中可能存在的异常行为如表 3 所示。

表 3 DNP3 中可能存在的异常行为

序号	异常行为描述	保密性	完整性	可用性	认证	授权	不可抵赖
1	关闭主动上送（21）			√	√	√	√
2	DNP3 端口（TCP/UDP20000）运行非 DNP3 通信				√		
3	长时间多重主动上送（响应风暴）				√		
4	授权客户冷重启（13）			√		√	√
5	未授权客户冷重启（13）			√	√	√	√
6	停止应用（18）			√	√	√	√
7	热重启（14）			√	√	√	√
8	授权客户的广播请求					√	
9	未授权客户的广播请求				√	√	
10	配置扫描（例如定义的点列及值）（30 秒内检测到至少 5 个指示字的第 9 或 10 位设置为 1）	√			√	√	√
11	可用功能码扫描（60 秒内 3 个指示字的第 8 位设置为 1）				√	√	√
12	更改时间（功能码 02，对象类型 50）			√	√	√	√
13	校验和错误				√		
14	认证失败					√	√
15	数据流控制标志（DFC）欺骗			√	√		
16	重新初始化数据对象（15）			√	√	√	√
17	重新初始化应用（16）			√	√	√	√
18	冰冻并清除可能重要的状态信息（9）		√		√	√	√
19	无通告冰冻并清除可能重要的状态信息（10）		√		√	√	√
20	未授权的操作（4）、直接操作（5）、无通告直接操作（6）				√	√	
21	源自或到达非显式认证的 DNP3 设备的 DNP3 通信				√		
22	配置崩溃（指示字的第 13 位设置）	√	√	√	√		
23	应用数据层 FIR/FIN 标记报文重组攻击		√	√	√		
24	应用数据层传输序号报文重组攻击		√	√	√		
25	伪传输层 FIR/FIN 标记报文重组攻击		√	√	√		
26	伪传输层传输序号报文重组攻击		√	√	√		
27	服务不可用或未实现（数据链路层功能码 14 或 15）			√	√		
28	重置用户进程（数据链路层功能码 1）			√	√	√	√
29	伪造数据链路层广播地址并向所有的从站发送错误请求				√		

四、结束语

为增强 ICS 的整体安全性，有必要改善工控协议的安全特征。对协议中可能存在的异常行为进行分析将有助于暴露协议中存在的安全问题，进而指导安全机制的开发和 IDS/IPS 规则的编写，并最终合并到协议描述中。

参考文献

[1] Sun Microsystems. Secure Enterprise Computing with the Solaris 8 Operating Environment.

www.sun.com/software/whitepapers/wp-s8security/wp-s8security.pdf

[2] P. Huitsing, R. Chandia, M. Papa and S. Shenoi. Attack taxonomies for the Modbus protocols. International Journal of Critical Infrastructure Protection, 2008, 1:37-44.

[3] S. East, J. Butts, M. Papa and S. Shenoi. A TAXONOMY OF ATTACKS ON THE DNP3 PROTOCOL. Critical Infrastructure Protection III, IFIP AICT 311, 2009, 67-81.

[4] Microsoft MSDN. Uncover Security Design Flaws Using the STRIDE Approach.

http://msdn.microsoft.com/zh-cn/magazine/cc163519.aspx

[5] 中国工控网 , SCADA 系统市场分析 , www.gongkong.com

[6] Guide to Industrial Control Systems (ICS) Security : NIST, SP800 82.,June,2011.

[7] Modbus IDA. MODBUS Application Protocol Specification v1.1a.

www.modbus.org/specs.php

[8] IEEE Power & Energy Society. IEEE Standard for Electric Power Systems Communications Distributed Network Protocol (DNP3), IEEE Std 1815-2010.

一种有效的CSRF漏洞检测技术

产品管理中心 向智　西安研发中心 邓永凯

关键字：CSRF　漏洞检测

摘要：OWASP TOP10 2013 排名第 8 的是跨站点请求伪造（CSRF）漏洞，Web 2.0 可以说是的它催化物，利用这种漏洞，黑客完全可以在用户毫无察觉的情况下发起攻击，造成网站被篡改、数据泄密等危害，甚至形成大规模的蠕虫攻击。但是由于 CSRF 漏洞特有的特性，造成很难对其有效地进行检测。本文主要从 CSRF 漏洞的原理入手，试图找寻一种更有效的检测方法。

一、引言

早在 1988 年，Norm Hardy 发布了第一篇关于跨站点请求伪造（Cross-Site Request Forgery，简称 CSRF）漏洞最初描述的文章，把其描述为一个应用程序级别的信任问题，并称之为“混淆责任（confused deputy）”。到 2000 年，在 bugtraq 有一篇文章解释了 ZOPE（一个开放源代码的 Web 应用服务器）是怎样被一个“混淆责任”的 Web 问题所影响，这个问题正是今天定义的 CSRF 漏洞。在后来的 2001 年，Peter Watkins 也在 bugtraq 的邮件列表条目里专门设置了 CSRF 术语，用以回应另一个称为“允许用户 POST 图片的危害”的问题。

CSRF 这种需要利用一定社会工程技巧以达到伪造效果的漏洞，随着 Web 优秀的交互体验技术的发展，例如 Web 2.0、HTML 5 等，利用变得更加简单和有效，也越来越被黑客所青睐。从 CVE 漏洞库检索到（如图 1）CSRF 漏洞数在 2008 年达到一个高峰后（258 个），逐步平稳在 90 以下，但在 2012 年又出现了一个小高潮（153 个），从数据可看到这种漏洞尚处于不规则发生期，同时，随着 Web 应用快速的发展，这种漏洞也存在着潜在爆发的可能。因此，也可以看到国际 OWASP 组织（Open Web Application Security Project）在 2007 年首次把 CSRF 漏洞收入到其 TOP10 漏洞后，在最新的 2013 年 TOP10 漏洞中仍将其继续收入。

同时，CVE 收录的 CSRF 漏洞大多是基于已知 Web 应用的，

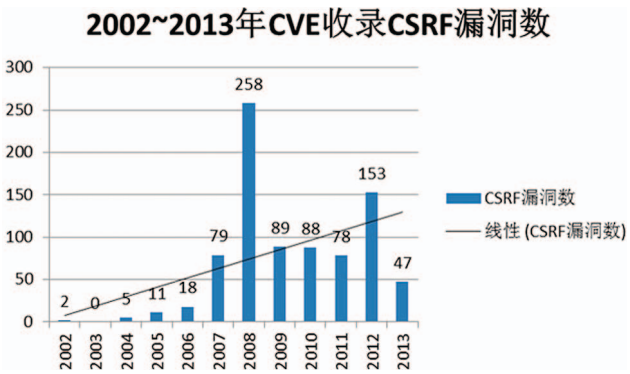

图 1 2002~2013 年 CVE 收录 CSRF 漏洞数 [1]

而对于网站中的 Web 应用系统而言，通常是根据不同业务目标进行了定制化开发，因此对于这些非已知或者说是定制化系统而言，同样存在 CSRF 漏洞，例如国内某著名第三方漏洞发布平台就发布了不少这样的存在 CSRF 漏洞的网站。

二、CSRF 漏洞定义

首先来看看 CSRF 漏洞的定义，WIKIPEDIA 对其的定义为：Cross-site request forgery, also known as a one-click attack or session riding and abbreviated as CSRF (sometimes pronounced sea-surf) or XSRF, is a type of malicious exploit of a website whereby unauthorized commands are transmitted from a user that the website trusts. Unlike cross-site scripting (XSS), which exploits the trust a user has for a particular site, CSRF exploits the trust that a site has in a user's browser[2]。

OWASP 对其的定义为：CSRF is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated. With a little help of social engineering (like sending a link via email/chat), an attacker may force the users of a web application to execute actions of the attacker's choosing. A successful CSRF exploit can compromise end user data and operation in case of normal user. If the targeted end user is the administrator account, this can compromise the entire web application[3]。

CWE 对其的定义为：The web application does not, or can not, sufficiently verify whether a well-formed, valid, consistent request was intentionally provided by the user who submitted the request. When a web server is designed to receive a request from a client without any mechanism for verifying that it was intentionally sent, then it might be possible for an attacker to trick a client into making an unintentional request to the web server which will be treated as an authentic request. This can be done via a URL, image load, XMLHttpRequest, etc. and can result in exposure of data or unintended code execution[4]。

从这三个定义可以看到，CSRF 漏洞主要具备了以下两个特性：

1. 受信任的用户被非受信任的用户利用对目标站点进行了非本意的伪造的请求；

2. 这种请求是跨站点的。

它与跨站脚本漏洞（XSS）的主要区别在于：XSS 漏洞利用的是站点对某个用户的信任，而 CSRF 漏洞利用的是站点对用户浏览器的信任，信任对象的不同，一个是用户本身，一个是浏览器。

三、CSRF 漏洞原理及危害

从 CSRF 漏洞定义可以看到，通过利用这个漏洞可以诱使受信任的用户在目标网站进行未经授权的操作，包括以受信任用户的名义发送邮件、发送消息、更改资料、获取账号信息、实现虚拟货币转账、以及现在流行的刷粉丝等未经授权的操作，从而造成个人隐私泄露、信息篡改、财产损失等。

那么是如何利用 CSRF 漏洞造成这些危害的呢？下面从几个典型的漏洞利用方式进行逐一阐述。

3.1 GET 方式 CSRF 漏洞利用

GET 请求方式的 CSRF 漏洞利用可参考图 2。

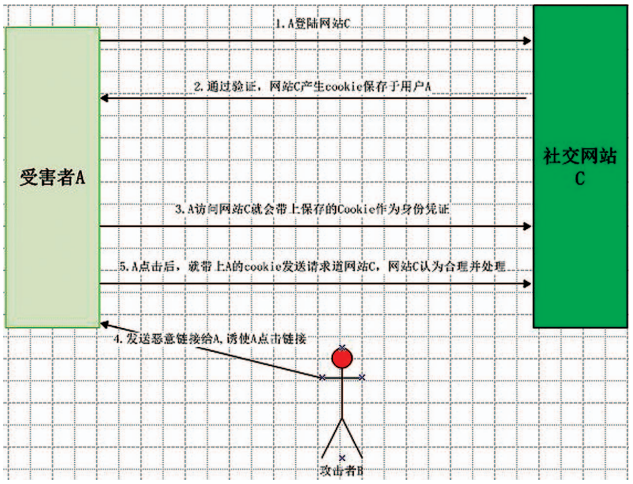

图 2 GET 方式 CSRF 漏洞利用

如图 2 所示，用户 A 现在登录了社交网站 C，每次访问 C 时浏览器都会自动带上 A 登录时留下的 Cookie。现在社交网站 C 存在 CSRF 漏洞，攻击者 B 给用户 A 发送了一个链接，或者攻击者 B 在网站上给用户 A 留了个言，内容就是本条链接。而此攻击链接的目的就是添加攻击者 B 为用户 A 的好友，或者是说用户 A 成为了攻击者 B 的粉丝。当用户 A 打开此恶意链接时，用户 A 就添加了攻击者 B 为好友，成为攻击者 B 的粉丝，这样就可以达到了刷粉丝效果。

攻击者 B 发送给用户 A 的链接可如下所示：

```
http://www.XXXX.com/addfriends.php?user_id=11223344
```

此链接中 user_id 的值就是攻击者 B 在社交网站 C 中的惟一身份表示 ID。当用户 A 打开此链接时，用户 A 就带上它的 Cookie 发送了一个 GET 请求，这时服务器认为这次请求是用户 A 发送的正常请求，于是效果就是用户 A 成功添加攻击者 B 为好友，成为攻击者 B 的粉丝。

对于一些看似需要 POST 方式才能起作用的 Web 应用，如果尝试把 POST 改成 GET，往往也能达到同样的效果。

3.2 POST 方式 CSRF 漏洞利用

POST 请求方式的 CSRF 漏洞利用可参考图 3。

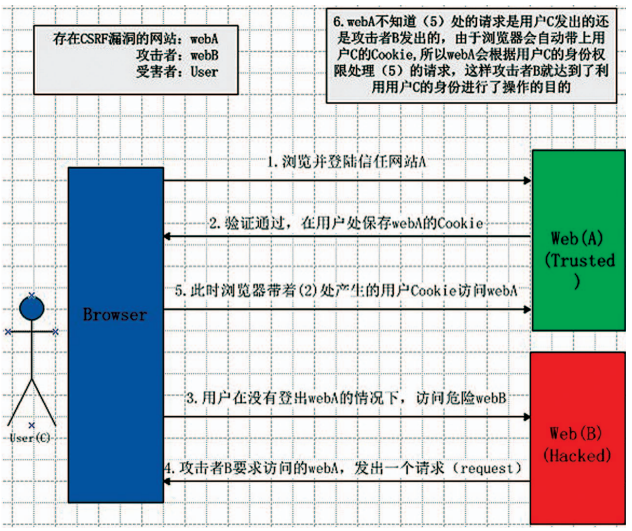

图 3 POST 方式 CSRF 漏洞利用

如图 3 所示，存在 CSRF 漏洞的网站是 WebA，当用户 C 登录了 WebA，并且没有登出。此时无论因为什么诱惑或者其他原因，用户 C 访问了攻击站点 WebB，攻击站点 WebB 的内容就是发送一个 POST 请求到 WebA。于是攻击站点 WebB 在此时要访问 WebA，攻击站点 WebB 发出了一个带有用户 C 在 WebA 的登录凭证 Cookie，WebA 服务器收到这个请求后，认为这是用户 C 发送的正常请求，于是就以用户 C 的身份处理了此次请求。

当用户 C 现在登录着社交网站 WebA，由于此社交网站 WebA 某处有 CSRF 漏洞。这个 CSRF 请求必须是 POST 的，于是攻击者在其攻击站点 WebB 上构造好了一个页面，此页面会发送 POST 请求到社交网站 WebA，内容是添加攻击者为好友。

请求内容如下：

```
POST /addfriends.php HTTP/1.1

uid=【粉丝 id】&aid=【关注的对象 id】&name=【可以不填】&is_follower=【可以不填】
```

当用户 C 在登录着社交网站 WebA 的同时访问了攻击者的网站 WebB，这时就会发送一个请求到 WebA。由于此请求带有用户 C 当前的 Cookie，于是社交网站 WebA 认为此请求是用户 C 发的，服务器 WebA 正常处理了此请求，结果就是添加了攻击者为好友，成为了攻击者的粉丝。

上面的过程就是利用 POST 的请求方式完成了一次 CSRF 攻击。这些请求都是在没有判断请求来源、没有设置同源策略的前提下进行的。因为你的 POST 大多会放到攻击者自己能控制的非目标域空间下，这样就需要进行跨域请求了。

3.3 利用 CSRF 漏洞形成 CSRF 蠕虫

把 CSRF 蠕虫单独列出来进行说明，主要是因为其影响大、传播广，利用方式可参见图 4。

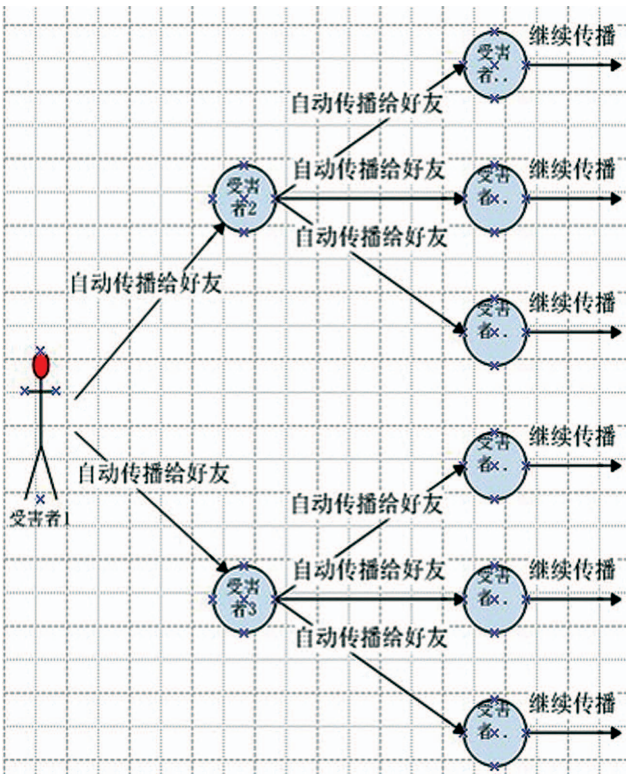

图 4 利用 CSRF 漏洞形成 CSRF 蠕虫

CSRF 蠕虫可以理解成能快速自动传播的多个单次 CSRF 攻击。当受害者触发了 CSRF 后，此次攻击还会因为受害者的此次触

发操作继续传播到其他用户，然后不断继续下去，攻击和影响更多的受害者。

我们用一个例子来进行说明，如图 4 所示，当受害者 A 登录了某存在 CSRF 漏洞的社交网站，一直没有登出，发现自己有一条私信，内容很诱惑，不小心 A 点击了这个私信里面的链接，这个链接的内容就是遍历 A 的好友列表，并给每一个 A 的好友都发送一条私信，内容相同，并且自动发送一条动态，于是这个 CSRF 蠕虫就开始传播了。当 A 的好友发现自己有私信时，进行了与 A 同样的操作，这样 CSRF 蠕虫就自动传播下去了。

上面的例子中，首先此社交网站必须存在 CSRF 漏洞，然后找到 CSRF 的利用点，如发私信、发动态等，而且最好有跨域请求，这样才能保证这个 CSRF 蠕虫要进行的动作能顺利进行，也就是 CSRF 蠕虫传播的可能性。下面就是这个蠕虫的传播性了。和单次 CSRF 攻击相比，蠕虫必须能自动传播下去，这就必须有用户群的驱动，而为了区分用户群，必须要每一个用户的惟一标识了，如用户的 user_id 等，这样就能不断传播下去。

3.4 小结

从上面的 GET、POST 方式和形成蠕虫原理可以看出，要完成一次 CSRF 攻击，受害者一般需要依次完成以下两个步骤：

1. 登录受信任网站 A，并在本地生成 Cookie。

2. 在不登出 A 的情况下，访问危险网站 B（或者访问了存在攻击行为的链接）。

虽然说可以尽量减少以上操作来避免 CSRF 攻击，但由于互联网易用、开放等特性，往往很难做到，比如：

1. 用户往往不会在登录了一个网站后，不再打开一个 tab 页面并访问另外的网站。

2. 用户往往在关闭浏览器了后，不能保证本地的 Cookie 立刻过期，使上次的会话结束。

3. 用户有时也会因为判断的原因，而去点击看起来很可信的链接，进而访问了危险页面。

因此 CSRF 漏洞也一直困扰着网站安全运维人员，如何通过自动化检测工具准确地检测出网站的 CSRF 漏洞成为其亟待解决的问题。

四、CSRF 漏洞检测技术探讨

就 CSRF 漏洞检测技术而言，没有一种特别合适的检测方法，类似于 SQL 注入、XSS 的检测方式，普遍采用的方法可以看成是一种“反证法”，从检验防御效果的角度来进行检测。

这类的方法一般包括了以下几种：

方法一，通过修改检测报文的 Referer 值，比较修改前后两次检测的响应内容是否相同或者相近。如果相同或者相近，则认为有可能存在 CSRF 漏洞。

方法二，使用相同的用户名和密码登录网站两次，相互替换两次生成的 sessionid，再请求同一个链接，查看替换前后的两次请求响应内容是否相同或者相近。如果相同或相近，则认为可能存在 CSRF。

方法三，在结合前两种检测方式的基础上，再增加对

crossdomain.xml 的检测，检查 allow-access-from 是否设置为 *。如果满足这三点，则认为可能存在 CSRF。

方法四，在对目标发送请求时，先后连续发送两次请求，看两次请求的内容以及连续请求的响应是否一致。如果请求内容一致，响应也一致，则认为可能存在 CSRF。

方法五，在同域名、同身份的情况下，如果某个 POST 的请求在变成 GET 请求发送后，返回的响应内容与 POST 请求本身返回的响应内容相同，则认为可能存在 CSRF。

这五种方法的优缺点如表 1。

表 1 CSRF 五种检测方法优缺点分析

方法	优点	缺点
方法一	使用简单	误报率较高
方法二	准确性较高	登录时容易受限于验证码
方法三	准确性高	当前域情况下容易漏报
方法四	使用简单	误报率较高
方法五	在当前域下，准确性较高	误报率较高

当然，现有检测技术仍有不少的局限性，比如以下几点：

1. 检测规则太少，响应内容对比相似度无标准，会出现大量的误报。

2. 对登录界面的要求太过苛刻，容易受限于验证码。

3. 在同域和跨域检测方法上，实现难易程度不统一。

4. 自动检测结果和手工实际场景攻击检测结果无法一一对应，导致无法判断是插件漏报还是误报，从而影响对插件规则进行修改和完善。

通过对类似于 DVWA、OWASP 等测试平台的分析，试图从这些局限性出发，找到一些新的解决方案，具体方法如下：

方法一：

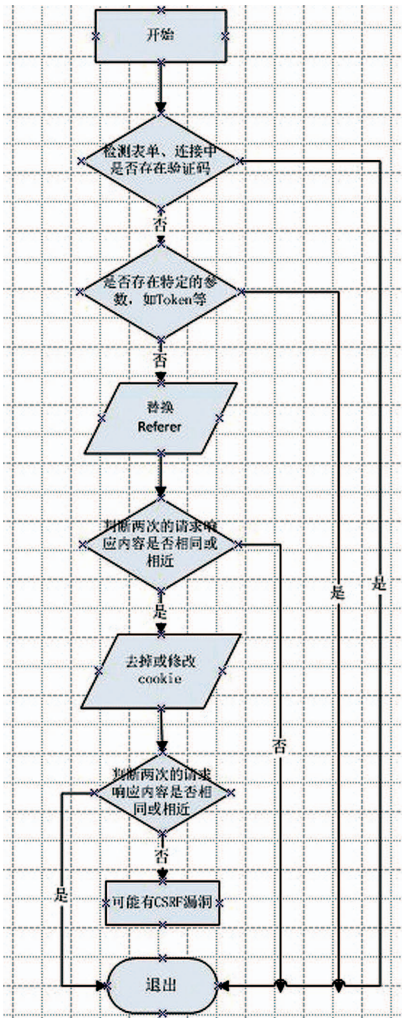

图 5

如图 5 所示：

第一步，为了避免出现误报，对当前页面或者链接以及 POST 表单中是否存在验证码，通过关键字等进行一个简单的判断。有验证码则退出，否则进行下一步。

第二步，如果当前链接的参数或者 POST 表单数据以及 Cookie 中存在类似 Token、formhash 等参数，或者两次相同的请求，请求内容以及响应内容不一致，则该站点可能对 CSRF 做了防御或者预防措施（但是会有漏报出现），认为当前链接不存在 CSRF 漏洞，退出，否则进入下一步。

第三步，替换 Referer，判断两次请求的响应内容是否相同或者相近，如果不同，则站点可能已经对 CSRF 做了防御（但是会有漏报出现），退出，否则进行下一步。

第四步，去掉 Cookie 或者修改 Cookie 值（可能会有漏报出现），判断两次请求的响应内容是否相同或者相近，如果不同，则说明此链接的访问受 Cookie 限制，因此判断此链接可能存在 CSRF，进行下一步，否则退出。

第五步，如果前面四个步骤都没有退出，到这里则认为该站点或者此链接可能存在 CSRF 漏洞。

方法二：

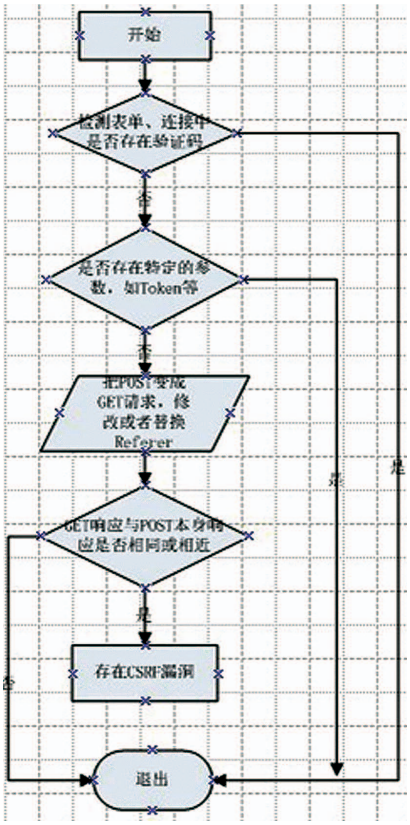

图 6

如图 6 所示，根据平时在实际场景中对 CSRF 漏洞进行检测的方法和判断标准，提出以下方法：

第一步，为了避免出现误报，对当前页面或者链接以及 POST 表单中是否存在验证码，通过关键字等进行一个简单的判断。有验证码则退出，否则进行下一步。

第二步，如果当前链接的参数或者 POST 表单数据以及 Cookie 中存在类似 Token、formhash 等参数，或者两次相同的请求，请求内容以及响应内容不一致，则该站点可能对 CSRF 做了防御或者预防措施（但是会有漏报出现），认为当前链接不存在 CSRF 漏洞，退出，否则进入下一步。

第三步，在登录状态下，把每次的 POST 请求都变成 GET 请求，在发送变型后的 GET 请求的同时，修改或者替换 Referer 的值为非当前域的值，如果此 GET 请求返回的响应内容与 POST 请求本身返回的响应内容不相同或者不相近，则认为不存在 CSRF，退出，否则进入下一步。

第四步，如果上面三个步骤都没有退出，到这里则认为此站点或者此链接可能存在

CSRF 漏洞。

方法三：

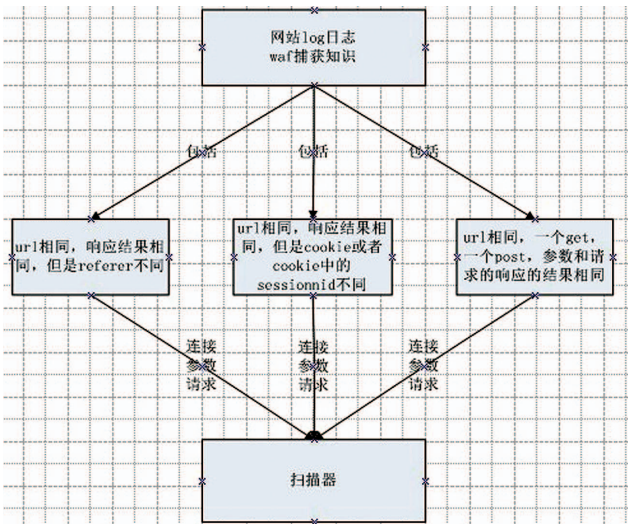

图 7

如图 7 所示，通过网站的 LOG 日志，或者 WAF 设备捕获的一些疑似 CSRF 的知识，这些知识包括：

第一种内容：两条相同的 URL 请求，两次请求的响应结果相同，但是它们的 Referer 不同（非同域名）。

第二种内容：两条相同的 URL 请求，两次请求的响应结果相同，但是它们的 Cookie 不同或者 Cookie 中的 Sessionid 不同。

第三种内容：两条相同的 URL 请求，一个是 GET 型，一个是 POST 型，它们的参数和两次请求的响应结果相同。

然后通过对这些知识的分析，以获取到疑似有 CSRF 漏洞的链接和参数，再把对应的 URL 链接和请求参数等交给扫描器，通过第一种和第二种方法，使用这些知识进一步进行 CSRF 的探测，从而判断是否存在 CSRF 漏洞。

五、结束语

CSRF 漏洞的检测和其他 Web 安全漏洞不同，因其形成原因的特殊性，给自动化工具进行检测时带来了挑战，本文也试图通过对多种检测方法的分析、比较，结合准确性、速度等因素给出了其他的一些检测思路。

限于笔者学识和所掌握信息，难免有疏漏和错误之处，请各位读者明察指正。

参考文献

【1】CVE – Search Result http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=CSRF

【2】Cross-site request forgery – Wikipedia,the free encyclopedia http://en.wikipedia. org/wiki/Cross-site_request_forgery

【3】Cross-Site Request Forgery (CSRF) - OWASP https://www.owasp.org/index.php/ Cross-Site_Request_Forgery

【4】CWE-352: Cross-Site Request Forgery (CSRF) http://cwe.mitre.org/data/ definitions/352.html

在WINDBG中定位ExceptionAddress

核心技术部　陈庆

关键字：调试　异常　windbg　second chance　ExceptionAddress

摘 要：在 windbg 中调试时，碰上 second chance 时，栈回溯已经看不到与 ExceptionAddress 直接相关的信息，但我们调试的目的就是要找到 ExceptionAddress。本文介绍了几种通用思路来解决这个问题。

一、问题起因

借助 ping.exe 的进程空间，临时研究一下 Intel 指令前缀。用 cdb.exe 加载 ping.exe，执行到 PE 入口，人工修改附近的代码，替换成欲测试的带有指令前缀的其它指令。

```
cdb.exe -hd -o ping.exe
(c24.188c): Break instruction exception - code 80000003 (first chance)
eax=00000000 ebx=00000000 ecx=cdc70000 edx=001be1e8 esi=fffffffe edi=00000000
eip=76f50fab esp=0015f338 ebp=0015f364 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b  efl=00000246
ntdll!LdrpDoDebuggerBreak+0x2c:
76f50fab cc              int     3
> g $exentry
eax=74e83398 ebx=7efde000 ecx=00000000 edx=002f2aa7 esi=00000000 edi=00000000
eip=002f2aa7 esp=0015f7bc ebp=0015f7c4 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b  efl=00000246
ping!mainCRTStartup:
002f2aa7 e805030000      call    ping!__security_init_cookie (002f2db1)
> eb eip f0 cc 90 90
```

```
> u eip l 3
ping!mainCRTStartup:
002f2aa7 f0cc            lock int 3
002f2aa9 90              nop
002f2aaa 90              nop
> !chkimg -d -lo 1 ping // 检查文件是否被篡改?
   002f2aa7-002f2aaa  4 bytes - ping!mainCRTStartup
      [ e8 05 03 00:f0 cc 90 90 ]
4 errors : ping (002f2aa7-002f2aaa)
> p
(c24.188c): Illegal instruction - code c000001d (first chance)
(c24.188c): Illegal instruction - code c000001d (!!! second chance !!!)
eax=00000000 ebx=0015f308 ecx=00000000 edx=002f2aa7 esi=00000000 edi=00000000
eip=76ed15de esp=0015f2f4 ebp=0015f7c4 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b  efl=00000246
ntdll!NtRaiseException+0x12:
76ed15de 83c404          add     esp,4
```

结果触发 c000001d 异常，停下来的时候已是 second chance。单就本例而言，我能猜到是“lock int 3”触发异常，扩展一下，有无通用思路在这种情况下定位触发异常的指令 (ExceptionAddress)。

二、解决方案

1) 基本原理

在 cdb 中碰上 seccond chance 时，先查看调用栈回溯，但此时已经看不到与 ExceptionAddress 直接相关的信息。

```
> kpn
 # ChildEBP RetAddr
00 0015f2f4 76ec014d ntdll!NtRaiseException+0x12
01 0015f2f4 00000000 ntdll!KiUserExceptionDispatcher+0x29
```

从上述显示中看到，碰上 second chance 时，停在 NtRaiseException() 中间。检查一下 NtRaiseException() 的函数体 :

```
> uf ntdll!NtRaiseException
76ed15cc b82f010000      mov     eax,12Fh
76ed15d1 33c9            xor     ecx,ecx
76ed15d3 8d542404        lea     edx,[esp+4]
76ed15d7 64ff15c0000000  call    dword ptr fs:[0C0h]    // TEB.WOW32Reserved
76ed15de 83c404          add     esp,4
76ed15e1 c20c00          ret     0Ch
> dt ntdll!_TEB WOW32Reserved @$teb
   +0x0c0 WOW32Reserved : 0x744c2320 Void
```

NtRaiseException() 的函数原型是 :

```
NTSTATUS NTAPI NtRaiseException
(
   IN  PEXCEPTION_RECORD   ExceptionRecord,
   IN  PCONTEXT            ThreadContext,
   IN  BOOLEAN             HandleException
);
```

KiUserExceptionDispatcher() 调 用 NtRaiseException() 时, 是这样的 :

```
KiUserExceptionDispatcher( ExceptionRecord, ThreadContext )
    NtRaiseException( ExceptionRecord, ThreadContext, FALSE )
```

检查一下 KiUserExceptionDispatcher () 的函数体 :

```
> uf ntdll!KiUserExceptionDispatcher
76ec0124 fc             cld
76ec0125 8b4c2404       mov     ecx,dword ptr [esp+4]
76ec0129 8b1c24         mov     ebx,dword ptr [esp]
76ec012c 51             push    ecx
76ec012d 53             push    ebx
76ec012e e8c8b10400     call    ntdll!RtlDispatchException (76f0b2fb)
76ec0133 0ac0           or      al,al
76ec0135 740c           je      ntdll!KiUserExceptionDispatcher+0x1f (76ec0143)
ntdll!KiUserExceptionDispatcher+0x13:
76ec0137 5b             pop     ebx
76ec0138 59             pop     ecx
76ec0139 6a00           push    0
76ec013b 51             push    ecx
76ec013c e88ffd0000     call    ntdll!NtContinue (76ecfed0)
76ec0141 eb0b           jmp     ntdll!KiUserExceptionDispatcher+0x2a (76ec014e)
ntdll!KiUserExceptionDispatcher+0x1f:
76ec0143 5b             pop     ebx
76ec0144 59             pop     ecx
76ec0145 6a00           push    0     // HandleException
76ec0147 51             push    ecx   // ThreadContext
76ec0148 53             push    ebx   // ExceptionRecord
76ec0149 e87e140100     call    ntdll!NtRaiseException (76ed15cc)
ntdll!KiUserExceptionDispatcher+0x2a:
76ec014e 83c4ec         add     esp,0FFFFFFECh
76ec0151 890424         mov     dword ptr [esp],eax
76ec0154 c744240401000000 mov   dword ptr [esp+4],1
76ec015c 895c2408       mov     dword ptr [esp+8],ebx
```

76ec0160 c744241000000000 mov dword ptr [esp+10h],0

76ec0168 54 push esp

76ec0169 e89ab60400 call ntdll!RtlRaiseException (76f0b808)

76ec016e c20800 ret 8

从前面的执行显示看到，碰上 second chance 时，esp=0015f2f4，此时栈布局如下：

> dds esp l 5

0015f2f4 76ed15de ntdll!NtRaiseException+0x12

0015f2f8 76ec014e ntdll!KiUserExceptionDispatcher+0x2a

0015f2fc 0015f308 ExceptionRecord

0015f300 0015f358 ThreadContext

0015f304 00000000 HandleException

windbg 有一个 .exr 命令，用于查看 ExceptionRecord 信息：

> .exr poi(esp+8)

ExceptionAddress: 002f2aa7 (ping!mainCRTStartup)

ExceptionCode: c000001d (Illegal instruction)

ExceptionFlags: 00000000

NumberParameters: 0

windbg 另有一个 .cxr 命令，用于查看 ThreadContext 信息：

> .cxr poi(esp+c)

eax=74e83398 ebx=7efde000 ecx=00000000 edx=002f2aa7 esi=00000000 edi=00000000

eip=002f2aa7 esp=0015f7bc ebp=0015f7c4 iopl=0 nv up ei pl zr na pe nc

cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00010246

ping!mainCRTStartup:

002f2aa7 f0cc lock int 3

无论是 .exr 还是 .cxr，都直接定位了触发异常的指令 (ExceptionAddress)。

碰上 second chance 时，就这个版本的 ntdll 而言，ebx 指向 ExceptionRecord(ecx 在 76ed15d1 处已被 xor 清零)，因此可以对 ebx 使用 .exr 命令：

> .exr ebx

ExceptionAddress: 002f2aa7 (ping!mainCRTStartup)

ExceptionCode: c000001d (Illegal instruction)

ExceptionFlags: 00000000

NumberParameters: 0

简单总结一下，碰上 second chance 时，执行：

.exr poi(esp+8)

或：

.cxr poi(esp+c)

或：

.exr ebx

上述三种办法都可以快速定位触发异常的指令 (ExceptionAddress)。

2) 更简捷的方案

上面的办法是最原理性的，不过最简捷的办法是这个：

```
> .exr -1
ExceptionAddress: 002f2aa7 (ping!mainCRTStartup)
ExceptionCode: c000001d (Illegal instruction)
ExceptionFlags: 00000000
NumberParameters: 0
```

指定 -1 做 .exr 的地址时，调试器显示最近一次异常对应的信息。

另有偷懒的办法：

```
> !analyze -v
...
FAULTING_IP:
ping!mainCRTStartup+0
002f2aa7 f0cc            lock int 3
EXCEPTION_RECORD:  ffffffff -- (.exr 0xffffffffffffffff)
ExceptionAddress: 002f2aa7 (ping!mainCRTStartup)
ExceptionCode: c000001d (Illegal instruction)
ExceptionFlags: 00000000
NumberParameters: 0
...
CHKIMG_EXTENSION: !chkimg -lo 50 -d !ping
    002f2aa7-002f2aaa  4 bytes - ping!mainCRTStartup
        [ e8 05 03 00:f0 cc 90 90 ]
    4 errors : !ping (002f2aa7-002f2aaa)
    ...
    CONTEXT:  0015f358 -- (.cxr 0x15f358)
    eax=74e83398 ebx=7efde000 ecx=00000000 edx=002f2aa7
esi=00000000 edi=00000000
    eip=002f2aa7 esp=0015f7bc ebp=0015f7c4 iopl=0         nv
up ei pl zr na pe nc
    cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00010246
    ping!mainCRTStartup:
    002f2aa7 f0cc            lock int 3
    Resetting default scope
    LAST_CONTROL_TRANSFER:  from 74e833aa to
002f2aa7
    ...
    FAILED_INSTRUCTION_ADDRESS:
    ping!mainCRTStartup+0
    002f2aa7 f0cc            lock int 3
    ...
```

三、延伸思路

1) 用 sxe 将 second chance 变成 first chance

前面介绍的办法都是碰上 second chance 时不进一步破坏现场

的情况下如何快速定位 ExceptionAddress。如果所遭遇的异常可以稳定触发，并且每次现场都差不多，还可以用 sxe 将 second chance 变成 first chance，此时停下来的 EIP 就是引发异常的指令：

```
cdb.exe -hd -o ping.exe
(1270.1668): Break instruction exception - code 80000003 (first chance)
eax=00000000 ebx=00000000 ecx=2e820000 edx=0012dfe8 esi=fffffffe edi=00000000
eip=76f50fab esp=001df800 ebp=001df82c iopl=0 nv up ei pl zr na pe nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000246
ntdll!LdrpDoDebuggerBreak+0x2c:
76f50fab cc int 3
> g $exentry
eax=74e83398 ebx=7efde000 ecx=00000000 edx=00f52aa7 esi=00000000 edi=00000000
eip=00f52aa7 esp=001dfc84 ebp=001dfc8c iopl=0 nv up ei pl zr na pe nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000246
ping!mainCRTStartup:
00f52aa7 e805030000 call ping!__security_init_cookie (00f52db1)
> eb eip f0 cc 90 90
> u eip l 3
ping!mainCRTStartup:
00f52aa7 f0cc lock int 3
00f52aa9 90 nop
00f52aaa 90 nop
> sxe c000001d
> p
(1270.1668): Illegal instruction - code c000001d (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=74e83398 ebx=7efde000 ecx=00000000 edx=00f52aa7 esi=00000000 edi=00000000
eip=00f52aa7 esp=001dfc84 ebp=001dfc8c iopl=0 nv up ei pl zr na pe nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00010246
ping!mainCRTStartup:
00f52aa7 f0cc lock int 3
```

2) 在引发异常的线程栈上搜索 ContextFlags

有时候通过分析栈布局寻找 CONTEXT 太费劲了，我们知道它一定在栈上，只是不知道它在哪里，此时可以尝试暴力搜索 CONTEXT。

先来学习一下 CONTEXT 结构：

```
> dt ntdll!_CONTEXT poi(esp+c)
+0x000 ContextFlags : 0x1007f
// CONTEXT_ALL | CONTEXT_XSTATE
+0x004 Dr0 : 0
+0x008 Dr1 : 0
+0x00c Dr2 : 0
```

```
   +0x010 Dr3          : 0
   +0x014 Dr6          : 0
   +0x018 Dr7          : 0
   +0x01c FloatSave    : _
FLOATING_SAVE_AREA
   +0x08c SegGs        : 0x2b
   +0x090 SegFs        : 0x53
   +0x094 SegEs        : 0x2b
   +0x098 SegDs        : 0x2b
   +0x09c Edi          : 0
   +0x0a0 Esi          : 0
   +0x0a4 Ebx          : 0x7efde000 //
流程到达 $exentry 时，CONTEXT.Ebx ==
$PEB
   +0x0a8 Edx          : 0x2f2aa7  //
流程到达 $exentry 时，CONTEXT.Edx ==
Eip
   +0x0ac Ecx          : 0
   +0x0b0 Eax          : 0x74e83398
   +0x0b4 Ebp          : 0x15f7c4
   +0x0b8 Eip          : 0x2f2aa7
   +0x0bc SegCs        : 0x23
   +0x0c0 EFlags       : 0x10246
   +0x0c4 Esp          : 0x15f7bc
   +0x0c8 SegSs        : 0x2b
   ...
   > ? $PEB
   Evaluate expression: 2130567168 =
7efde000
```

CONTEXT 结构第一个成员 ContextFlags 的取值在 winnt.h 中定义：

```
   #define WOW64_CONTEXT_i386
0x00010000
   #define WOW64_CONTEXT_i486
0x00010000
   #define CONTEXT_CONTROL
(CONTEXT_i386 | 0x00000001L)
   #define CONTEXT_INTEGER
(CONTEXT_i386 | 0x00000002L)
   #define CONTEXT_SEGMENTS
(CONTEXT_i386 | 0x00000004L)
   #define CONTEXT_FLOATING_
POINT      (CONTEXT_i386 |
0x00000008L)
   #define CONTEXT_DEBUG_
REGISTERS      (CONTEXT_i386 |
0x00000010L)
   #define CONTEXT_EXTENDED_
REGISTERS  (CONTEXT_i386 |
0x00000020L)
   #define CONTEXT_FULL
(CONTEXT_CONTROL | ...
   #define CONTEXT_ALL
(CONTEXT_CONTROL | ...
   #define CONTEXT_XSTATE
(CONTEXT_i386 | 0x00000040L)
```

一般来说 ContextFlags 是常量，可以作为特征值。我们只想在栈区搜索特征值，可以通过 TEB 中的 StackLimit、StackBase 在定位栈区。

```
   > dt -b ntdll!_TEB NtTib.
     +0x000 NtTib  :
       +0x000 ExceptionList : Ptr32
       +0x004 StackBase : Ptr32
       +0x008 StackLimit : Ptr32
       +0x00c SubSystemTib : Ptr32
       +0x010 FiberData : Ptr32
       +0x010 Version : Uint4B
       +0x014 ArbitraryUserPointer :
```

Ptr32

+0x018 Self : Ptr32

下述命令在引发异常的线程栈上搜索 ContextFlags，以此定位 CONTEXT:

> ~#e s -d poi(@$TEB+8) poi(@$TEB+4) 1007f

0015f358 0001007f 00000000 00000000 00000000

.....

2005 年 Ivan Brugiolo 给了如下搜索方案：

> .foreach (CONTEXT {~#e s -[1]d poi(@$TEB+8) poi(@$TEB+4) 1007f}) {.echo CONTEXT;.cxr CONTEXT}

0x0015f358

eax=74e83398 ebx=7efde000 ecx=00000000 edx=002f2aa7 esi=00000000 edi=00000000

eip=002f2aa7 esp=0015f7bc ebp=0015f7c4 iopl=0 nv up ei pl zr na pe nc

cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00010246

ping!mainCRTStartup:

002f2aa7 f0cc lock int 3

1007f 是我当前环境的值，用上述命令的时候可能需要修正这个值，比如 1003f。

3) 在引发异常的线程栈上搜索 SegGs、SegFs、SegEs、SegDs

2009 年 Ken Johnson 给了另一种暴力搜索 CONTEXT 的方案，拿 SegGs、SegFs、SegEs、SegDs 作为特征值进行暴力搜索，定位 SegGs 之后间接定位 CONTEXT:

> .foreach (PSegGs {s -[w1]d 0 L? ffffffff @gs @fs @es @ds}) {? ${PSegGs}-8c;.cxr ${PSegGs}-8c}

L 后面有个？号，这是为了取消 windbg 默认的 256MB 范围上限。这个方案的误报会很多，要当心。稍好点的方案是在引发异常的线程栈上搜索：

> .foreach (PSegGs {~#e s -[1]d poi(@$TEB+8) poi(@$TEB+4) @gs @fs @es @ds}) {.cxr ${PSegGs}-8c}

eax=74e83398 ebx=7efde000 ecx=00000000 edx=002f2aa7 esi=00000000 edi=00000000

eip=002f2aa7 esp=0015f7bc ebp=0015f7c4 iopl=0 nv up ei pl zr na pe nc

cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00010246

ping!mainCRTStartup:

002f2aa7 f0cc lock int 3

四、参考文献

[1] Windows Debugging Help

[2] Debugger tricks: Find all probable CONTEXT records in a crash dump - [2009-03-30]

BOOTSCAN的设计与实现

核心技术部　董阳

关键字：Native Application　BOOTSCAN　反病毒扫描

摘要：本文讲解了传统杀毒软件最常用的反病毒技术之一，如何在开机启动时对系统进行文件扫描以及如何同用户进行交互。

一、引言

BOOTSCAN 是当前杀毒软件经常采用的一种技术，用于在系统启动时对病毒进行查杀。瑞星、江民、Avast 等杀毒软件都集成了 BOOTSCAN 功能。采用 BOOTSCAN 技术查杀病毒，是由于部分恶意程序启动时采用了一些恶意技术，使得程序常驻内存并且无法被正常清除，因此杀毒软件使用 BOOTSCAN 在病毒启动前对其进行查杀，实际上操作系统的开机磁盘扫描程序 AUTOCHK 也是采用了相同的技术。

二、什么是 Native Application

NT 系统被设计成为支持多个子系统，它可以执行在不同平台上的代码，包括但不限于 POSIX、OS/2 和 Win32 子系统。为了管理这些子系统，NT 内核输出了大量名为 Native API 的 API 函数，子系统服务将这些函数包装为它们自己的子系统接口，例如 CreateFile() 和 fopen() 都被映射到 NtCreateFile()。由于 Native Application 是早于子系统启动的，因此它不能够调用子系统中丰富的 API 接口（KERNEL32、GDI、USER32 等），而只能够调用 NTDLL 里的 Native API 和运行时函数，这也正是被称为 Native Application 的原因。

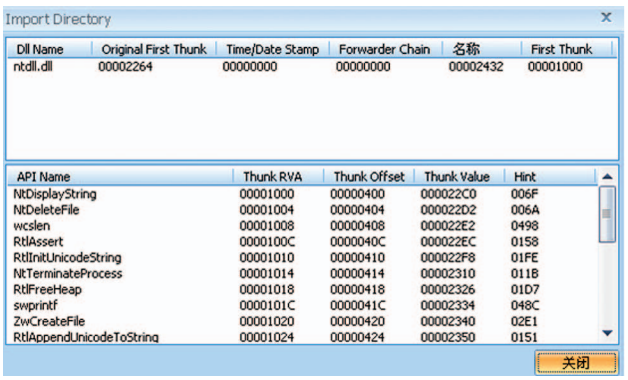

图 1 一个典型的 Native Application 的导入表

Native Application 应用程序的入口点是 NtProcessStartup，类似普通程序中的 WinMain 或 main。不同于其他的程序入口点的是，Native Application 采用了 STARTUP_ARGUMENT 结构体指针来作为参数输入，具体内容如下：

```
typedef struct
{
    ULONG    Unknown[3];
    PENVIRONMENT_INFORMATION Environment;
} STARTUP_ARGUMENT, *PSTARTUP_ARGUMENT;
```

其中 ENVIRONMENT_INFORMATION 的结构体内容为：

```
typedef struct
{
    ULONG          Unknown[21];
    UNICODE_STRING   CommandLine;
    UNICODE_STRING   ImageFile;
} ENVIRONMENT_INFORMATION, *PENVIRONMENT_INFORMATION;
```

该结构体里包含了 UNICODE_STRING 格式的命令行参数 CommandLine 和映像名 ImageFile。

由于只能使用 Native API 而不能使用任何子系统的 API，Native Application 必须使用 RtlCreateHeap() 函数来创建自己的堆来分配存储，使用 RtlAllocateHeap() 函数来分配内存以及用 RtlFreeHeap() 函数来释放内存。Native Application 需要使用 NtDisplayString() 函数才可以打印想要的内容到屏幕上（将被输出到初始化时的蓝色屏幕上）。Native Application 不像 WIN32 程序那样简单地从它们的启动函数返回，而需要程序自己调用 NtProcessTerminate() 函数来结束自身进程。其代码框架的基本结构如下：

```
void NtProcessStartup( PSTARTUP_ARGUMENT Argument )
{
        // do something else
        NtProcessTerminate( NtCurrentProcess(), 0 );
}
```

三、Native Application 的启动时机

理解 Native Application 的启动时机很重要，因为我们是开发启动查杀程序，因此启动的早晚会对我们的查杀能力有很大影响：启动过晚，会导致病毒程序先于我们的程序启动，从而使病毒得到优先的执行权，我们的查杀功能就会受到干扰。

当 Windows 内核的引导完成以后，就会启动会话管理器 smss.exe 进程了，其实 smss.exe 也是一个 Native Application，smss 主线程执行的步骤之一就是运行在 HKKM\SYSTEM\CurrentControlSet\Control\Session Manager\BootExecute 中定义的任何程序也就是 Native Application，执行完其他必须的初始化任务后，最后 smss 启动子系统进程，包括 csrss。

以上是一个对会话管理器 smss 执行过程简化的描述，详细步骤请参考《深入解析 Windows 操作系统》5.1 章节关于操作系统引导过程的说明。

根据上述过程可以看到，Native Application 是先于所有的子系统启动的，所以那个时候还没有任何子系统的程序被加载（包括服务），因此对于一般的应用层的恶意软件，基本都可以通过 BOOTSCAN 进

行查杀。

四、用户输入交互处理

之前网络上有些文章说，“无法简单地从用户那里接受到输入或显示些什么到屏幕上，因为那些接口都无法再使用（控制台 API 都是 Win32 API)”，实际上并非如此，Native Application 依然可以接受用户的键盘输入，只不过不像常规的应用层程序那么简单到只需要调用一个 scanf() 等函数就可以接受用户的键盘输入，而是需要我们自己枚举系统内的键盘，并打开键盘类驱动（KBDCLASS）创建的设备，利用 NtReadFile() 函数来读取键盘缓冲区，从而实现键盘的输入功能。

```
for (; i < MAX_KEYBOARD_COUNT; i++)
{
    RtlZeroMemory(szwKbdDeviceName, sizeof(szwKbdDeviceName));
    swprintf(szwKbdDeviceName, L"\\Device\\KeyboardClass%d", i);
    RtlInitUnicodeString(&usDeviceName,szwKbdDeviceName);
    InitializeObjectAttributes(&ObjectAttributes,&usDeviceName,
        OBJ_CASE_INSENSITIVE ,NULL,NULL);
    ntStatus = NtCreateFile(&KeyboardHandle, \
        0x80100080, &ObjectAttributes, &pIoStatusBlocks[i], NULL, FILE_ATTRIBUTE_NORMAL, 0, 1, 1, NULL ,0);
...
}
```

上面的代码很简单，就是枚举 \Device 目录下由键盘类驱动创建的设备对象 KeyboardClassX，X 表示数字。MAX_KEYBOARD_COUNT 是我们定义的键盘个数，我们自己实现的程序把这个宏定义为 128，也就是最多同时支持 128 个键盘。枚举到键盘类驱动创建的设备对象后，就利用 NtCreateFile 来打开设备，并创建对应的同步事件对象：

```
NtCreateEvent(&KeyboardEventList[NumOfKeyboards],
    EVENT_ALL_ACCESS, &ObjectAttributes, SynchronizationEvent, FALSE);
```

这个事件对象的作用就是，当接收到键盘数据后，底层驱动会利用该事件来通知上层（键盘类驱动），这样我们得到键盘事件通知，就可以利用 NtReadFile() 函数来读取键盘输入的数据了。

接着我们在读取键盘内容之前先清空键盘的缓存：

```
do
{
    for (i = 0; i<ulKbdNum; i++)
    {
        ntStatus = NtReadFile(hKbdTable[i], hReadKbdEvt[i], NULL,
            NULL,
            &pIoStatusBlocks[i],
            &pKeyboardInputData[i],
            sizeof(KEYBOARD_INPUT_DATA),
```

```
                &ByteOffset, NULL);
}
}while (NT_SUCCESS(ntStatus));
```

然后使用 NtReadFile() 函数循环读取键盘输入：

```
while(TRUE)
{
    for (i = 0; i < ulKbdNum; i++)
    {
        RtlZeroMemory(&pKeyboardInputData[i],
sizeof(KEYBOARD_INPUT_DATA));
        ntStatus = NtReadFile(hKbdTable[i], hReadKbdEvt[i],
NULL, NULL,
&ploStatusBlocks[i],
                              &pKeyboardInputData[i],
                              sizeof(KEYBOARD_INPUT_
DATA), &ByteOffset, NULL);
    }
}
NtWaitForMultipleObjects(ulKbdNum,hReadKbdEvt,WaitAny,FA
LSE, NULL);
    ..... 省略部分代码
```

每次读入键盘数据前，先把结构清空，然后通过 NtReadFile() 函数来读取键盘内容。由于是异步操作，因此我们调用 NtWaitForMultipleObjects() 函数来等待键盘消息读取完成。当有任意一个键盘得到输入后，就立刻从 NtWaitForMultipleObjects() 函数返回，并开始处理键盘数据。利用 NtReadFile 读取到的键盘内容并不是我们在应用层看到的键盘内容，而是读入了一个结构体，里面包含了键盘的扫描码和标志等信息，我们需要自己将键盘的扫描码转换成对应的键盘数据。其结构体如下：

```
typedef struct _KEYBOARD_INPUT_DATA {
    USHORT UnitId;
USHORT MakeCode;
USHORT Flags;
  USHORT Reserved;
  ULONG ExtraInformation;
} KEYBOARD_INPUT_DATA, *PKEYBOARD_INPUT_DATA;
```

在上面的结构体中包含了多个字段，但我们比较关心的是前两个即 UnitId 和 MakeCode。其中 UnitId 表示单元号码，表示当前数据对应的是哪一个键盘，比如 \Device\KeyboardPort0 的 UnitId 就是 '0'，\Device\KeyboardPort1 的 UnitId 就是 '1'，以此类推。第二个字段是 MakeCode，我们这里简单说明一下键盘的扫描码。当键盘上有键被按下、松开、按住，键盘将产生扫描码 (Scan Code)，这些扫描码将被 i8048 键盘控制器直接得到。扫描码有两种，Make Code 和 Break Code。当一个键被按下或按住时产生的是 Make Code，当一个键被松开产生的是 Break Code。每个键被分配了惟一的 Make Code 和 Break Code，这

样主机通过扫描码就可以知道是哪一个键。简单的说就是按下键，产生一个 Make Code，松开键，产生一个 Break Code。到目前为止一共有三套扫描码集（Scan Code Set），ps/2 键盘默认使用第二套。不过可以设置 i8042，让 i8042 把得到的 Scan Code 翻译成 Scan Code Set 1 中的 Scan Code，这样键盘驱动从 i8042 得到的所有 Scan Code 都是第一套中的 Scan Code（实际中驱动也是这么做的）。所以我们只讨论 Scan Code Set 1。需要说明的是，Scan Code 和 ASCII 码完全不相同。这样，如果我们只处理简单的键盘输入，那么就可以直接将键盘的扫描码映射为键盘对应的实际字符，这样做一张表就可以了。

```
UCHAR ScanCodeTable[] = {
0,
0, //ESC
'1','2','3','4','5','6','7','8','9','0',0,0,0,0,'Q',
'W','E','R','T','Y','U',    'I','O','P',0,0,
0, //ENTER
0,'A','S','D','F','G','H','J','K','L',0,0,0,0,0,'
Z','X','C','V','B','N','M'
};
```

使用这个数组，将获取到的 KEYBOARD_INPUT_DATA 里的 MakeCode 字段作为索引，就可以得到对应的键盘字符。

经过以上过程就可以简单地获取键盘的输入了。对于键盘驱动更详细的信息可以参考 JIURL 的《JURL 键盘驱动》1,2,3 系列文章。

五、磁盘文件扫描

我们可以利用 NtQueryDirectoryFile() 函数的子参数 FileBothDirectoryInformation 来获取目录和文件的信息，然后利用递归或者迭代算法对目录进行完整的遍历，即可实现对磁盘文件的扫描。当枚举到文件后，就可以调用反病毒引擎的接口对其内容进行检查。

六、如何构建

我们这里采用 WDK 来构建 Native Application，只需要一个简单的 SOURCES 放在工程目录下，即可使用 WDK 的 BUILD 工具进行编译构建。SOURCES 文件的内容如下：

```
TARGETNAME=native
TARGETPATH=OBJ
TARGETTYPE=PROGRAM
INCLUDES=$(DDK_INC_PATH);D:\
WINDDK\2600\inc\ddk\wdm\wxp
SOURCES=native.c
```

同一般驱动编译不同的是，需要将 TARGETTYPE 设置成 PROGRAM，其他基本相同。

七、如何安装

将文件复制到 system32 目录下，后再将程序的文件名和参数写到注册表路径 HKEY_LOCAL_MACHINE\SYSTEM\Current-ControlSet\Control\Session Manager 的 BootExecute 键值中，该键值的类型是 MULTI_SZ，也就是说可以包含多个 Native Application。每一行的格式为：

程序名 参数

会话管理器在 system32 目录下查找该值列出的可执行程序并执行。

编辑完注册表后重新启动计算机，Native Application 即可得到执行。

图 2 Native Application 所在启动项及内容

八、实现效果

图 3 展示了我们编写的演示程序的实际执行效果。

图 3 实现效果

参考文献

1.《Native Application 开发详解》http://www.cnblogs.com/BoyXiao/archive/2011/09/21/2183059.html

2.《Inside Native Applications》Mark E.Russinovich

3.《深入解析 WINDOWS 操作系统》第四版 Mark E.Russinovich, David A.Solomon

4.《JIURL 键盘驱动》JIURL

绿盟科技DDoS威胁态势报告

研究院　鲍旭华　洪海

执行摘要

多年来，绿盟科技致力于帮助客户实现业务的安全顺畅运行。每天，我们的防护产品和监测系统会发现数以千计的 DDoS（分布式拒绝服务）攻击在危害客户安全。为了向用户提供更多关于这类攻击的信息，绿盟科技威胁响应中心特别发布此报告。

2013 上半年，DDoS 攻击备受瞩目。黑客组织伊兹丁 • 哈桑网络战士向美国发起的挑战依然持续，许多知名银行因受到攻击而服务中断。反垃圾邮件组织 Spamhaus 受到的攻击，被认为是史上规模最大的 DDoS，流量达到了惊人的 300G。面对汹涌而来的数据洪流，没有谁敢说自己的防御系统万无一失。

大企业和大机构总是成为新闻头条。然而中小企业面对 DDoS 威胁时的境遇同样惨烈。上半年，九成以上的攻击发生在半小时之内，八成以上流量小于 50Mbps，三分之二的受害者遭受过不止一次攻击。攻击者反复采用时间短且流量小的 DDoS，可能只是因为这样可以用较小的成本来达成目的。

本报告从 DDoS 的概述、目标和方法角度对这一威胁进行了描述。其中的数据源自 90 次重大新闻报道和 168459 次攻击事件。所有数据在进行分析前都已经过匿名化处理，不会在中间环节出现泄露，任何与客户有关的具体信息均不会出现在报告中。

正文目录

图目录

DDoS 的攻击概述

2013 上半年 DDoS 事件和攻击频发。平均每两天，就会有一次重大 DDoS 事件被报道；而每两分钟，绿盟科技就会监测到一次 DDoS 攻击。攻击和重大事件在 4、5 月各自达到峰值。黑客行动主义是重大 DDoS 事件发生的主要原因，其次是商业犯罪和网络战。

观点 1：平均每两天发生一起重大 DDoS 攻击事件，每两分钟发生一次普通 DDoS 攻击事件

绿盟科技在 2013 上半年跟踪了新闻媒体报道的 90 次重大 DDoS 事件，平均每两天一次。同时，我们共监测到 168459 次 DDoS 攻击，平均每两分钟发生 1.29 次。其中重大事件在五月最多，而 DDoS 攻击则在四月达到峰值。

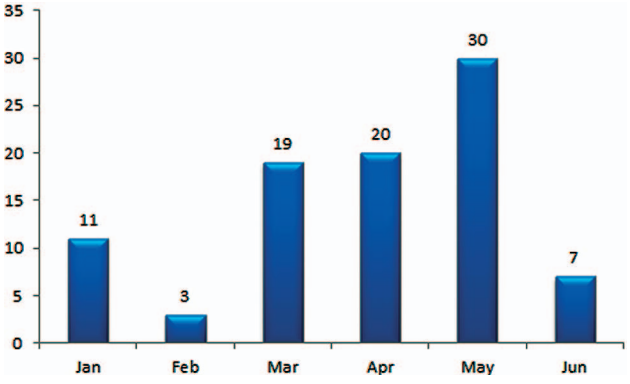

图 1 2013 上半年重大 DDoS 事件

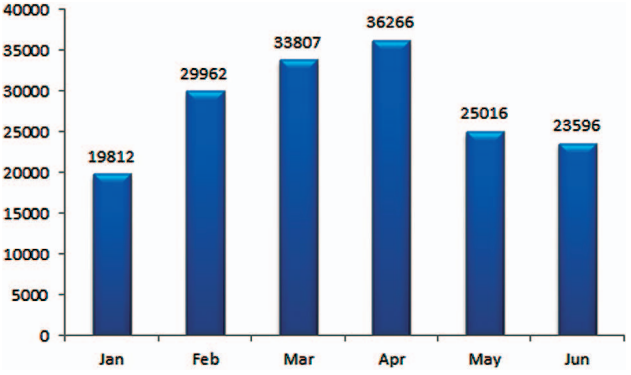

图 2 2013 上半年 DDoS 攻击

观点 2：黑客行动主义是重大 DDoS 事件发生的最主要原因，其次是商业犯罪和网络战。

从绿盟科技跟踪的 90 次重大 DDoS 事件报道来看，黑客行动主义是发生重大 DDoS 事件的最主要原因，其次是商业犯罪和网络战。

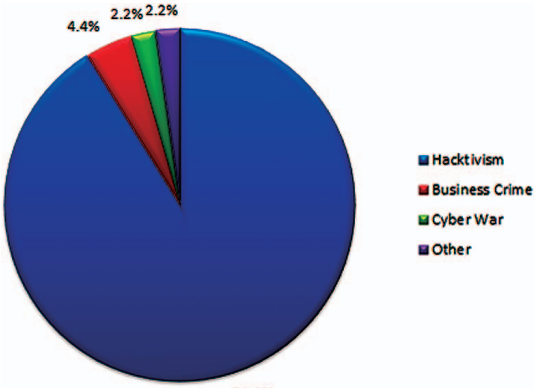

图 3 发生重大 DDoS 事件的原因

DDoS 的攻击对象

2013 上半年，DDoS 是安全界的焦点之一，主要原因是 Qassam 组织开展的“燕子行动”，美国银行业成为了这次行动的最大受害者。最受关注的事件中，被攻击者还包括一些政府和商业公司。此外，在绿盟科技监测的一般性攻击中，九成的对象位于中国。而三分之二的受害者遭遇了多次攻击。

事件 1：“燕子行动”(Operation Ababil)

2012 年 7 月，一个由美国人萨姆 - 巴西利 (Sam Bacile) 制作并导演的关于伊斯兰教先知穆罕默德的影片的预告片被放到 YouTube 上，引来了穆斯林世界的强烈抗议。同年 9 月 18 号，一群号称伊兹丁 • 哈桑网络战士（Cyber fighters of Izz ad-din Al-Qassam）在 pastebin 网站上发布公告，声明称其将美国银行和纽约交易所列为攻击目标，在 YouTube 上这部亵渎穆斯林先知的影片被移除之前，攻击将一直持续。从此，代号为“燕子行动”（Operation Ababil）的一系列针对美国金融机构的 DDoS 攻击事件拉开了序幕。“燕子行动”(Operation Ababil) 这个代号引用自《可兰经》里的安拉派燕群去摧毁一队由也门国王派出攻击麦加的象群的故事。

到 2013 年 6 月为止，整个行动经历了三个阶段：第一阶段始于 2012 年 9 月 18 号，持续了 5 个星期；第二阶段从 2012 年 12 月 10 号开始，持续 7 个星期；第三阶段从 2013 年 3 月 5 日开始，持续 9 个星期，到 5 月 6 日停止。其中 2013 年上半年的行动如图 4 所示。2013 年 7 月 23 日，第四阶段也已开始。

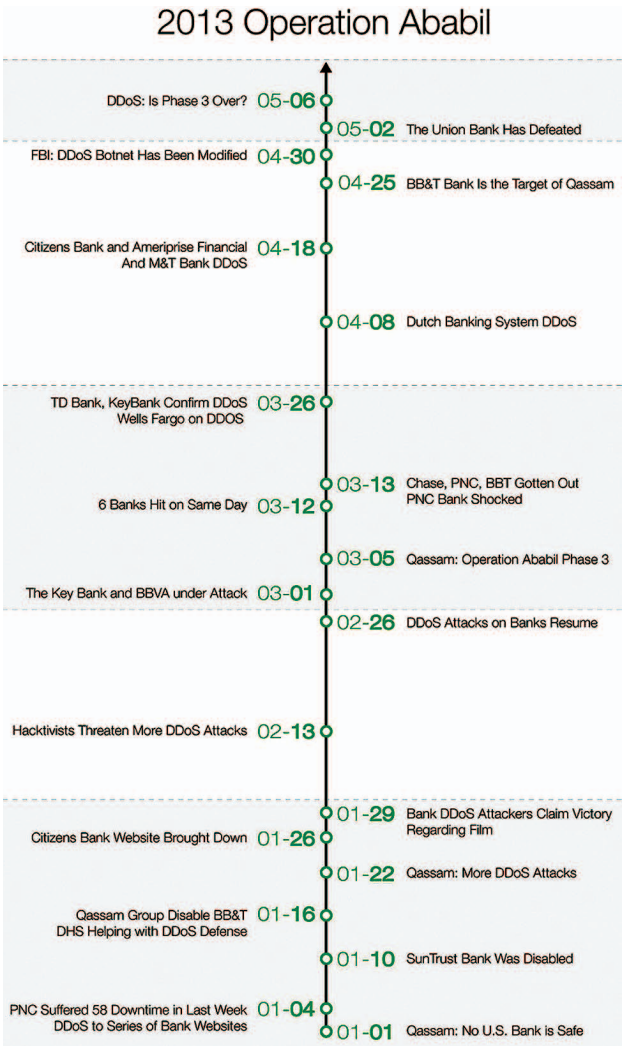

图 4 燕子行动时间线

在整个行动中，大多数美国金融机构的在线银行业务都遭受到了攻击，其中包括美国银行 (Bank of America)、花旗集团 (Citigroup)、富国银行 (Wells Fargo)、美国合众银行 (US Bancorp)、PNC 金融服务集团、第一资本 (Capital One)、五三银行 (Fifth Third Bank)、BB&T 银行和汇丰银行 (HSBC)。DDoS 攻击对上述银行网站业务的连续性和可获得性造成了严重的影响，同时也对银行的声誉造成了不可估量的损失。由于事态的严重性，美国政府部门，包括国土安全部 (DHS)、联邦调查局（FBI）以及金融监管机构，均参与了事件的调查和处理。

观点 3：**银行、政府和商业公司是 DDoS 攻击的最大受害者**

绿盟科技跟踪了上半年世界上较为重大的 90 次 DDoS 攻击事件，其中针对银行的事件共计 39 次，占 43%。造成这种现象的主要原因是黑客组织 Cyber fighters of Izz ad-din Al-Qassam 在从去年开始的“燕子行动”中，对美国银行业发动了持续数月的 DDoS 攻击。此外，针对政府 DDoS 攻击发生 26 次，占 29%；而针对商业企业的攻击发生 19 次，占 21%。而非政府组织和网络服务提供商也是攻击目标。

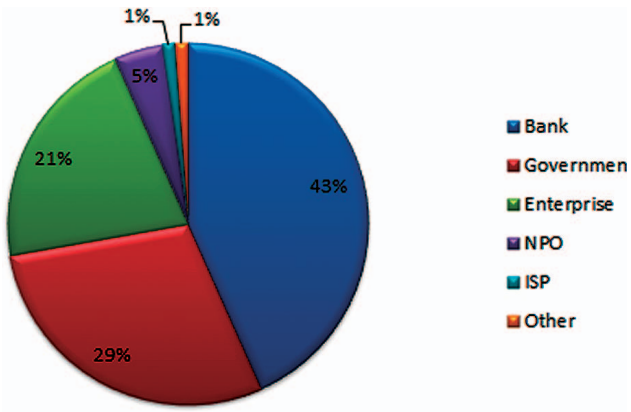

图 5 DDoS 攻击目标的行业分布

观点 4：**中国依然是 DDoS 攻击的主要受害者，其次是美国、韩国等地**

中国依然是 DDoS 攻击的主要受害者，占被攻击总数的 92.3%。其次是美国，占 5.8%。这一结果可能带有地域性，这是由于当前绿盟科技的大部分监测和防护业务在亚太地区，所以得到的数据也以这一区域为主。

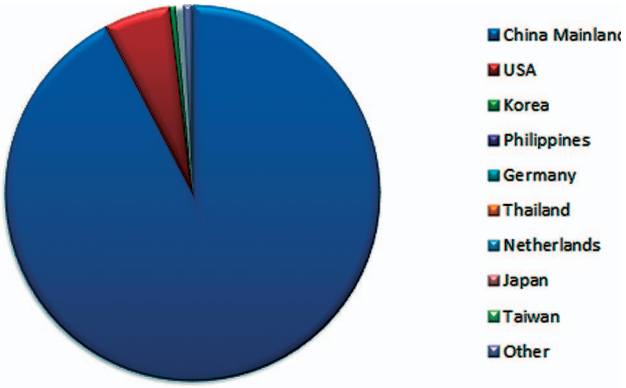

图 6 DDoS 攻击目标的地理分布

观点 5：**三分之二的受害者遭遇多次攻击，6% 在 10 次以上**

攻击者往往会对同一目标发起多次攻击。2013上半年，31.3%的受害者遭受了一次DDoS攻击，而6.1%甚至遇到了10次以上。在2012全年内，有50.7%的受害者遭受了一次DDoS攻击，5.2%遇到了10次以上。也就是说，去年只有一半受害者遭到了多次攻击，而今年上半年这一比例则达到了三分之二。很明显，攻击者现在更倾向于多次攻击同一目标，在2013下半年这一趋势还可能加强。产生这种现象的原因可能有两种：第一，攻击者进行DDoS攻击也需要考虑成本，例如租用僵尸网络，短期多次的攻击可以耗费较少的资金，而使效果看起来比较大；第二，一些没有防御能力的网站在受到攻击时不得不支付勒索金，消息传出后，自然成为其他攻击者的优先目标。

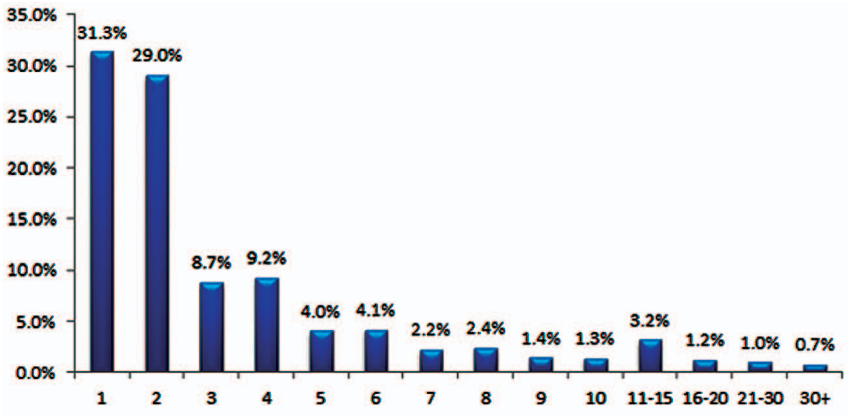

图7 DDoS的攻击频次

DDoS的方法

2013上半年，DDoS攻击所采用的方式正在发生分化。一方面，攻击者不断追求更大的攻击流量。三月Spamhaus攻击事件被认为是史上最大规模的DDoS攻击，达到了惊人的300G攻击流量。另一方面，消耗应用资源的DDoS攻击大行其道，以HTTP FLOOD为代表，被攻击者广泛采用。这类攻击产生的流量和包速率并不大，破坏作用却同样显著。此外，混合攻击所占比重也越来越大，ICMP+TCP+UDP FLOOD成为最常见的组合。

事件2：史上最大规模的DDoS攻击

Spamhaus是一家致力于反垃圾邮件的非盈利组织，总部在伦敦和日内瓦。Spamhaus维护了一个巨大的垃圾邮件黑名单，这个黑名单被很多大学/研究机构、互联网提供商、军事机构和商业公司广泛使用。

从2013年3月18日起，Spamhaus开始遭受DDoS攻击。攻击者通过僵尸网络和DNS反射技术进行攻击，攻击流量从10G不断增长，在3月27日达到惊人的300G攻击流量，被认为是互联网史上最大规模的DDoS攻击事件。

攻击者使用的主要攻击技术是DNS反射技术。从2012年开始，DNS反射技术

已经成为大规模第三层 DDoS 攻击的主要部分。DNS 反射攻击技术的基本方式是：向大量开放 DNS 解析器发送带有扩展字段 OPT RR（伪资源记录）的 DNS 查询请求，并将该 DNS 请求的源 IP 地址伪造成想要攻击的目标 IP 地址。开放 DNS 服务器在接收到请求后会对该请求进行解析查询，并将大范围域名查询的响应数据发送给攻击目标。由于请求数据比响应数据小得多，攻击者就可以利用该技术有效地放大其掌握的带宽资源和攻击流量。

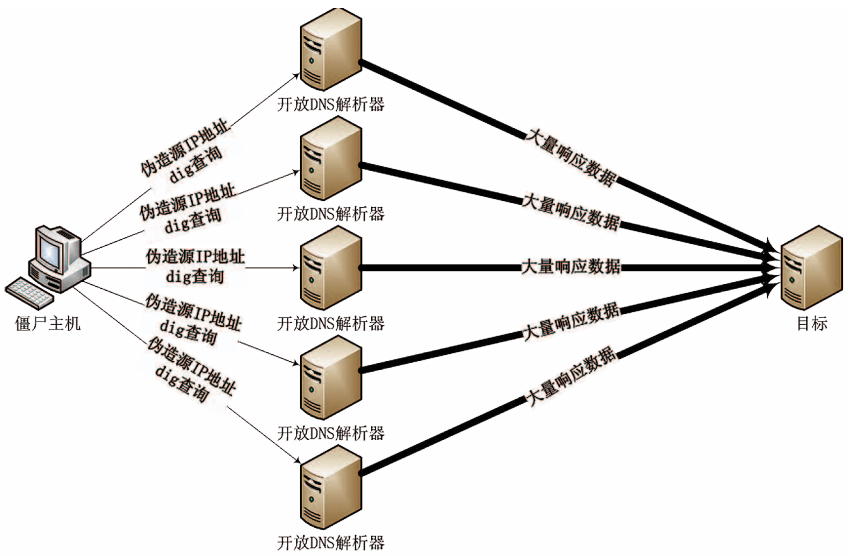

图 8 DNS 反射攻击

在本次事件中，攻击者向三万多个开放 DNS 服务器发送了对 ripe.net 域名的解析请求，并将源 IP 地址伪造成 Spamhaus 的 IP 地址，大量开放 DNS 服务器的响应数据产生了大约 300G 的攻击流量。DNS 请求数据的长度约为 36 字节，而响应数据的长度约为 3000 字节，这意味着利用 DNS 反射能够产生约 100 倍的放大效应，因此，攻击者只需要掌握和控制一个能够产生 3G 流量的僵尸网络，就能够进行这么大规模（300G）的攻击。除了 DNS 反射技术，攻击者还使用了 ACK 反射等其他技术进行攻击。

2013 年 7 月 25 日，互联网系统协会（ISC, Internet Systems Consortium）宣布，为了防御利用 DNS 发起的反射式 DDoS 攻击，最新版的 BIND 软件增加了响应速率限制（RRL，Response Rate Limiting）模块，并声称这会是缓解 DNS 反射攻击的最有效方法。

观点 6：TCP FLOOD 和 HTTP FLOOD 是最主要的 DDoS 攻击方式，两者占总数的四分之三

2013 上半年，我们共发现了 168459 次 DDoS 攻击，增长既来自攻击数量的增加，也由于观测范围的变大。其中，TCP FLOOD 占 38.7%，重新占据排行榜的首位；而 HTTP FLOOD 占总数的 37.2%，与

2012 年相比下降 5.5%；DNS FLOOD 攻击则占 13.1%，依然是一种重要的攻击形式。这半年中，我们观测到 4.1% 的 DDoS 采用了混合攻击的方式，成为一种新的普遍现象，所以我们将其单独列为一类。

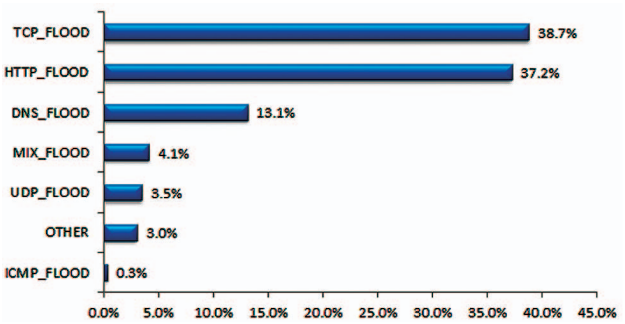

图 9 DDoS 攻击类型

观点 7：九成以上的 DDoS 攻击发生在半小时内，1.5% 的攻击会持续一天以上

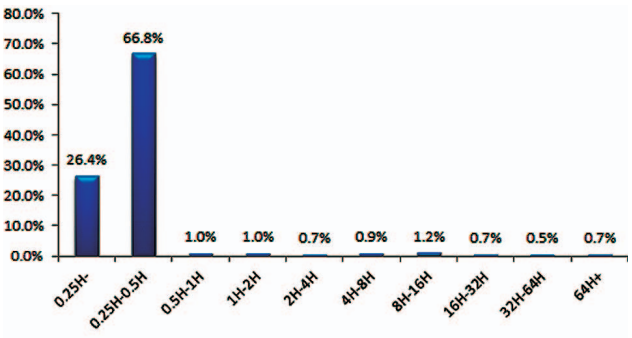

图 10 DDoS 的攻击持续时间

大部分 DDoS 攻击的持续时间并不长。三十分钟之内的攻击占到了 93.1%，与 2013 年的 93.2% 基本持平。一个可能的猜测是攻击者的尝试性攻击在逐步减少。

观点 8：峰值流量 50Mbps 以下的攻击占八成，包速率 0.2Mpps 以下的攻击占七成

在监测到的 DDoS 攻击中，流量 50Mbps 以下的占 80.1%，而 2G 以上的攻击只占 0.9%。近年来 HTTP FLOOD 攻击被攻击者大量采用，流量不再与攻击效果成正比，通过消耗应用资源而实现拒绝服务显然更有效率。

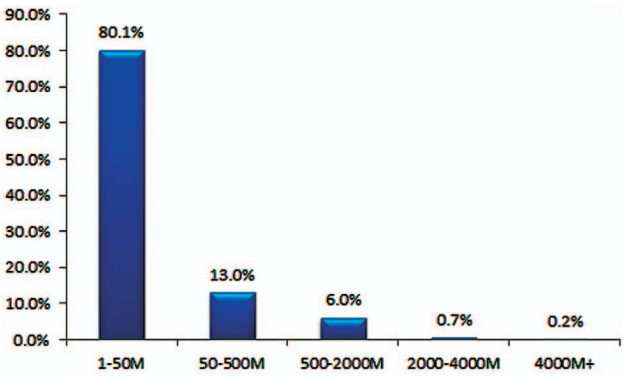

图 11 DDoS 攻击流量分布（bps）

此外，从每秒的峰值包速来看，69.1% 攻击在 0.2M 以下。与流量的变化相比，包速率的范围更为集中，仅有 0.2% 大于 3.2Mpps。

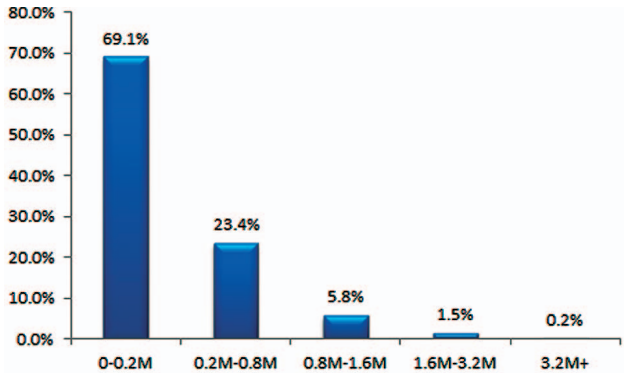

图 12 DDoS 攻击的包速率分布（pps）

观点 9：混合攻击所占比重逐步增加，其中一半是 ICMP+TCP+UDP FLOOD 的组合

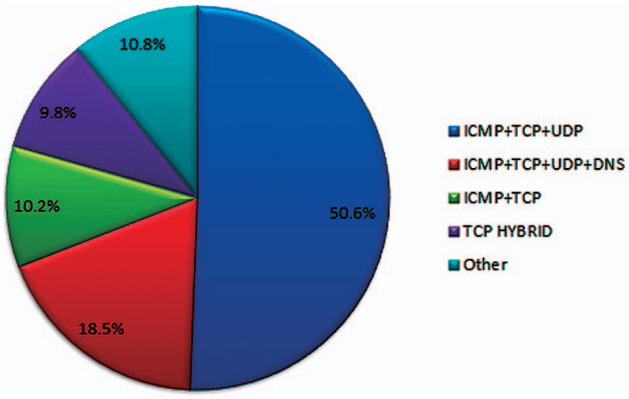

图 13 DDoS 混合攻击

2013 上半年绿盟科技监测到的 DDoS 混合攻击共计 6956 次，占攻击总数的 4.1%。对于其中的 6427 次，我们分析了其组成，并按照攻击使用的协议类型进行了分类。在这些混合攻击中，ICMP+TCP+UDP 的方式最为普遍，占总数的 50.6%；其次是前者再加入 DNS 的组合，占 18.5%；使用不同种类的 TCP FLOOD 进行组合攻击（例如 SYN FLOOD 和 ACK FLOOD）的占 9.8%。

结束语

同一现象，从不同角度看会有不同的理解。除本文中给出的观点之外，我们可以做一些可能合理的猜测。DDoS 攻击的数量在短期内不断起伏，长期来则处于持续增长的状态。媒体关注的事件往往是网络战和黑客行动主义，但更多的普通攻击单纯出于商业竞争或恶意勒索的目的。攻击者渐渐分为两类，一类专注于吸引眼球，一类则专注于赚钱，后者会越来越多地考虑“黑客经济学”的问题。受害者会发现，攻击给他们带来的单位时间损失，正变得越来越大。消耗应用资源的 DDoS 就符合这一目的，所以 HTTP FLOOD 大行其道。但传统的 TCP FLOOD 和 UDP FLOOD 也不会消失，当流量足够大时，任何信息系统都会难以应对。

这场战争，您准备好了吗?

作者和贡献者

作者　鲍旭华　洪海

贡献者　刘亚　曹志华　周志彬　王洋

NSFOCUS 2013年7月之十大安全漏洞

声明：本十大安全漏洞由 NSFOCUS(绿盟科技) 安全小组 <security@nsfocus.com> 根据安全漏洞的严重程度、利用难易程度、影响范围等因素综合评出，仅供参考。

http://www.nsfocus.net/index.php?act=sec_bug&do=top_ten

1. 2013-07-17 Apache Struts 多个前缀参数远程命令执行漏洞 (CVE-2013-2251)

NSFOCUS ID: 24131

http://www.nsfocus.net/vulndb/24131

综述：

Struts2 是第二代基于 MVC 模型的 Java 企业级 Web 应用框架。

Apache Struts2 在实现其功能的过程中使用了 Ognl 表达式，并将用户通过 URL 提交的内容拼接入 Ognl 表达式中，攻击者可以通过构造恶意 URL 来执行任意 Java 代码。

危害：

攻击者可以通过向服务器发送恶意请求来利用此漏洞，从而控制服务器系统。

2. 2013-07-05 Android 绕过数字签名可修改代码漏洞

NSFOCUS ID: 24038

http://www.nsfocus.net/vulndb/24038

综述：

Android 是 Google 公司公布的嵌入式操作系统。

Android 安全模块存在安全漏洞，可使攻击者修改 APK 代码，将合法应用转变为恶意木马，而不破坏应用的签名。

危害：

攻击者可以利用此漏洞绕过安卓签名，在 APK 中植入恶意代码，从而控制受害者系统。

3. 2013-07-10 Microsoft Internet Explorer 内存破坏漏洞 (CVE-2013-3150)（MS13-055)

NSFOCUS ID: 24084

http://www.nsfocus.net/vulndb/24084

综述:

Windows Internet Explorer，简称 MSIE，是微软公司推出的一款网页浏览器。

Microsoft Internet Explorer 7 / 8 / 9 / 10 不正确地访问内存中的对象时，存在远程执行代码漏洞。

危害:

攻击者可以通过诱使受害者打开恶意网页来利用此漏洞，从而控制受害者系统。

4. 2013-07-11 Adobe Flash Player 多个远程代码执行漏洞 (APSB13-17)

NSFOCUS ID: 24094

http://www.nsfocus.net/vulndb/24094

综述:

Adobe Flash Player 是一个集成的多媒体播放器。

Adobe Flash Player 在实现上存在多个远程代码执行漏洞，攻击者可利用这些漏洞在受影响应用上下文中执行任意代码，破坏内存，执行拒绝服务攻击等。

危害:

攻击者可以通过诱使受害者打开恶意 SWF 文件来利用此漏洞，从而控制受害者系统。

5. 2013-07-26 SIM 卡弱加密及机密信息泄露漏洞

NSFOCUS ID: 24189

http://www.nsfocus.net/vulndb/24189

综述:

SIM 卡也称用户身份识别卡，GSM 数字移动电话机必须装上此卡方能使用。

SIM 卡内的数据及应用可以通过 OTA 技术进行远程管理，推送 Java 软件到 SIM 卡上。部分 SIM 卡设计实现上存在问题，使用过时的 70 年代的 DES 加密方式进行数据和应用的签名。

危害:

攻击者可以破解 SIM 卡密钥，推送恶意应用到 SIM 卡安装执行，获取更多机密信息，从而复制用户的 SIM 卡。

6. 2013-07-30 ISC BIND 9 DNS RDATA 处理远程拒绝服务漏洞 (CVE-2013-4854)

NSFOCUS ID: 24218

http://www.nsfocus.net/vulndb/24218

综述:

BIND 是一个应用非常广泛的 DNS 协议的实现。

ISC BIND 9.8.0-9.8.5-P1、9.9.0-9.9.3-P1 在解析 DNS 查询内的 RDATA 时出错。

危害:

远程攻击者通过特制的查询利用此漏洞可触发 REQUIRE 断言，

使服务器崩溃。

7. 2013-07-22 Cisco IOS GET VPN Encryption Policy 安全绕过漏洞 (CVE-2013-3436)

NSFOCUS ID: 24154

http://www.nsfocus.net/vulndb/24154

综述：

Cisco IOS 是多数思科系统路由器和网络交换机上使用的互联网络操作系统。

Cisco IOS 的 Group Encrypted Transport VPN (GET VPN) 功能，在默认配置中使用了不正确的机制启用 GDOI 网络流量。

危害：

远程攻击者通过 UDP 端口 848 利用此漏洞，可绕过加密策略。

8. 2013-07-30 GE Proficy CIMPLICITY 'CimWebServer' 远程栈缓冲区溢出漏洞

NSFOCUS ID: 24212

http://www.nsfocus.net/vulndb/24212

综述：

GE Proficy CIMPLICITY 是客户端 / 服务器业务可视化和控制解决方案。

GE Proficy CIMPLICITY 的 CimWebServer 组件存在远程代码执行漏洞。

危害：

攻击者可以通过向服务器发送恶意请求来利用此漏洞，从而控制服务器系统。

9. 2013-07-17 Oracle MySQL Server 远程安全漏洞 (CVE-2013-3783)

NSFOCUS ID: 24130

http://www.nsfocus.net/vulndb/24130

综述：

Oracle MySQL Server 是一个轻量的关系型数据库系统。

Oracle MySQL Server 5.5.31 之前版本存在远程安全漏洞，此漏洞可通过 MySQL 协议利用，Server Parser 子组件受到影响。

危害：

攻击者可以通过向服务器发送恶意请求来利用此漏洞，从而控制服务器系统。

10. 2013-07-19 Autodesk 多个产品 DWG 处理代码执行漏洞

NSFOCUS ID: 24146

http://www.nsfocus.net/vulndb/24146

综述：

Autodesk 是二维和三维设计、工程及娱乐软件厂商，旗下拥有拥有 “AutoCAD”、“3DS Max” 等知名软件。

多个 Autodesk 产品在处理 DWG 文件时存在安全漏洞，可被恶意用户利用执行任意代码，从而破坏用户系统。

危害：

攻击者可以通过诱使受害者打开恶意 DWG 文件来利用此漏洞，从而控制受害者系统。

NSFOCUS 2013年8月之十大安全漏洞

声明：本十大安全漏洞由 NSFOCUS(绿盟科技) 安全小组 <security@nsfocus.com> 根据安全漏洞的严重程度、利用难易程度、影响范围等因素综合评出，仅供参考。

http://www.nsfocus.net/index.php?act=sec_bug&do=top_ten

1. 2013-08-30 Apple iOS6 特殊阿拉伯字符拒绝服务漏洞

NSFOCUS ID: 24544

http://www.nsfocus.net/vulndb/24544

综述：

Apple iOS 是由苹果公司开发的手持设备操作系统。

Apple iOS6 版本在处理包含某些特殊阿拉伯字符的邮件、短信或是微博时，就会造成应用程序闪退。

危害：

远程攻击者可以通过向受害者发送恶意消息来利用此漏洞，从而控制受害者系统。

2. 2013-08-14 Microsoft Internet Explorer 内存破坏漏洞 (CVE-2013-3199)(MS13-059)

NSFOCUS ID: 24373

http://www.nsfocus.net/vulndb/24373

综述：

Windows Internet Explorer，简称 MSIE，是微软公司推出的一款网页浏览器。

Internet Explorer 访问内存对象的方式时存在远程代码执行漏洞。

危害：

远程攻击者可以通过诱使受害者打开恶意网页来利用此漏洞，从而控制受害者系统。

3. 2013-08-23 Adobe Reader 任意代码执行漏洞

NSFOCUS ID: 24465

http://www.nsfocus.net/vulndb/24465

综述：

Adobe Reader(也被称为 Acrobat Reader) 是美国 Adobe 公司开发的一款优秀的 PDF 文档阅读软件。

Adobe Reader 11.0.03 在处理用户提供的输入时存在不明漏洞。

危害：

攻击者可以通过诱使受害者打开恶意 PDF 文件来利用此漏洞，从而控制受害者系统。

4. 2013-08-15 ISC BIND 9 SRTT 算法授权服务器选择安全漏洞

NSFOCUS ID: 24383

http://www.nsfocus.net/vulndb/24383

综述：

BIND 是一个应用非常广泛的 DNS 协议的实现。

ISC BIND 9 内的 SRTT 算法实现中存在漏洞，攻击者可以手动降低递归服务器与授权服务器相关联的 SRTT 值，从而影响特定授权服务器从 NS 资源记录集值内确定要查询的域服务器。

危害：

远程攻击者可以通过向服务器发送恶意请求来利用此漏洞，进行 DNS 缓存投毒。

5. 2013-08-21 Google Chrome 29.0.1547.57 之前版本多个安全漏洞

NSFOCUS ID: 24441

http://www.nsfocus.net/vulndb/24441

综述：

Google Chrome 是由 Google 开发的一款设计简单、高效的 Web 浏览工具。

Chrome 29.0.1547.57 之前版本存在多个安全漏洞，包括执行任意代码、泄露敏感信息等。

危害：

远程攻击者可以通过诱使受害者打开恶意网页来利用此漏洞，从而控制受害者系统。

6. 2013-08-20 Oracle Java BytePackedRaster.verify() 签名整数溢出

NSFOCUS ID: 24429

http://www.nsfocus.net/vulndb/24429

综述：

Java 是网络化应用程序的基础，也是开发和提供移动应用程序、游戏、基于 Web 的内容和企业软件的全球化标准。

Oracle Java 7u25 之前版本内的 BytePackedRaster.verify() 方法存在签名整数溢出漏洞，可允许绕过 "dataBitOffset" 边界检查。

危害：

攻击者可以向 Java 应用提交恶意数据来利用此漏洞，从而控制受害者系统。

7. 2013-08-19 Juniper Networks Junos Space 安全绕过漏洞 (CVE-2013-5096)

NSFOCUS ID: 24427

http://www.nsfocus.net/vulndb/24427

综述:

Junos 是用在 Juniper Networks 硬件系统内的应用开发平台或网络操作系统。

JA1500 设备上使用的 Juniper Networks Junos Space 没有正确实现基于角色的访问控制。

危害:

具有只读权限的用户可以修改配置信息，绕过安全限制，执行未授权操作。

8. 2013-08-30 Cisco IOS XR Software 拒绝服务漏洞 (CVE-2013-3470)

NSFOCUS ID: 24531

http://www.nsfocus.net/vulndb/24531

综述:

Cisco IOS 是多数思科系统路由器和网络交换机上使用的互联网络操作系统。

Cisco IOS XR Software 的路由信息协议（RIP）进程存在安全漏洞，未经身份验证的远程攻击者可利用此漏洞造成 RIP 进程崩溃。

危害:

远程攻击者可以通过向服务器发送恶意 RIP 版本 2 报文来利用此漏洞，导致拒绝服务。

9. 2013-08-13 GNU glibc 远程缓冲区溢出漏洞 (CVE-2013-4237)

NSFOCUS ID: 24356

http://www.nsfocus.net/vulndb/24356

综述:

glibc 是绝大多数 Linux 操作系统中 C 库的实现。

GNU glibc 的 readdir_r() 例程在处理远程攻击者提供的特制 NTFS 或 CIFS 图形时存在越界写漏洞。

危害:

攻击者可以通过诱使受害者打开恶意远程目录来利用此漏洞，从而控制受害者系统。

10. 2013-08-19 Dell BIOS 多个缓冲区溢出漏洞 (CVE-2013-3582)

NSFOCUS ID: 24415

http://www.nsfocus.net/vulndb/24415

综述:

BIOS 即微机的基本输入输出系统 (Basic Input-Output System)，是集成在主板上的一个 ROM 芯片。

Latitude 及 Precision 系统上运行的 Dell BIOS 受到多个缓冲区溢出漏洞的影响。

危害:

攻击者利用这些漏洞可在受影响应用上下文中执行任意代码。

NSFOCUS 2013年9月之十大安全漏洞

声明：本十大安全漏洞由 NSFOCUS(绿盟科技) 安全小组 <security@nsfocus.com> 根据安全漏洞的严重程度、利用难易程度、影响范围等因素综合评出，仅供参考。

http://www.nsfocus.net/index.php?act=sec_bug&do=top_ten

1. 2013-09-18 Microsoft IE MSHTML 内存破坏远程代码执行漏洞 (CVE-2013-3893)

NSFOCUS ID: 24772

http://www.nsfocus.net/vulndb/24772

综述：

Internet Explorer，简称 MSIE，是微软公司推出的一款网页浏览器。

MSIE 的 MSHTML.DLL 组件存在一个远程代码执行漏洞。利用此漏洞访问已经被删除或未正确分配的对象，可导致内存破坏。

危害：

远程攻击者可以通过诱使受害者打开恶意网页来利用此漏洞，从而控制受害者系统。

2. 2013-09-06 Android WebView 组件 addJavascriptIn-terface 方法远程命令执行漏洞

NSFOCUS ID: 24622

http://www.nsfocus.net/vulndb/24622

综述：

Android 的 SDK 中提供了一个 WebView 组件，用于在应用中嵌入一个浏览器来进行网页浏览。

WebView 组件中的 addJavascriptInterface 方法用于实现本地 Java 和 JavaScript 的交互。可对 Android 移动终端进行网页挂马，从而控制受影响设备。

危害：

远程攻击者可以通过诱使受害者打开恶意网页来利用此漏洞，

从而控制受害者系统。

3. 2013-09-22 Adobe Acrobat/Reader 缓冲区溢出漏洞 (CVE-2013-3356)

NSFOCUS ID: 24774

http://www.nsfocus.net/vulndb/24774

综述:

Adobe Reader 是美国 Adobe 公司开发的一款优秀的 PDF 文档阅读软件。

Adobe Reader/Acrobat 10.1.8 之前版本、11.0.04 之前版本存在缓冲区溢出漏洞。

危害:

攻击者可以通过诱使受害者打开恶意 PDF 文件来利用此漏洞，从而控制受害者系统。

4. 2013-09-11 Adobe Flash Player/AIR 远程内存破坏漏洞 (CVE-2013-3361)

NSFOCUS ID: 24718

http://www.nsfocus.net/vulndb/24718

综述:

Adobe Flash Player 是一个集成的多媒体播放器。

Adobe Flash Player、AIR 存在远程内存破坏漏洞，目前细节未知。

危害:

攻击者可以通过诱使受害者打开恶意 SWF 文件来利用此漏洞，从而控制受害者系统。

5. 2013-09-24 Apache Struts 远程代码执行漏洞 (CVE-2013-4316)

NSFOCUS ID: 24801

http://www.nsfocus.net/vulndb/24801

综述:

Struts2 是第二代基于 Model-View-Controller (MVC) 模型的 Java 企业级 Web 应用框架。

Apache Struts 2.3.15.2 之前版本的 “Dynamic Method Invocation” 机制是默认开启的，远程攻击者可利用此机制在受影响应用上下文中执行任意代码。

危害:

远程攻击者可以通过向服务器发送恶意请求来利用此漏洞，从而控制服务器。

6. 2013-09-22 Apple iTunes ActiveX 控件内存破坏漏洞 (CVE-2013-1035)

NSFOCUS ID: 24785

http://www.nsfocus.net/vulndb/24785

综述:

iTunes 是一款数字媒体播放应用程序，是供 Mac 和 PC 使用的一款免费应用软件，能管理和播放数字音乐和视频。

Apple iTunes 11.1 之前版本的 ActiveX 控件存在内存破坏漏洞。

危害:

远程攻击者可以通过诱使受害者打开恶意网页来利用此漏洞，从而控制受害者系统。

7. 2013-09-16 Apple Mac OS X 内存破坏漏洞 (CVE-2013-1032)

NSFOCUS ID: 24754

http://www.nsfocus.net/vulndb/24754

综述：

Apple Mac OS X 是苹果电脑操作系统软件。

Mac OS X 10.8 - 10.8.4 在处理 QuickTime 视频文件时，IDSC 原子存在内存破坏漏洞。

危害：

远程攻击者可以通过诱使受害者打开恶意视频来利用此漏洞，从而控制受害者系统。

8. 2013-09-22 WebKit 内存破坏漏洞 (CVE-2013-1037)

NSFOCUS ID: 24789

http://www.nsfocus.net/vulndb/24789

综述：

WebKit 是一个开源的浏览器引擎，也是苹果 Mac OS X 系统引擎框架版本的名称。

Apple iOS 7 之前版本内使用的 WebKit 存在内存破坏漏洞。

危害：

远程攻击者可以通过诱使受害者打开恶意网页来利用此漏洞，从而控制受害者系统。

9. 2013-09-26 Cisco IOS 和 IOS XE DHCP 拒绝服务漏洞 (CVE-2013-5475)

NSFOCUS ID: 24828

http://www.nsfocus.net/vulndb/24828

综述：

Cisco IOS 是多数思科系统路由器和网络交换机上使用的互联网络操作系统。

Cisco IOS 在 DHCP 实现中存在安全漏洞，未经身份验证的远程攻击者利用此漏洞可造成拒绝服务。

危害：

远程攻击者可以通过向服务器发送恶意请求来利用此漏洞，导致拒绝服务。

10. 2013-09-12 Django 目录遍历序列 (CVE-2013-4315)

NSFOCUS ID: 24724

http://www.nsfocus.net/vulndb/24724

综述：

Django 是 Python 编程语言驱动的一个开源 Web 应用程序框架。

Django 在 ssi 模板标签的实现上存在目录遍历漏洞，攻击者可利用此漏洞获取敏感信息。

危害：

攻击者可以利用此漏洞进行目录遍历，获取敏感信息。

NSFOCUS 2013年10月之十大安全漏洞

声明：本十大安全漏洞由 NSFOCUS(绿盟科技) 安全小组 <security@nsfocus.com> 根据安全漏洞的严重程度、利用难易程度、影响范围等因素综合评出，仅供参考。

http://www.nsfocus.net/index.php?act=sec_bug&do=top_ten

1. 2013-10-09 Microsoft Internet Explorer 内存破坏漏洞 (CVE-2013-3871)(MS13-080)

NSFOCUS ID: 24910

http://www.nsfocus.net/vulndb/24910

综述：

Internet Explorer 是微软公司推出的一款网页浏览器。

Microsoft Internet Explorer 6 / 7/ 8/ 9/ 10 存在内存破坏漏洞。

危害：

远程攻击者可以通过诱使受害者打开恶意网页来利用此漏洞，从而控制受害者系统。

2. 2013-10-10 Adobe Acrobat/Reader 远程安全限制绕过漏洞 (CVE-2013-5325)(apsb13-25)

NSFOCUS ID: 24915

http://www.nsfocus.net/vulndb/24915

综述：

Adobe Reader 是美国 Adobe 公司开发的一款优秀的 PDF 文档阅读软件。

Adobe Acrobat XI (for Windows) 11.0.04、Adobe Reader XI (for Windows)11.0.04 存在 JavaScript 安全控件相关的回归漏洞。

危害：

攻击者可利用此漏洞绕过控件并运行 JavaScript URI。

3. 2013-10-11 Google Chrome 30.0.1599.66 之前版本多个安全漏洞

NSFOCUS ID: 24939

http://www.nsfocus.net/vulndb/24939

综述：

Google Chrome 是由 Google 开发的一款设计简单、高效的 Web 浏览工具。

Chrome 30.0.1599.66 存在多个安全漏洞，攻击者可利用这些漏洞绕过安全限制，欺骗地址栏内的 URI，造成拒绝服务等。

危害：

远程攻击者可以通过诱使受害者打开恶意网页来利用此漏洞，从而控制受害者系统。

4. 2013-10-16 Oracle Database Server "Core RDBMS" 组件远程安全漏洞（CVE-2013-3826）

NSFOCUS ID: 24970

http://www.nsfocus.net/vulndb/24970

综述：

Oracle Database Server 是一个对象一关系数据库管理系统。

Oracle Database Server 在 Core RDBMS 组件的实现上存在远程安全漏洞。

危害：

远程攻击者可以通过向服务器发送恶意请求来利用此漏洞，从而控制服务器。

5. 2013-10-24 Apple Mac OS X 多个安全漏洞 (APPLE-SA-2013-10-22-3)

NSFOCUS ID: 25057

http://www.nsfocus.net/vulndb/25057

综述：

OS X（前称 Mac OS X）是苹果公司为麦金塔电脑开发的专属操作系统的最新版本。

OS X 10.9 之前版本存在多个漏洞，攻击者可利用这些漏洞执行任意代码，造成拒绝服务、劫持任意会话、未授权访问、获取敏感信息、绕过安全限制等。

危害：

远程攻击者可以利用这些漏洞控制受害者系统。

6. 2013-10-25 SAP Sybase Adaptive Server Enterprise 远程代码执行漏洞（CVE-2013-6245）

NSFOCUS ID: 25069

http://www.nsfocus.net/vulndb/25069

综述：

Sybase Adaptive Server Enterprise 是关系型数据库管理系统。

SAP Sybase Adaptive Server Enterprise 在实现上存在远程代码执行漏洞，攻击者可利用此漏洞在受影响应用上下文中执行任意代码。

危害：

远程攻击者可以通过向服务器发送恶意请求来利用此漏洞，从而控制服务器。

7. 2013-10-30 Mozilla Firefox/Thunderbird/SeaMonkey 多个漏洞 (MFSA 2013-93-102)

NSFOCUS ID: 25113

http://www.nsfocus.net/vulndb/25113

综述:

Firefox 是一款非常流行的开源 Web 浏览器。Thunderbird 是一个邮件客户端，支持 IMAP、POP 邮件协议以及 HTML 邮件格式。

Firefox 25.0、Firefox ESR 24.1、Firefox ESR 17.0.10、Thunderbird 24.1、Thunderbird ESR 17.0.10、Seamonkey 2.22 存在多个安全漏洞。

危害:

远程攻击者可以通过诱使受害者打开恶意网页来利用此漏洞，从而控制受害者系统。

8. 2013-10-16 Oracle MySQL Server 远程安全漏洞 (CVE-2013-5793)

NSFOCUS ID: 24972

http://www.nsfocus.net/vulndb/24972

综述:

Oracle MySQL Server 是一个轻量的关系型数据库系统。

Oracle MySQL Server 5.1.70 之前版本、5.5.32 之前版本、5.6.12 之前版本存在远程安全漏洞。

危害:

远程攻击者可以通过向服务器发送恶意请求来利用此漏洞，从而控制服务器。

9. 2013-10-31 Cisco IOS XE TCP 段重组拒绝服务漏洞 (CVE-2013-5546)

NSFOCUS ID: 25142

http://www.nsfocus.net/vulndb/25142

综述:

Cisco IOS 是多数思科系统路由器和网络交换机上使用的互联网络操作系统。

Cisco IOS XE 的 TCP 段重组功能存在安全漏洞，未经身份验证的远程攻击者利用此漏洞可造成系统重载。

危害:

远程攻击者可以通过向服务器发送恶意请求来利用此漏洞，导致拒绝服务。

10. 2013-10-14 D-Link 路由器固件后门漏洞

NSFOCUS ID: 24957

http://www.nsfocus.net/vulndb/24957

综述:

D-Link DIR-100 是集成了防火墙功能的小型宽带路由器。

D-Link 的几款 Planex 路由器所使用的固件 v1.13 版本中存在后门漏洞。

危害:

远程攻击者无需身份验证即可访问路由器的 Web 接口，查看或者更改受影响设备设置。

THE EXPERT
BEHIND GIANTS

NSFOCUS

公司总部: 北京市海淀区北洼路4号益泰大厦三层 010-68438880
服务热线: 400-818-6868 值班热线: 13321167330（非工作时间） 技术支持传真: 010-68437328
技术支持网站: http://support.nsfocus.com 技术支持邮箱: support@nsfocus.com

www.nsfocus.com

JUST CHANGE

JUST HERE JUST NOW

全新的网络环境

你需要下一代防火墙

一体化安全解决方案：安全、易用、稳定

NSFOCUS NF

绿盟下一代防火墙

NSFOCUS NEXT-GENERATION FIREWALL