基于动态异构冗余机制的路由器拟态防御体系结构

马海龙, 伊 鹏, 江逸茗, 贺 磊

解放军信息工程大学信息技术研究所 郑州 中国 450000

摘要 路由器作为网络空间的基础核心要素, 其安全性能对网络安全具有决定性意义。但由于它的封闭性、专用性和复杂性, 导致其存在的漏洞更多, 后门隐藏更深。目前对路由器的安全防御手段均为被动式“补漏洞、堵后门”的“亡羊补牢”式的防御, 不仅防御滞后更无法应对未知的安全威胁。本文基于拟态防御技术, 在路由器体系架构上引入异构冗余功能执行体, 通过动态调度机制, 随机选择多个异构执行体工作, 在相同外部激励的情况下, 通过比对多个异构功能执行体的输出结果, 对功能执行体进行异常检测, 实现路由系统的主动防御。实验结果表明, 该架构可以明显提升攻击链中每一步攻击的实施难度, 增加攻击成本, 并能抵御基于未知漏洞与后门的攻击。

关键词 拟态防御; 动态; 异构; 冗余; 路由器

中图法分类号 TP393.08 DOI 号 10.19363/j.cnki.cn10-1380/tn.2017.01.003

Dynamic Heterogeneous Redundancy based Router Architecture with Mimic Defenses

MA Hailong, YI Peng, JIANG Yiming, HE Lei

Institute of Information Technology, PLA Information Engineering University, Zhengzhou 450000, China

Abstract As a fundamental core element of cyberspace, the security performance of router plays a decisive significance in network security. However, the closeness, specificity and complexity of router lead to more loopholes and make backdoors hidden deeper. Currently, defense means of router are passive, which is “mend the fold after the sheep have been stolen”-like. Such defense means is not only hysteretic but also helpless against unknown security threats. Based on mimicry defense technology, heterogeneous redundancy function entities are introduces to the architecture of router. With dynamic scheduling mechanism, multiple heterogeneous execution entities are randomly selected to work. Under the same external motivations, by comparing the output of heterogeneous executing entities and conducting anomaly detection on heterogeneous executing entities, the routing system could perform active defense. Experimental results show that this architecture can significantly increase the attack difficulty in every step of the attack chain, increase the cost of attacks, and can withstand attacks based on unknown vulnerabilities and backdoors.

Key words Mimic defense; Dynamic; Heterogeneous; Redundancy; Router

1 引言

1.1 背景

路由是信息交互的大脑, 决定网络中的数据从源到目的地时端到端的路径。路由器是通过路由计算实现数据包路由转发的设备。路由器作为网络空间的基础核心要素, 位于网络空间底层, 互联多种异构网络。它作为网络空间信息基础设施的核心装备, 覆盖整个互联网的核心层、汇聚层和接入层。根据路由器在网络中的位置和功能不难发现, 路由器作为网络基础设施的核心枢纽, 其安全性能对网络安全具有决定性意义。然而, 路由器的安全现状并不容乐观。斯诺登称: 美国国家安全局通过思科路由器监控中国网络和主机, 而思科几乎参与了中国所有大型网络项目的建设[1]。思科 CEO 约翰钱伯斯致信奥巴马称: 美国国家安全局的监控活动损害了公众对美国科技产品的信心, 要求总统进行干预, 遏制国安局的监控活动[2]。国家互联网应急中心对 Cisco、Linksys、Netgear、Tenda、D-link 等主流网络设备厂商的多款路由器产品进行分析, 确认其存在预置后门

通讯作者: 马海龙, 博士, 副研究员, Email: longmanclear@163. com。

本课题得到国家重点研发计划(2016YFB0800103)资助。

收稿日期: 2016-09-12; 修改日期: 2016-10-09; 定稿日期: 2016-12-03

漏洞[3]。这些事件充分反映了路由器安全形势之严峻。

路由器在网络中的位置和路由转发功能决定了它将是攻击者实施攻击的一个极佳切入点。路由器一旦被攻击者控制, 将会对整个网络产生难以估量的危害。针对主机的攻击, 主要危害的是主机自身, 而针对路由器的安全攻击, 危害的则是与路由器相连接的整个网络, 其危害远远超过针对主机的攻击。通过路由器可以方便地获取用户隐私数据、监控用户上网行为、获取账户密码信息、篡改关键用户数据、推送传播虚假信息、扰乱网络数据流向、瘫痪网络信息交互、直接发起网络攻击等等。

路由器的安全威胁, 国产品牌主要来自两个方面: 一是系统设计与实现中的漏洞无法避免, 二是使用开源代码无意识带入的陷门。对国外品牌而言, 漏洞同样无法避免, 更重要的是, 外国政府为实现战略意图植入的后门。因此, 漏洞和后门等带来的不确定威胁是路由器安全的头号大敌。

作为一种专用封闭系统, 路由器的防御难点主要表现在三个方面。一是缺乏辅助手段, 和通用系统不同, 它没有防火墙、防病毒等相关安全防护手段, 大多数路由器毫无保留地暴露在恶意流量中; 二是可用漏洞更多, 因为设计与实现环节的巨量代码决定了其潜在漏洞众多, 而由于路由器是一种专用封闭系统, 门槛更高, 设计缺陷更难发现, 因此具有大量的潜在可用漏洞; 三是后门隐藏更深, 路由器作为一种专用封闭系统, 其后门更难被发现, 而且作为网络基础设施, 其后门植入往往承载着国家战略意图, 是一种国与国之间的对抗, 技术水平更高。要在不改变、不掩饰路由器功能和性能的前提下, 使用“有毒带菌”的器部件, 包容系统的漏洞和后门, 同时提供高安全性, 就如同在沙滩上搭建一座稳固的沙塔, 其难度不言而喻。面对这一难题, 研究提出了一种基于动态异构冗余机制(Dynamic Heterogeneous Redundancy, DHR)的路由器拟态防御体系结构, 以具有不确定性的防护机制, 对抗不确定的安全威胁。改变传统的“挖漏洞、堵后门”打补丁式的网络安全防御方法, 改变易攻难守的安全防御窘境。

1.2 相关研究

Andrew 在文献[4]中对思科路由器面临的安全威胁、如何对其进行攻击以及如何进行安全防护等内容进行了详尽的阐述。通过分析可以看出, 对路由器的安全防护手段更多的侧重在管理员能否正确的配置和使用路由器、能否及时查看分析日志、能否及时给系统打补丁堵漏洞, 而缺乏有效的防御技术与手段。目前针对路由器漏洞和后门的应对措施主要采用事后补救的被动防御方法, 而本文所基于的 DHR 防御机制则是系统架构级别的主动防御手段。

在异构方面, Zhang 等人[5]提出了采用异构结构的网络模型解决网络的生存性难题。Byung-Gon 等人[6]研究了采用异构结构解决单机拜占庭故障问题。Caballero 等人[7]采用图论的方法研究了路由器厂商的多样性对提升网络弹性的作用。Keller 等人[8]提出了基于异构的开源路由软件, 通过并行运行并比对的方法, 解决 BGP 协议的内在漏洞问题。在动态机制方面, 美国提出的移动目标防御(Moving Target Defence, MTD)[9]技术, 通过主动改变通信服务端点的地址或端口[10-12]来实现主动防御。

这些相关研究要么利用异构性, 要么利用动态性来提升目标系统的生存性, 实现对内在漏洞(或 BUG)的容忍。本文所采用的 DHR 机制, 通过引入动态异构冗余等多个特性以及多模表决机制, 在不影响目标系统功能性能的前提下, 在容忍系统漏洞和后门的条件下, 实现路由器的体系化防御。

2 攻击场景与防御体系模型

2.1 攻击场景

攻击者首先通过扫描探测的方法, 确认目标路由器的相关信息, 例如设备型号、操作系统版本、周围网络拓扑情况、开放端口号、启动的网络服务及版本等信息。基于扫描探测信息识别目标路由器上存在的脆弱点和漏洞, 然后利用脆弱点或漏洞进行权限提升。在提权后, 可以修改路由器的 ACL 或者 Context-Based Access Control(CBAC)规则, 打开恶意流量进入内部网络大门; 也可以部署隐蔽通道, 对经过路由器的所有数据实施嗅探和中间人攻击。更进一步, 可以在路由器系统上设置后门, 例如著名的 SYNful Knock[13], 进行长期持续的隐蔽控制和信息获取。

2.2 基于动态异构冗余的路由器拟态防御体系模型

路由器从功能上可以划分为三个平面: 控制平面、管理平面和数据平面。控制平面包括各种不同的路由协议软件(如 BGP[14]、OSPF[15]、RIP[16]等), 负责路由计算。管理平面包括多种配置管理软件(如 CLI、SNMP、HTTP 等), 负责系统配置管理维护。数据平面负责数据查表转发, 绝大多数采用硬件逻辑实现, 本文没有涉及针对数据平面拟态防御机制。

对于路由器软件系统, 无论是管理软件还是控制软件, 其功能流程都可以概括为的“输入-处理-输

出”模型(Inputs Process Outputs, IPO 模型)。将进行消息处理的单元定义为功能执行体, 路由器软件系统包含各种路由协议的功能执行体和各种管理软件的功能执行体。功能执行体存在的漏洞和后门可以被攻击者扫描探测并利用, 进而进行提权、系统控制和信息获取。针对攻击者对路由器各功能执行体的攻击步骤, 本文采用邬江兴院士提出的拟态防御机制[17-19], 设计了基于动态异构冗余的路由器拟态防御体系(DHR based Router Architecture for Defense)模型, 简称 DHR2 模型, 如图 1 所示。该结构模型针对每一种软件系统的功能, 引入多个异构冗余的功能执行体, 对同一输入进行处理, 并对多个功能执行体输出的消息进行多模表决, 识别哪个功能执行体输出消息异常, 进而进行路由系统的安全防御。

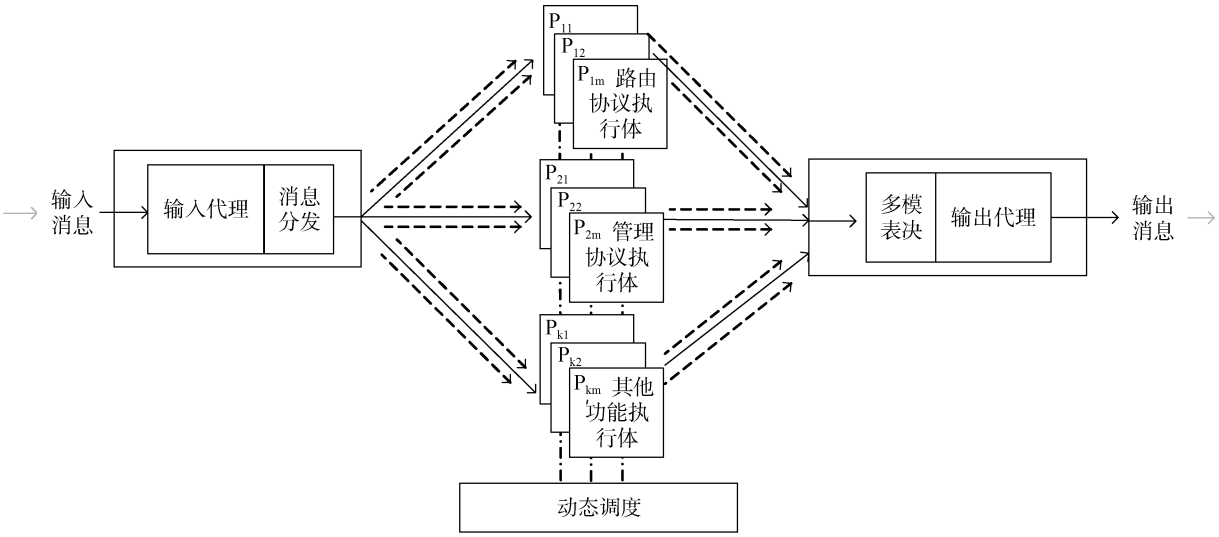

图 1　基于动态异构冗余的路由器拟态防御体系结构模型

针对系统的每一种软件功能单元可以用以下模型来描述: 存在一个功能等价的异构功能执行体池 $\overline{P}=(P_1,P_2,\cdots,P_n)$, 动态从池中选择 m 个执行体 $\overline{P}'=(P_1,P_2,\cdots,P_m)$ 进行工作。由输入代理模块将输入消息分发给 $m(m\leqslant n)$个执行体, 由它们对同一个输入进行计算后得到 m 个输出结果, 对 m 个结果进行基于某种算法的多模表决, 得到归一化的输出结果, 输出结果由输出代理操作后输出。

基于动态异构冗余的路由器防御体系结构模型, 内含移动目标防御机制, 是一种主动防御模型, 可以有效的应对已知和未知的安全威胁。这是由该结构的三个特性决定的。

异构性: 异构性即功能等价的两个执行体结构组成不相同, 描述的是两个执行体之间的差异性, 这种差异性可以保证同样的攻击不会同时使两个执行体同时失效[20-22], 当然不包含针对功能原理的攻击。功能执行体的漏洞和后门具有独特性, 思科 2600 系列路由器的 SNMP 软件功能漏洞与 6500 系列的 SNMP 漏洞不同, 与 Juniper 路由器的 SNMP 漏洞相同的概率几乎为零。因此, 一种攻击方法只会对一种功能执行体发生作用。DHR2 模型通过引入多个异构功能执行体, 并行对输入的攻击消息进行处理, 并通过对多个异构执行体的输出结果进行一定语义上的多模表决, 就可以识别出异常的发生, 进而对威胁进行感知, 并针对威胁做更一步处理。但是异构性的大小是有上界的, 完全异构的部件实际中是不存在的, 在工程上也是难以实现的。

冗余性: 冗余性是指工作集的异构并行执行体的多样化。直观上认为, 冗余性可以支持表决结果的正确性。异构执行体在面临同一种攻击时, 其输出响应是不同的, 如果仅依赖于两个执行体的输出结果, 就难以准确甄别出究竟哪个执行体遭受了攻击。通过增加异构执行体工作集中的执行体的数量, 可以明显提升威胁感知的准确率, 同时也为动态性提供支撑。当然, 冗余性过强, 势必增加系统的工程成本。

动态性: 动态性是指在不同时刻下轮换对外呈现的工作执行体, 是一种主动防御手段。动态性的作用主要体现在以下两个方面: 1)通过不定时地改变工作执行体, 降低单位时间内特定部件的暴露时间, 增加系统结构信息的不确定性, 减小漏洞被发现的风险; 同时使系统处于不断更新的状态, 针对渐进式攻击等最初可能难以被察觉的行为, 动态性的存在可以清除攻击的前期努力, 对于潜在漏洞、后门的状态跳转等也具有清理作用。2)动态性是在时间维度上对多样性的扩展, 在未感知到威胁时, 动态性可

以阻碍依赖特定后门的攻击行为, 有效降低了攻击的成功率; 在感知到威胁发生时, 动态替换并隔离被感染执行体, 可有效阻断攻击者对目标系统的持续控制, 并保证系统功能的完整性和持续性。

3 系统架构设计

3.1 总体框架

要依据 DHR2 模型构建内嵌主动防御机制的路由器, 需要解决以下几个问题。

一是功能执行体的异构性构建。对于商业公司而言, 不会投入人力开发两套功能相同的软件, 但是对于路由协议而言, 目前已经存在几种较为成熟的开源软件包, 例如 XORP[23], Quagga[24], BIRD[25]等, 它们采用不同的语言, 由不同的团队采用不同的架构开发, 存在相同漏洞的概率几乎为 0[26-27]。而且, 一些主流路由器厂商推出了路由器仿真器, 例如思科的 Simulator[28]、PacketTracer[29]、Juniper 的 Olive[29]、华为的 eNSP[31]等, 以及开源仿真器 GNS3[32]、Qemu[33]等的出现, 可以使得在没有源码的情况下, 基于路由器二进制文件仿真出具备等价功能的路由系统, 这为异构性的构建提供了更富足的条件。更进一步, 同一款路由软件的多个实现版本之间也存在一定程度上的异构性, 在修复旧版本的 bug 时, 会引入新 bug, 但是可以同时使用新旧版本并行运行来确保旧的 bug 出现时用新的版本替换, 同时新引入的 bug 可利用旧版本来屏蔽。例如, Quagga0.99.9 中引入的 bug, 有 30%没有在 0.99.1 中出现[26]。这种方法可以促进新版本的成熟。

二是功能执行体的动态调度。DHR2 模型的动态性要求功能执行体能够从执行体池中动态选择加载, 对异常执行体能够下线清洗, 对工作的执行体能够动态调度其对外的呈现方式。为了实现动态性, 采用网络功能虚拟化(Network Function Virtualization, NFV)技术来实例化异构冗余的执行体, 将功能执行体模块通过虚拟机承载, 从而方便地实现了执行体的动态调度, 同时降低异构冗余引入的实现成本。

三是消息处理路径上的分发代理和多模表决点的插入。传统路由器实现结构中软硬件紧耦合, 内部通信接口自定义, 要在消息处理路径上插入相应的消息分发或者多模表决模块, 从工程角度而言, 将难以实现。软件定义网络(Software-defined networking, SDN)技术[34]的出现为 DHR2 模型的实现提供了良机。SDN 的南向接口可以将传统路由器中软硬件之间的内部消息接口标准化, 所有消息通过标准 Openflow 接口[35]承载, 通过在 Openflow 控制器(Openflow Controller, OFC)之上插入分发代理和多模表决模块, 就可以实现对进出路由器的所有消息进行分发和判决处理。同时, 异构路由执行体计算的路由表可以在此处进行多模表决, 确保路由计算结果的正确性。

基于上述论述, 设计的 DHR2 路由器系统架构如图 2 所示。分为硬件层面和软件层面。硬件层面为标准的 Openflow 交换机(Openflow Switch, OFS), 软件层面包括 OFC、代理插件、多模表决、异构执行体池、动态调度以及感知决策单元(这些单元统称为拟态插件)。

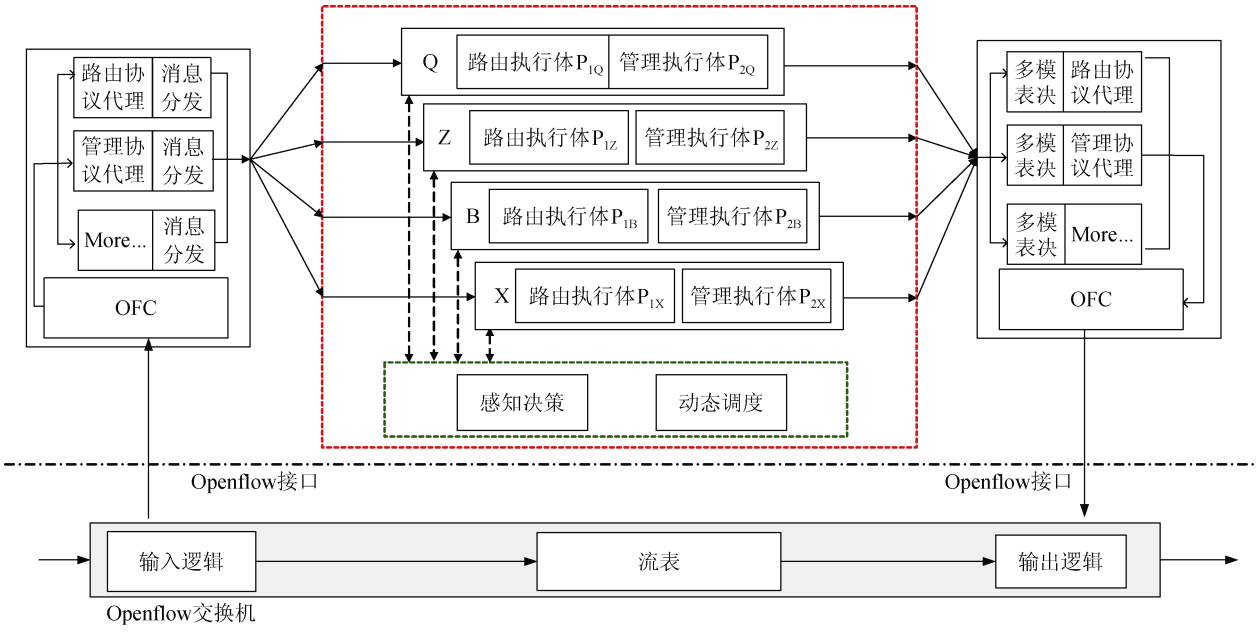

图 2 路由器 DHR 工作模式

OFC 基于进入消息的协议类型进行消息分发, 分发至不同的代理插件, 由代理插件基于各自的协议机制进行有状态或者无状态处理, 并对产生的多份副本消息进行分发; 各个执行体的协议软件对收到的消息进行处理, 产生流表信息和对应的输出消息; 多份输出消息到达多模表决, 进行判决后经由 OFC 承载, 并通过 OFS 发送到线路上; 同时各个执行体产生的路由表信息到达多模表决, 经由路由表多模表决后产生最终的表项, 经由OFC下发给OFS。

3.2 功能单元设计

本节详细介绍本系统中各重要部件的功能特性和工作原理。

3.2.1 代理插件单元

代理插件单元是消息的出入口, 消息包括路由协议报文、网管协议报文等。因此, 代理插件单元按照代理的协议类型可以分为路由协议代理和管理协议代理等, 其功能主要包括以下几个方面:

1) 输入消息的动态复制分发

代理插件负责将收到的消息复制分发给多个异构功能执行体进行处理。对消息的操作并非是简单的复制分发, 而是与所代理的协议机理有一定的关联性。这些操作可以概括有状态的操作和无状态的操作, 前者需要对复制分发的消息做一定的修改并记录状态, 后者就是仅仅简单的复制分发。像 OSPF、ISIS 等路由协议代理在进行消息分发时需要对原始报文中的某些域做一定的修改以适配各个执行体中的协议状态机; 而像 SNMP 或者 Telnet 等管理协议代理, 则仅仅需要将传输层承载的载荷复制分发到执行体即可。

2) 非授权业务消息的识别、过滤及威胁感知

每一个代理插件处理的消息分别具有不同的功能特性, 如果代理插件处理了不属于自己功能范围的消息, 可能会引入一定的安全威胁。因此, 在代理插件层叠加一定的安全过滤列表, 对进入各个代理的消息进行缜密的检测, 可以把一定的安全威胁阻挡在功能执行体之外。

同时, 代理插件作为消息进入系统的第一关口, 将执行体面临的攻击面前移到此处。这也给系统嵌入防火墙、入侵防护和入侵检测等功能提供了架构上支持。通过在此处应用传统的安全防护手段, 一方面, 可以将纷繁复杂的恶意流量阻挡在外, 减少了内部功能执行体遭受的威胁; 另一方面, 又可以通过统计分析和入侵检测方法, 进行威胁感知, 并提前预警。

3) 子网隔离

复制后的消息被分发至不同的异构执行体, 这些异构执行体采用相同的地址配置和功能配置, 必须采用特殊的子网隔离策略, 才能保证功能执行体能够同时并行运行, 且功能机制正常。

3.2.2 多模表决单元

多模表决单元是输出消息的必经通道, 以多个异构执行体的输出为输入, 通过进行比特级、载荷级行为级甚至是内容级的比对, 实现对系统内部功能执行体异常的感知。感知决策单元依据安全等级要求指定该单元所采用的多模表决算法, 例如择多判决、权重判决、随机判决等。多模表决并不能准确判决出哪个异构执行体发生了异常, 却能够识别出内部执行体发生了异常, 并且将异常屏蔽掉。例如, 路由器转发表的某个表项更改, 当且仅当多数功能执行体输出结果完全一致情况下才能实现更改操作。否则, 仅仅一个执行体发出表项修改请求, 这种请求将被多模表决裁决掉, 使系统具备入侵容忍能力, 保证输出结果的一致性和正确性。同时该判决单元会提取多维度的多模表决结果, 反馈给感知决策单元, 为决策单元对执行体的可信评估提供素材数据。

如果要绕过多模表决环节, 必须精确协同输出结果, 这是非常困难的事情, 因为开发团队不同、平台不同、算法不同, 相互之间除了有共同的输入序列和给定的功能外, 独立工作的异构功能执行体中的“恶意”功能, 很难协调各自的输出实现完全或者多数的一致, 更何况无法知道具体的表决方法。所以, 系统的异构冗余程度越高, 表决内容越丰富, 协同化攻击的难度就越大。

3.2.3 动态调度单元

动态调度单元的主要功能是管理异构执行体池及其功能子池内执行体的运行, 按照决策单元指定的调度策略, 调度多个异构功能执行体, 实现功能执行体的动态性和多样性, 增加攻击者扫描发现的难度, 隐藏未知漏洞和后门的可见程度。动态调度单元设计的关键环节是执行体的调度策略。设计了两种维度的调度策略供决策单元选择:

1) 基于执行体可信度的随机调度策略

异构执行体池中的每一个执行体都有一个可信度的属性值, 调度单元依据可信度的大小, 采用基于可信度权重的随机调度方法, 即可信度越高, 被优先调度的可能性越大。系统初始状态时, 各个执行体的可信度相同, 当某个执行体被检测出工作异常时, 其可信度会急剧下降, 同时, 随着执行体持续正常工作时间的增加, 其可信度会缓慢的提升。该调度策略在保证系统基本性能需求的基础上, 追求防御安全增益的最大化。

算法 1. 执行体可信值生成算法.

初始化流程.

输入: 执行体池 *EE*

FOREACH *ee* IN *EE:*

ee.trust=2000, ee.state=NORMAL, ee.increaseRate=0

多模表决发现执行体 *eex* 为异常执行体时 *trust* 值处理算法.

输入: 异常执行体 *eex*

IF *eex* IS *abnormal:*

eex.state = SUSPECT

eex.trust = eex.trust - 500

IF *eex.trust < 1:*

eex.trust = 1

*eex.increaseRate=eex.trust/HALF_STEP*60*

#HALF_STEP=20, 具体设置与调度时间有关, 设置原则为保证下一个调度时刻, 执行体的 *trust* 值不会增长到最大信用值 2000

IF *eex.trust* < 100:#信用值低于 100 后, 该执行体将不再被调度

eex.state = FORBIDDEN

eex.forbiddenTime = now()

执行体 *trust* 值的定时更新算法.

输入: 执行体池 *EE*

WHILE *TRUE:*

sleep(5)#每 5s 递增一次信用值

FOREACH *ee* IN *EE:*

ee.trust = ee. trust + ee.increaseRate

IF *ee.trust > 2000:*

ee.trust = 2000

ELIF *ee.trust > 1250:*

ee.state = NORMAL

ELIF *ee.trust > 100:*

ee.state = SUSPECT

IF *ee.state == FORBIDDEN:*

IF *now() - ee.forbiddenTime > 3600:*#如果一个执行体被禁用超过 1 小时, 将被重新启用

ee.state = NORMAL

ee.forbiddenTime = now()

2) 基于执行体性能权重的随机调度策略

异构执行体池中的每一个执行体都有一个性能权重的属性值, 用于表征每一个执行体的性能优劣。调度器依据性能权重的大小, 采用基于性能权重的随机调度方法, 即性能权重越高, 被优先调度的可能性越大; 每个执行体的性能权重在系统初始化时由决策单元赋予, 通常情况下保持不变, 特殊情况出现时(如执行体系统故障等)由感知决策单元进行修改。该调度策略在为系统提供基本安全性保障的基础上, 追求系统性能的最大化。执行体性能权重生成算法与可信值生成算法类似。

3.2.4 异构执行体池

异构执行体池中存储了具有不同元功能的异构执行体单元, 其具有异构性、冗余性、多样性的特点。异构执行体池保证了漏洞扫描工具的输出结果的多样性, 增大攻击者分析漏洞和利用后门的难度, 使整个路由系统具有入侵容忍能力。相同功能的执行体被划分到不同的子网, 彼此之间相互隔离, 不能通信。处于同一个子网的异构执行体属于不同的功能面。由代理插件确保不同子网内执行体数据和状态机的一致性和完整性。

理想状态下, 每一种功能对应一个异构执行体池, 通过灵活组装功能执行体的方法构建出一个具备全路由器功能集的路由软件实例, 例如, 一种路由器软件实例可以使用 Quagga0.99.9 的 OSPF 功能执行体, XORP1.8.5 的 BGP 功能执行体, Cisco c7200 的 SNMP 功能执行体和 Juniper M10i 的 Telnet 功能执行体等。但由于功能执行体之间缺乏标准的接口, 使得这种组装方案难以实施。实际工程中, 采用粒度较粗的方法进行实现异构冗余, 即将一个团队开发的适用于一种路由器型号的一个版本的整套软件套件视为一个全功能的异构执行体, 例如 Quagga0.99.9 是一个异构执行体, Cisco 的 c2600-i-mz.120-3.T3.bin 视为另一种异构执行体。动态调度和感知决策的对象也是整个软件套件, 如果检测到 Quagga0.99.9 实例中的 BGP 协议存在异常, 则 Quagga0.99.9 整套软件实例被认定存在问题而被调度下线, 而不是仅仅调度操作 Quagga0.99.9 中的 BGP 协议软件。虽然这种执行体的异构性的构建粒度较粗, 但开发团队、操作系统、开发工具等均存在较大差异, 依然可以保证各漏洞和后门的不一致性, 使得各个执行体发生共模故障的可能性很低。

3.2.5 感知决策单元

感知决策单元主要负责统管代理插件、多模表决、异构执行体池、动态调度等单元, 它定义了代理插件的消息分发方法, 多模表决单元的表决算法, 动态调度单元的调度方法等。同时, 它负责从多单元收集系统运行过程中的各类异常和状态信息, 进行系统环境感知, 并在此基础上, 通过统计分析研判, 实现对异构执行体的动态组合方式、多模表决算法以及代理插件等单元的运行参数的主动调整, 使其自主跳变, 主动防御。还要对不可信的功能执行体进行下线清洗和数据回滚, 确保工作执行体实例的功能正常。

4 系统实现

依据系统架构设计方案, 采用 SDN 技术、NFV

技术以及虚实结合的方法实现 DHR2 验证系统。

4.1 系统实现架构

系统分为三层, 如图 3 所示, 包括设备层、控制层和应用层。底层为设备层, 采用硬件设备实现路由器的查表转发功能。中间层为控制层, 实现代理插件、动态调度、多模表决和感知决策等功能。上层为应用层, 部署异构执行体, 实现路由控制和网络管理功能。设备层采用品科 P-3297 SDN 交换机, 控制层运行在 Ubuntu12.04 TLS 操作系统之上, 包括 OFC 与拟态插件集。应用层采用虚拟化技术和实体设备搭建异构执行体, 虚拟机 Hypervisor 管理器采用 KVM[36]实现。系统内嵌的七种异构执行体, 包括开源的 Quagga 0.99.22.4 路由套件、XORP1.8.5 路由套件、思科 c2600-i-mz.120-3.T3.bin、Juniper Juniper10.2R1.8.vmdk 以及中兴 ZXR10、烽火 Fengine S5800、迈普 MP3900 路由器各一套, 构建了一个包含七个异构执行体的 DHR2 验证系统。这些执行体研发团队不同、采用的操作系统迥异, 足以保证 DHR2 系统的异构性。

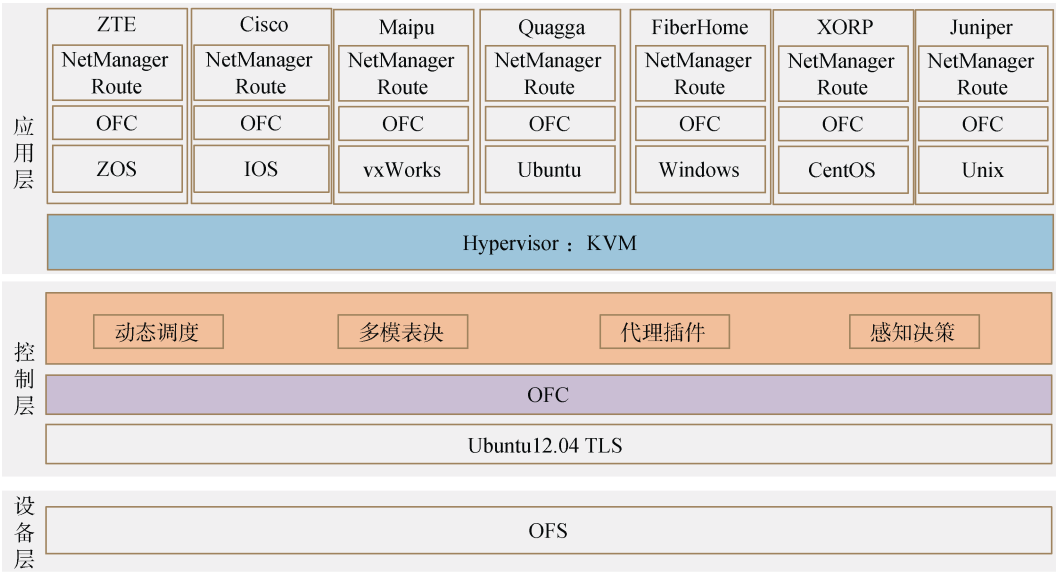

图 3 DHR2 系统逻辑结构

该分层结构实际上就是在传统的 SDN 路由器结构基础上, 在 OFS 和 OFC 之间引入了实现 DHR 功能所需的相关模块, 并在 Controller 层面引入多个异构执行体。设备层与控制层采用标准 Openflow 接口, 建立一个 Openflow 通道; 控制层与应用也采用标准的 Openflow 接口, 每一个功能执行体分别与控制层建立独立的 Openflow 通道。对于 OFS 而言, 它连接控制层的控制器, 由控制层对 OFS 进行配置管理。对于应用层而言, 每一个功能执行体分别视控制层为一台 OFS, 并对其进行配置与管理。

消息处理流程如图 4 所示, 需要本地处理的消息 A, 经由 Openflow PacketIn 消息 PI(A)承载, 到达控制层, 运行于控制层的代理插件对消息 A 处理后衍生多份副本消息(A1, A2, A3), 这些副本消息被分发给多个虚拟交换机代理(Visual Switch Agent, VSA), 并由 VSA 通过对应的 Openflow 通道传送至应用层的各个异构执行体。各个异构执行体产生的路由、管理消息以及计算得到的路由表(或称流表), 通过各自 Openflow 通道传送给控制层的 VSA。控制层的 VSA 将 Openflow 通道中承载的消息解封装后, 交付

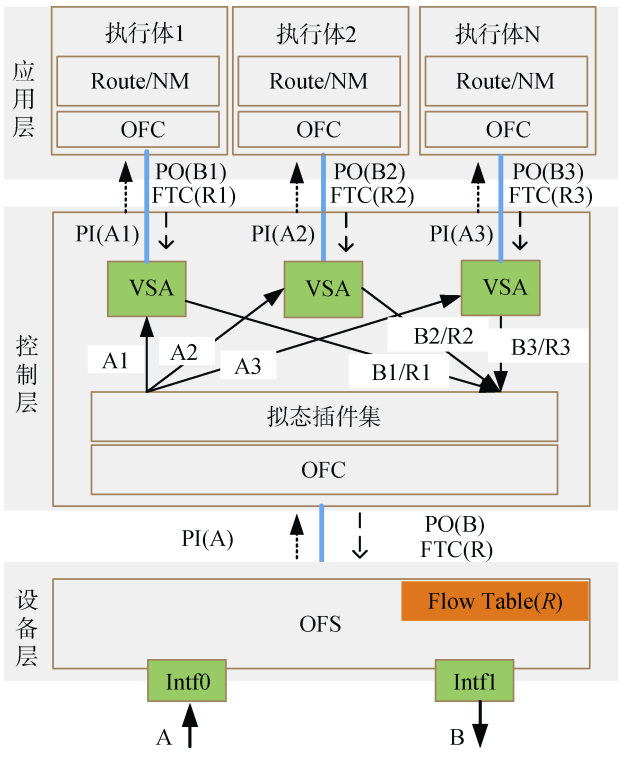

图 4 消息处理流程

给多模表决单元, 由它对多个异构功能执行体产生的同类消息和路由表进行判决后, 通过 Openflow 通道封装为 PO(B)、FTC(R)消息, 下发给 OFS。对于 PacketOut 消息 PO(B), OFS 将其中载荷 B 从指定接口发送出去; 如果是流表配置消息 FTC(R), 则更新本地流表 *R*。

4.2 拟态插件集的实现

拟态插件集中代理插件单元作为功能执行体和外部路由器之间通信的关口, 需要对进出执行体的消息进行操作。这些消息包括各个层面的消息, 例如 IP 层的 OSPF、ISIS, 传输层 TCP、UDP 以及应用的 BGP、RIP、FTP、SNMP 等。每一种协议都有自己的状态机, 简单的复制分发可能导致各个执行体的相应协议无法正确运行。同时, 还要考虑执行体的切换以及上下线过程中的状态恢复、数据同步等对各个协议状态机的影响。因此, 每一种类型的协议, 其代理插件机制是不同的, 具体可以查阅相关专利[37-38]。

4.3 异构执行体的实现

异构执行体要求提供标准的 Openflow 接口, 在控制器上运行各种路由协议和管理协议。而目前能够运行在控制器上路由软件很少, 开源软件中 Routeflow[39]提供了一种架构, 使得一些开源路由软件无缝迁移到该架构下, 实现对 Openflow 接口的支持。异构执行体 Quagga 和 XORP 便采用这种方式构建。

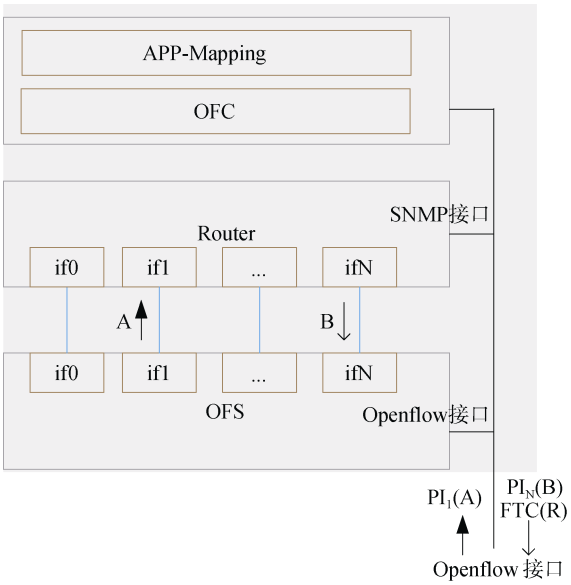

图 5 基于路由器的 Openflow 接口实现框架

国内外路由厂商正在研发 Openflow 接口的路由软件, 但未有商业化的版本出现。为了保证异构性, 必须利用传统路由器实现对 Openflow 接口的支持。为此, 设计了一种基于路由器的 Openflow 接口实现框架。如图 5 所示, 通过在路由器前置一台 OFS, 并在 OFC 上增加一个 APP-Mapping 软件即可构建。APP-Mapping 通过 OFC 建立两个 Openflow 通道, 分别连接外部的 OFS 和内部的 OFS, 并维护两者的接口映射表, 当从外部收到 PacketIn 消息 PI_1(A)后, 查映射表, 将载荷 A 重新封装为 PacketOut 消息发送给内部 OFS, 由 OFS 拆封装后通过 if1 接口发送给 Router。同理, Router 产生消息 B 到达内部 OFS 的 ifN 后, 封装为 PacketIn 消息到达 OFC, 经 App-Mapping 查映射表后, 重新封装为 PO_N(B)发送给外部 OFS。同时, Router 提供了 SNMP 接口, APP-Mapping 可以通过该接口获取 Router 上的路由表, 生成流表, 通过流表配置消息 FTC(R)配置到外部 OFS。该框架与内置的路由器类型无关, 但可以使用传统路由器实现对 Openflow 接口的支持。

利用该框架采用 GNS3 虚拟仿真环境, 构建了支持 Openflow 接口的思科执行体和 Juniper 执行体。如图 6 所示, 事先在 OpenvSwitch 上配置缺省流表, 将从所有数据接口收到的消息统一上交给 OFC。在 OFC 上建立了 OpenvSwitch 接口与 Router 接口以及外部交换机的接口映射表, 通过 SNMP 协议读取 Router 路由表、ARP 表以及接口表, 组合生成流表, 实现转发表到流表的转换。

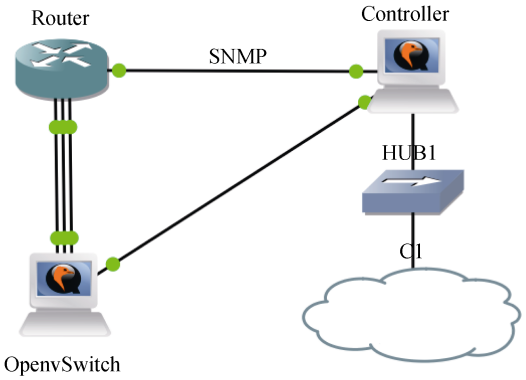

图 6 支持 Openflow 接口的思科执行体

对于没有开源, 也没有提供虚拟化运行机制的其他厂商路由器, 只能通过组合路由器和 SDN 交换机构建, 中兴、迈普、烽火的异构执行体就是基于上述结构, 采用实体设备构建。

5 实验测试与分析

DHR2 验证系统采用基于执行体可信度的随机调度策略, 每一个执行体的初始权重值一样, 调度时间间隔为 2 至 5 分钟之间的一个随机值。

5.1 系统信息的扫描探测

依据攻击场景的描述, 本节首先采用 Nmap[40]工具对 DHR2 验证系统进行扫描探测, 检测目标系统开放的端口、启动的服务以及操作系统版本等信息, 这是攻击者做渗透攻击所进行的第一步操作, 这些信息将决定攻击者后续攻击所采用的方法手段和工具。

```
PROTOCOL STATE         SERVICE
1        open          icmp
6        open          tcp
17       open          udp
88       open|filtered eigrp
89       open|filtered ospfigp

PORT    STATE         SERVICE VERSION
23/tcp  open          telnet  Cisco IOS telnetd
79/tcp  open          finger  Cisco fingerd
179/tcp open          bgp     ?
67/udp  open|filtered dhcps
161/udp open          snmp    SNMPv1 server (public)
520/udp open|filtered route
Device type: router
Running: Cisco IOS 12.X
OS CPE: cpe:/h:cisco:2514_router cpe:/o:cisco:ios:12.1
OS details: Cisco 2514 router (IOS 12.1)
Network Distance: 1 hop
Service Info: OS: IOS; Devices: switch, router; CPE: cpe:/o:cisco:ios
```

图 7 第 1 次扫描探测结果

首先, 在不同时刻对目标系统进行了两次扫描探测, 下图 7 和图 8 给出了两次扫描探测的结果。从结果上分析得知, 两次扫描得知系统开放的端口信息基本一致, 但是服务版本和操作系统版本则差异性较大, 第一次扫描结果认为是思科路由器 c2514, IOS 版本为 12.1, 第二次扫描结果认为是 Quagga 软件路由器, 版本为 0.99.22.4。两次迥然不同的测试结果将会很大程度上让攻击者困惑, 使得攻击者无法进行下一步的攻击行为。

```
PROTOCOL STATE SERVICE
1        open  icmp
6        open  tcp
17       open  udp

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.0 (protocol 2.0)
179/tcp  open  bgp?
2601/tcp open  quagga  Quagga routing software 0.99.22.4 (Derivat
2602/tcp open  quagga  Quagga routing software 0.99.22.4 (Derivat
2604/tcp open  quagga  Quagga routing software 0.99.22.4 (Derivat
2605/tcp open  quagga  Quagga routing software 0.99.22.4 (Derivat
2608/tcp open  quagga  Quagga routing software 0.99.22.4 (Derivat
68/udp   open|filtered dhcpc
520/udp  open|filtered route
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.5
Network Distance: 1 hop
```

图 8 第 2 次扫描探测结果

那么, 通过增加扫描的次数和频度, 是否能够确定目标系统的准确信息呢? 为此, 对目标系统做进一步扫描探测。具体方法为: 进行 10 次扫描, 每次扫描的开始时间是随机选定的, 从扫描开始一直到返回扫描结果视为一次扫描。在扫描过程中目标执行体采用前文描述的随机调度方法进行调度切换。由于这 10 次得到的结果基本类似, 选择其中一次扫描结果进行分析。扫描结果如图 9 所示, 这次扫描持续了 1094 秒, 通过开放的端口号和服务来确定这是一台路由设备, 确定操作系统版本为 Linux2.4.1。通过查看目标系统的调度日志得知, 本次扫描过程中共有四个执行体轮流上线: Cisco 执行体在线 239 秒, Juniper 执行体在线 215 秒, Quagga 执行体在线 289 秒, 中兴执行体在线 291 秒。在此扫描期间, Nmap 在使用 TCP 扫描手段时, Cisco 和 Juniper 执行体依次上线, 使用 UDP 扫描手段时, Quagga 和中兴执行体依次上线。因此, Nmap 扫描得到的最后的测试结果实际上是对不同目标进行扫描探测得到结果, 基于这些结果做出的测试结论必然是不准确的。汇总这 10 次扫描结果, 每次得到的结果都是不一致的, 也就是说扫描测试结果具有不可预测性。

```
PORT      STATE         SERVICE         VERSION
23/tcp    open          tcpwrapped
179/tcp    open          tcpwrapped
|_finger: ERROR: Script execution failed (use -d to debug)
502/udp   open|filtered mbap
520/udp   open|filtered route
686/udp   open|filtered hcp-wismar
1039/udp  open|filtered sbl
1234/udp  open|filtered search-agent
6001/udp  open|filtered X11:1
|_x11-access: ERROR: Script execution failed (use -d to debug)
7000/udp  open|filtered afs3-fileserver
Device type: general purpose
Running: Linux 2.4.X
OS CPE: cpe:/o:linux:linux_kernel:2.4.21
OS details: Linux 2.4.21
Network Distance: 1 hop
Nmap done: 1 IP address (1 host up) scanned in 1094.98 seconds
```

图 9 第 3 次扫描探测结果

扫描探测是攻击者实施攻击第一步, 准确的获知目标系统的相关信息, 将决定了下一步攻击所采用的方法与手段。在不清楚目标系统信息的情况下攻击成功的难度和代价将会非常之大。DHR2 验证系统同时运行多个异构冗余的工作执行体, 这些执行体通过随机调度机制对外呈现。分析实验测试结果, 可以看出, 这种随机调度机制, 使得系统对外呈现的信息发生了跳变, 仅仅凭借一两次扫描探测难以准确识别目标系统的相关信息。为了获得准确的目标信息, 攻击者将不得不增加扫描探测的次数和频度, 这将明显增加攻击成本。这些恶意行为使得攻击

者更容易被入侵检测系统检测到。而且上述实验表明，即使增加了扫描探测的次数和频度，也不一定能够获得目标准确的系统信息。所以，在扫描探测阶段，DHR2 系统通过动态性、异构性和冗余性，可以有效地迷惑攻击者，明显降低攻击者获取目标系统信息的准确度，增加扫描探测花费的时间，显著提升了攻击者获取目标系统信息的难度。

5.2 系统漏洞扫描

上节测试过程中，通过 Nmap 扫描探测并未确定目标系统的具体信息，但是可以基本明确系统开放了哪些端口，运行了哪些服务协议，例如 Telnet、SNMP、OSPF、BGP、RIP 等。本节在上节扫描探测的基础上，对这些开放的服务做进一步的漏洞扫描，试图发现这些服务存在的漏洞信息。

采用 Nessus[41]工具对目标系统的几个服务进行漏洞扫描，共进行 10 次扫描，仅 1 次发现了目标系统存在高危脆弱点(CVSS 基本评分 7.5)，结果如图 10 所示。该高危脆弱点为 SNMP 协议的团体字。这是因为 DHR2 系统中有一个执行体使用了 public 团体字。由于系统动态性的存在，其他执行体并没有采用这种易猜的团体字，所以，在 10 次扫描过程中仅有 1 次发现目标系统存在该脆弱点。

101.95.19.210

Summary

Critical	High	Medium	Low
0	1	0	1

Details

Severity	Plugin Id	Name
High (7.5)	41028	SNMP Agent Default Community Name (public)
Low (2.6)	42263	Unencrypted Telnet Server
Info	10114	ICMP Timestamp Request Remote Date Disclosure
Info	10281	Telnet Server Detection
Info	10287	Traceroute Information
Info	10551	SNMP Request Network Interfaces Enumeration
Info	10800	SNMP Query System Information Disclosure
Info	11219	Nessus SYN scanner
Info	11936	OS Identification
Info	12053	Host Fully Qualified Domain Name (FQDN) Resolution
Info	14274	Nessus SNMP Scanner
Info	19506	Nessus Scan Information
Info	22964	Service Detection
Info	34022	SNMP Query Routing Information Disclosure
Info	35296	SNMP Protocol Version Detection
Info	35716	Ethernet Card Manufacturer Detection

图 10　漏洞扫描结果

漏洞扫描是攻击者实施攻击第二步，该阶段的结果将直接决定了后续攻击手段、工具和攻击的难易程度。如果扫描到目标系统存在已知的漏洞，那么攻击成功的概率将会大增。DHR2 验证系统通过随机调度机制，多个执行体以跳变的形式对外呈现，使得漏洞扫描工具面向变化的目标，很难在短时间内准确得到目标系统的漏洞信息。因此，动态异构冗余机制可以在不消除系统漏洞或脆弱点前提下改变漏洞或系统脆弱点的呈现特性。

5.3 系统漏洞利用

本节假定攻击者已经获知目标系统存在一个可利用漏洞，通过实验方法测试评估目标系统在引入了动态异构冗余机制后，该漏洞的利用难度。

```
msf auxiliary(cisco_config_tftp) > exploit
[*] Starting TFTP server...
[*] Scanning for vulnerable targets...
[*] Trying to acquire configuration from 1.1.1.66...
[*] Scanned 1 of 1 hosts (100% complete)
[*] Providing some time for transfers to complete...
[*] Incoming file from 1.1.1.66 - 1.1.1.66.txt 860 bytes
[*] Saved configuration file to /home/1.1.1.66.txt
[+] 1.1.1.66:161 Unencrypted Enable Password: !qaz@wsx
[*] Collecting :!qaz@wsx
[+] 1.1.1.66:161 Username 'nmgdcy' with Password: n@Qos$mpls
[*] Collecting nmgdcy:n@Qos$mpls
[+] 1.1.1.66:161 SNMP Community (RO): nmgdcy@demaxiya
[*] Collecting :nmgdcy@demaxiya
[+] 1.1.1.66:161 SNMP Community (RW): public
[*] Collecting :public
[*] Shutting down the TFTP service...
[*] Auxiliary module execution completed
```

图 11　漏洞利用结果

DHR2 验证系统内嵌的一个工作执行体就存在一个 SNMP 漏洞[42]，该执行体为思科 IOS c2600-i-mz.120-3.T3.bin。为了验证该漏洞可被利用，暂停了 DHR2 系统的动态调度机制，并设定思科执行体为对外呈现的工作执行体，利用 Metasploit[43]工具对其进行渗透测试，得到结果如图 11 所示，可以看到该漏洞可被成功利用，利用该漏洞可以获得目标系统的 enable 口令，以及相应的系统配置文件。

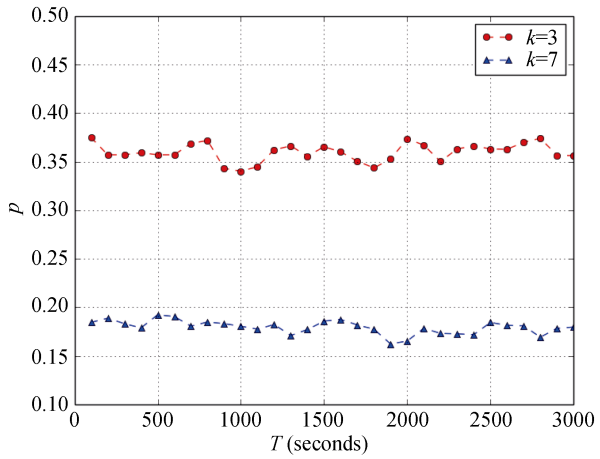

图 12　在给定调度时间间隔和执行体数量 *k* 条件下，漏洞利用成功率 *p* 随攻击间隔 *T* 的变化趋势

下面启动 DHR2 系统的动态调度机制，并设计如下测试方法：在测试 0 点，利用 Metasploit 进行一次渗透测试，返回测试结果即视为本次测试结束，记录成功与否。然后，延迟时间 *T* 后再进行一次渗透

测试, 如此循环, 持续 3 小时。记录渗透测试的次数 n 和能够获取到目标系统管理员口令的次数 m, 通过 $p=m/n$ 的值来反映漏洞利用的成功率。最后, 改变 DHR2 系统内置执行体数量 k, 重复上述实验, 得到统计结果如图 12 所示。从统计数据可以看到, 在引入动态机制后, 系统存在的漏洞并不是每次都能被成功利用。随着攻击时间间隔的增长, 观测窗口内发起攻击的次数减少, 漏洞利用成功的次数也相应线性减少, 漏洞利用成功率保持在一定的比值范围。实施攻击的时间间隔对漏洞利用成功率的影响并不明显, 但是系统内置的异构执行体的数量对漏洞利用成功率的影响较大, 在内置 7 个执行体的情况下, 平均的漏洞利用成功率在 20%以下。

下面分析执行体调度时间间隔对漏洞利用成功率的影响。假定攻击时间间隔 T=100s, 调度执行体的切换时间为[$0.4S$,S]之间的一个随机值。重复上述实验, 得到统计结果如图 13 所示。从统计数据来看, 随着调度时间间隔的增加, 漏洞利用成功的比率呈线性缓慢的增长。这是因为, 随着调度时间间隔的增长, 系统的动态性降低, 存在漏洞的执行体对外暴露的时间就越长, 观测窗内该执行体的漏洞被利用成功的可能性就越大。同时, 从图中可以看出, 内置执行体的数量多少并没有影响漏洞利用成功率的改变趋势。但是相比较而言, 相同条件下, 执行体数量越多, 漏洞利用成功的比率越低。

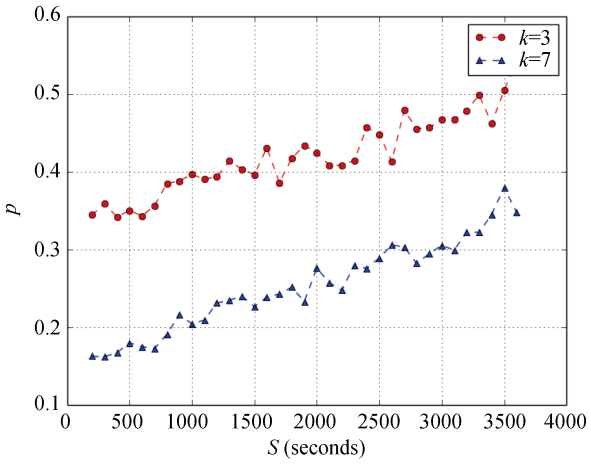

图 13　在给定攻击时间间隔和执行体数量 k 条件下, 漏洞利用成功率 p 随调度时间间隔 S 的变化趋势

通过上述实验可以得知, DHR2 系统中内置的执行体数量与漏洞利用成功的比率存在密切的关系; 系统的调度时间间隔与漏洞利用成功的比率也存在密切的关系。在执行体数量和调度时间间隔的双重作用下, 系统漏洞外呈现的时间窗口变的更小, 更随机, 使得漏洞被成功利用的可能性大大减少。另外, 漏洞利用的成功率与实施攻击的时间间隔之间没有关联性, 这就意味着攻击者通过增加攻击的时间窗口和频次, 并不能提高其攻击成功的概率。因此, DHR 机制可以明显提升系统漏洞被利用的难度。

5.4　系统后门持续性利用

本节假定攻击者在目标系统预置后门的情况下, 评估在引入了动态异构冗余机制后, 该后门被持续利用的难度。为此, 在系统内部一个执行体上进行了后门设置: 如果收到的 OSPF hello 报文中 Neighbor 域含有 0x5a5a5a02 字段, 则后门被触发, 执行体自动增加一条到 A.B.C.D 的路由, 其中 A.B.C.D 由 hello 报文中的 Designated Router 字段指定。

设计测试拓扑如图 14 所示, 测试方法为: 主机 A 通过 Tcpreplay[44]持续发送目的地为主机 B 的流量, 同时在主机 B 和主机 C 通过 Wireshark[45]观测是否接收到该流量。在测试 0 点, 主机 A 向 DHR2 系统发送后门触发包, 其中的 Designated Router 字段设置为 C 的地址。每隔 15s 发送一次后门触发包, 持续一小时, 调度执行体的切换时间为[$0.4S$,S]之间的一个随机值。记录后门触发包的发送次数 n 和后门被触发的次数 m, 通过 $p=m/n$ 的值来反映后门被成功触发的比率。

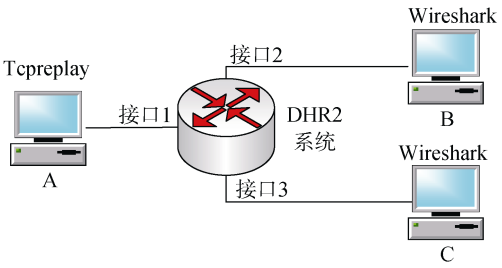

图 14　后门触发测试的拓扑场景

实验得到统计结果如图 15 所示。从统计数据来看, 随着调度时间间隔的增长, 后门被触发的概率呈非线性趋势急剧下降。执行体数量越多, 后门被触发的概率越低。在 3 个工作执行体情况下, 后门被触发的比率在 4%以下。与漏洞利用成功率相比, 后门被触发的比率更低, 这是因为漏洞被利用的过程没有被多模表决检测到, 即使利用成功了, 下次仍然可以利用成功, 只要存在漏洞的执行体一直在线。但是对于后门触发而言则不同, 后门触发后的操作被多模表决检测到后, 就立刻将其下线清洗, 并调度其他执行体上线工作, 因此, 后续后门触发包将无法触发后门, 直到后门执行体再次被调度上线。

通过记录存在后门的执行体的累计工作时间与调度时间间隔的数据, 得到统计分布如图 16 所示。

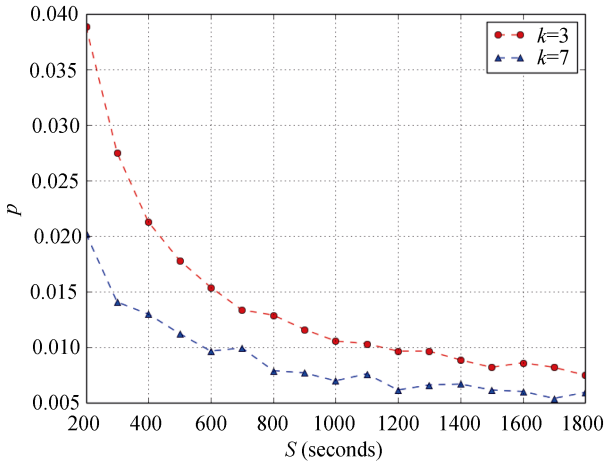

图 15　在给定攻击时间间隔和执行体数量 *k* 条件下, 后门触发成功比率 *p* 随调度时间间隔 *S* 的变化趋势

执行体数量越多, 后门执行体累计工作时间越短。在设定的观测窗内, 当系统内置 7 个工作执行体情况下, 平均每一个执行体的累计在线时间为 514 秒, 而后门执行体的累计在线时间最长为 50 秒, 这意味着安全威胁越大的执行体可信度越低, 被使用的时间越短。产生上述现象的原因是 DHR 机制的动态调度和多模表决机制决定的。后门触发条件仅仅能够触发一个执行体的后门, 而对其他执行体不起作用, 当后门触发后, 通过比对路由表变化, 检测出仅后门执行体的路由表发生了改变, 由此判决该执行体发生异常, 立刻将其下线, 调度其他执行体上线。同时, 降低异常执行体的可信度, 减小下一次被调度上线的机率。在这种机制下, 当后门执行体的后门没有触发时, 后门执行体正常在线工作; 一旦后门被触发, 多模表决就可以检测出来, 并通过调度机制, 将其下线清洗, 这就解释了后门执行体累计工作时间很短的原因。

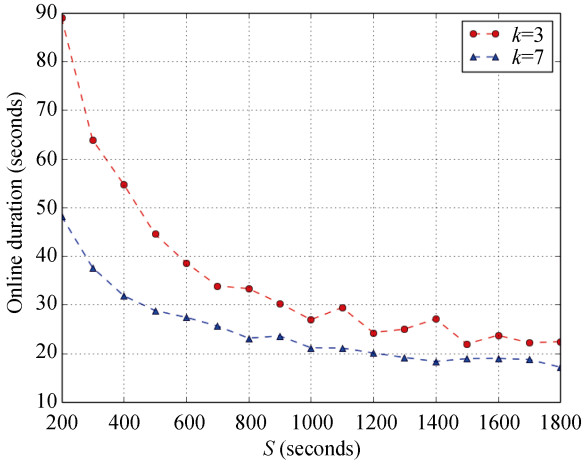

图 16　在给定攻击时间间隔和执行体数量 *k* 条件下, 后门执行体累计在线时长随调度时间间隔 *S* 的变化趋势

在上述测试过程中, 主机 C 一直未收到目的为主机 B 的流量。即使工作执行体的后门被触发、路由表被修改, 但多数执行体的路由表没有发生变化, 通过多模表决机制, 不会触发 DHR2 系统修改转发表。所以, 在整个测试过程中流量并没有被改向。

通过上面实验可以看出, 路由器在引入 DHR 机制后, 多模表决机制可以使得产生异常的执行体工作时间更短, 使得系统脆弱点对外暴露的机率更小。而且, 合理的设置多模表决点, 将使得后门在触发的情况下, 根本无法工作。多模表决机制通过比对输出结果实现异常的检测, 因此, 根本不需要知道系统的漏洞/后门是什么, 所以说 DHR 机制允许系统内部存在漏洞/后门, 并可以实现对未知漏洞/后门的有效防御。

5.5　路由表生效时间

本节分析 DHR2 系统引入动态异构冗余机制后带来的性能指标变化。DHR2 系统中的多模表决需要以多个执行体生成的路由为输入, 进行路由表的多模表决, 这势必对最终路由表的生效时间产生影响。

测试方法为通过测试仪利用 BGP 协议向目标系统分别注入 500,1000,1500 条路由, 分别在单执行体和多执行体环境下测量目标系统的路由表生效时间, 得到结果如表 1 所示。

表 1　路由表生效时间对比结果

测试条件	单执行体(ms)	三执行体(ms)	五执行体(ms)
500 条路由	28.0	32.0	32.2
1000 条路由	30.2	34.8	35.2
1500 条路由	45.0	52.0	52.8

测试结果表明, 路由器在引入 DHR 机制后, 路由表的生效时间比单执行体环境下的路由表生效时间长, 但没有大幅增长。这是因为 DHR 机制中的路由表多模表决单元要等待多个执行体计算得到路由表后, 才能进行多模表决。也就是说引入多模表决后, 路由表的生效时间最小值为所有执行体的路由表生效时间的最大值。同时, 由于所有执行体的路由表生效时间相差不大, 因此, 执行体的数量对最终的路由表生效时间的影响甚微。

6　结论

本文以邬江兴院士提出的拟态防御机制为指导, 提出了一种路由器的动态异构冗余实现架构, 并给出了该架构的验证系统实现方法。通过实验测试, 说明路由器在引入 DHR 机制后, 提升了攻击者扫描探测的难度, 使得系统漏洞/后门更难被锁定, 使得系

统漏洞更难被利用、后门更难被触发, 使得系统后门即使被触发也无法工作。路由器引入 DHR 机制后, 将容许系统在存在漏洞/后门的条件下正常工作, 而且该防御方法是由DHR机制决定, 与漏洞/后门特性正交, 因此, 对未知的漏洞/后门也有同样的防御效果。DHR 机制的引入, 不影响路由器的功能, 也没有明显降低系统的性能。目前验证系统集成度差, 执行体的虚拟化部分有待进一步研发, 感知决策部分还需要进一步研究基于统计的异常检测机制。

参考文献

[1] Z Shao and L. Jiao, “prism doors Refracting the developing direction of industrial software,” China Information: e Manufacturing, 2013, 11: 24-33
(邵泽宇,皎丽丽. “‘棱镜门’折射我国工业软件何去何从,” 2013, 11:24-33.)

[2] “Cisco CEO sent a letter to Obama asked the NSA to contain monitor activities,” http://tech.qq.com/a/20140520/000986.htm, 2014. 5
(思科CEO致信奥巴马要求遏制国安局监控活动, http://tech.qq.com/a/20140520/000986.htm, 2014. 5)

[3] “About the presence of a variety of preset router backdoor vulnerability briefing,” http://www.cert.org.cn/publish/main/9/2014/20140429121938383684464/20140429121938383684464_.html, 2016.9
(关于多款路由器设备存在预置后门漏洞的情况通报, http://www.cert.org.cn/publish/main/9/2014/20140429121938383684464/ 20140429121938383684464_.html, 2016.9)

[4] A.V. Andrew, V. G. Konstantin and N.V. Janis, “Exposing Cisco Network Hackinng,” Tsinghua University Press, 2008
(A.V. Andrew, V. G. Konstantin and N.V. Janis, “思科网络黑客大曝光”, 清华大学出版社, 2008)

[5] Y. Zhang, S. Dao, H. Vin, L. Alvisi, and W. Lee. “Heterogeneous Networking: A New Survivability Paradigm,” *Workshop on New Security Paradigms* (2001):33-39.

[6] C. Byung-Gon, M. Petros, S. Scott, “Diverse replication for single-machine Byzantine-fault tolerance,” *Usenix Technical Conference*, Boston, Ma, Usa, Jun. 2008:287-292.

[7] J. Caballero. T. Kampouris, D. Song, J. Wang, “Would Diversity Really Increase the Robustness of the Routing Infrastructure against Software Defects? ,” *Network and Distributed System Security Symposium*, NDSS 2008, San Diego, California, Usa, Feb. 2008.

[8] E. Keller, M. Yu, M. Caesar, J. Rexford, “Virtually eliminating router bugs,” *ACM Conference on Emerging NETWORKING Experiments and Technology*, CONEXT 2009, Rome, Italy, December 2009:13-24.

[9] S Jajodia, AK Ghosh, V Swarup, C Wang, XS Wang, “Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats,” Springer Ebooks (2011).

[10] M. Sifalakis, S. Schmid, and D. Hutchison. “Network address hopping: a mechanism to enhance data protection for packet communications,” *IEEE International Conference on Communications IEEE*, 2005:1518-1523 Vol. 3.

[11] M. Dunlop, S. Groat,W. Urbanski, R. Marchany, and J. Tront, “MT6D: A Moving Target IPv6 Defense,” *Military Communications Conference*, 2011 - MILCOM 2011 IEEE, 2011: 1321-1326.

[12] J. J. Haadi, E. Al-Shaer, and Q. Duan. “Openflow random host mutation: transparent moving target defense using software defined networking,” *The Workshop on Hot Topics in Software Defined Networks ACM*, 2012:127-132.

[13] “SYNful Knock - A Cisco Router Implant – Part I”. FireEye Inc. https://www.fireeye.com/blog/threat-research/2015/09/synful_ knock_-_acis.html, 2015.9

[14] Y. Rekhter, and T. Li. “A Border Gateway Protocol 4 (BGP-4),” RFC4271, 2006.

[15] J. Moy, “OSPF Version 2,” RFC2328, 1998.

[16] G. Malkin, “RIP Version 2,” RFC2453, 1998

[17] J. Wu, “mimic securiy defence of cyberspace,” *Secrecy science and technology*, 2014(10)
(邬江兴, “网络空间拟态安全防御,” 保密科学技术, 2014(10))

[18] J. Wu, “Meaning and vision of mimic computing and mimic security defense,” *Telecommunications science*, 2014, 30(7): 1-7
(邬江兴, “拟态计算与拟态安全防御的原意和愿景,” *电信科学*, 2014, 30(7): 1-7)

[19] J. Wu, “The key rebalancing strategy of network security – Mimic defense,” http://blog.sina.com.cn/s/blog_5946bd590102wi07.html, 2016.8
(邬江兴, “网络安全“再平衡战略”抓手--拟态防御,” http://blog.sina.com.cn/s/blog_5946bd590102wi07.html, 2016.8)

[20] B.-G. Chun, P. Maniatis, and S. Shenker. “Diverse replication for single-machine byzantine-fault tolerance,” *Usenix Technical Conference*, Boston, Ma, Usa, 2008:287-292.

[21] F. Junqueira, R. Bhgwan, A. Hevia, K. Marzullo, and G. Voelker. “Surviving Internet catastrophes,” *Usenix Technical Conference*, Anaheim, Ca, Usa 2005:45-60.

[22] Y. Zhang, S. Dao, H. Vin, L. Alvisi, and W. Lee. “Heterogeneous networking: A new survivability paradigm,” *Workshop on New Security Paradigms* (2001): 33-39.

[23] Xorp, inc. http://xorp.org, 2016.9

[24] “Quagga software routing suite”. www.quagga.net, 2016.9

[25] “The BIRD Internet Routing Daemon”. http://bird.network.cz. 2016.9

[26] E. Keller, M. Yu, M. Caesar, and J. Rexford. “Virtually Eliminating Router Bugs,” *ACM Conference on Emerging NETWORKING Experiments and Technology*, CONEXT 2009, Rome, Italy, Dec. 2009:13-24.

[27] J. Knight and N. Leveson. “A reply to the criticisms of the Knight & Leveson experiment,” *ACM SIGSOFT Software Engineering Notes*, 15.1(1990): 24-35.

[28] “Cisco 7200 simulator. (software to run Cisco IOS images on desktop PCs)”, http://www.ipflow.utc.fr/index.php/Cisco_

7200_Simulator, 2016.9

[29] “Packet Tracer”, http://www.cisco.com/web/learning/netacad/course_catalog/PacketTracer.html, 2016.09

[30] “Olive. (software to run Juniper OS images on desktop PCs)”, http://juniper.cluepon.net/index.php/Olive, 2016.9

[31] “eNSP: Enterprise Network Simulation Platform”, http:// support. huawei.com/enterprise/toolNewInfoAction!toToolDetail?contentId=TL1000000015&productLineId=7919710, 2016.09

[32] GNS3, GNS3 Technologies Inc. https://www.gns3.com/

[33] “QEMU”, http://wiki.qemu.org/, 2016.09

[34] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, “Ethane: Taking Control of the Enterprise,” *ACM Sigcomm Computer Communication Review* 37.4(2007): 1-12.

[35] P. Goransson, and C. Black. “The OpenFlow Specification. Software Defined Networks,” 2014:81-118.

[36] “Kernel Virtual Machine”, http://www.linux-kv m.org/, 2016.09

[37] H. Ma, J. Wu, and H. Chen, “Routing protocol parallel multi-instance execution system and parallel execution method,” 201510436410.5, 2015.7
(马海龙, 邬江兴, 陈鸿昶等, “路由协议多实例并行执行系统及其并行执行体方法,” 201510436410.5, 2015.7)

[38] J. Shen, P. Zhang, and H. Ma, “state pool based dynamic running method of routing protocol,” 201510580553.3, 2015.09
(申涓, 张鹏, 马海龙, “一种基于状态池的路由协议构件动态运行方法,” 201510580553.3, 2015.09)

[39] M. R. Nascimento, C. E. Rothenberg, and M. R. Salvador, “Virtual routers as a service: the RouteFlow approach leveraging software-defined networks,” *International Conference on Future Internet Technologies ACM*, 2011:34-37.

[40] G. Lyon. “Nmap security scanner,” http://namp.org, 2016.8

[41] “Nessus. Tenable Network Security”. http://www.tenable.com/products/nessus-vulnerability-scanner. 2016.8

[42] “Cisco IOS 11.x/12.0 - ILMI SNMP Community String”. https://www.exploit-db.com/exploits/20652. 2016.8

[43] Metasploit. https://www.metasploit.com. 2016.8

[44] “tcpreplay-lite tool,” https://sourceforge.net/projects/tcpreplay/, 2016.09.

[45] “Wireshark”, https://www.wireshark.org/, 2016.09

马海龙 于 2011 年在信息工程大学通信与信息系统专业获得博士学位。现任信息工程大学信息技术研究所副研究员。研究领域为网络安全、路由工程。研究兴趣包括: 创新网络体系、网络安全管控等。Email: longmanclear@163.com

江逸茗 于 2014 年在信息工程大学通信与信息系统专业获得博士学位。现任信息工程大学信息技术研究所助理研究员。研究领域为新型网络体系结构、网络空间安全防御。研究兴趣包括: 云计算、虚拟化等技术。Email: j8403@163.com

伊鹏 于 2007 年在信息工程大学通信与信息系统专业获得博士学位。现任信息工程大学技术研究所研究员。研究领域为新型网络体系结构、网络安全管控和主动防御技术研究。Email: 15238363586@139.com

贺磊 于2008年在信息工程大学计算机软件专业获得博士学位。现任信息工程大学信息技术研究所副研究员。研究领域为网络安全。研究兴趣包括网络空间安全防御、大数据处理等。Email: hl.helei@163.com