

软件定义网络下的拟态防御实现架构

王禛鹏¹, 扈红超¹, 程国振¹, 张传浩^{1,2}

(1. 国家数字交换系统工程技术研究中心, 河南 郑州 450003;

2. 铁道警察学院公安技术系, 河南 郑州 450053)

摘要: 针对传统防御技术难以应对未知漏洞和后门的问题, 拟态安全防护 (MSD, mimic security defense) 通过构造动态异构冗余模型, 提高系统的不确定性, 增加攻击者的攻击难度和成本, 提升网络安全性能。基于软件定义网络, 提出了一种拟态防御的实现架构, 首先, 按照非相似冗余度准则构建异构冗余执行体, 而后借助软件定义网络的集中管理控制实现动态选调和多模判决等功能。实验验证了架构的入侵容忍能力和可用性。

关键词: 拟态安全防护; 软件定义网络; 主动防御; 动态异构冗余

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2017.00205

Implementation architecture of mimic security defense based on SDN

WANG Zhen-peng¹, HU Hong-chao¹, CHENG Guo-zhen¹, ZHANG Chuan-hao^{1,2}

(1. National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450003, China;

2. Public Security Technology Department, Railway Police College, Zhengzhou 450053, China)

Abstract: To deal with the attacks employing unknown security vulnerabilities or backdoors which are difficult for traditional defense techniques to eliminate, mimic security defense (MSD) that employs “dynamic, heterogeneity, redundancy (DHR)” mechanism can increase the difficulty and cost of attack and uncertainty of system so as to improve network security. Based on the software defined networking (SDN), an implementation architecture of MSD was proposed. First, diverse functional equivalent variants for the protected target were constructed, then leverage the rich programmability and flexibility of SDN to realize the dynamic scheduling and decision-making functions on SDN controller. Simulation and experimental results prove the availability and the intrusion tolerant ability of the architecture.

Key words: mimic security defense, software defined networking, active defense, dynamic heterogeneous redundancy

1 引言

网络技术已广泛应用于电力、金融、交通等

关键基础设施, 在提供便捷的同时, 其安全问题也为人类带来了极大的安全威胁。相较于防御技术, 网络攻击呈现愈演愈烈的态势, 究其原因, 主

收稿日期: 2017-08-13; 修回日期: 2017-09-24。通信作者: 扈红超, 13633833568@139.com

基金项目: 国家自然科学基金资助项目 (No.61309020, No.61602509); 国家自然科学基金创新群体基金资助项目 (No.61521003); 国家重点研发计划基金资助项目 (No.2016YFB0800100, No.2016YFB0800101); 河南省科技攻关基金资助项目 (No.172102210615, No.172102210441)

Foundation Items: The National Natural Science Foundation of China (No.61309020, No.61602509); The Foundation for Innovative Research Groups of the National Natural Science Foundation of China (No.61521003), The National Key Research and Development Program of China (No.2016YFB0800100, No.2016YFB0800101), The Key Technologies Research and Development Program of Henan Province of China (No.172102210615, No.172102210441)

要有以下几点。1) 现有的网络空间组成要素呈现的单一性, 根据 Net Market Share 公布的数据^{注1}, 桌面操作系统的市场份额中, Windows 占 91.39%; 搜索引擎方面, Google 占 79.45%; 硬件方面, 由 Synergy Research 公布的 2016 年的数据显示, 思科占全球交换路由器市场 53% 的份额^{注2}。这些都导致攻击者极易利用单个安全缺陷, 对网络服务可用性等造成多次、大规模的损害^[1,2]。2) 网络元素、结构和配置呈现的静态性, 如使用静态 IP、静态端口等, 这样攻击者很容易对信息系统进行持续的探测和入侵。3) 现有的防御技术较为被动, 如入侵检测^[3]、蜜罐^[4]、沙箱^[5]等, 属于“亡羊补牢式”的防护手段, 这种先瞄准(发现攻击特征)后打击的防御手段, 显然无法满足某些网络关键基础设施的安全需求, 并对未知的漏洞威胁束手无策。

通过动态化“改变游戏规则”的思想解决上述问题已广泛应用。其中最为典型的是美国提出的移动目标防御(MTD, moving target defense)^[6], 其核心思想是通过为系统引入动态性和随机性, 增加系统的不可预测性, 从而阻断攻击者的攻击过程, 提高攻击者的攻击难度和攻击成本。而邬江兴院士^[7]提出的创新思想拟态安全防御除了具备上述的内生安全防护性, 还兼有检测功能。拟态防御的核心思想是动态异构冗余机制, 首先构建异构冗余的执行空间, 然后在系统运行期间, 从执行空间中动态地调度若干执行体提供服务, 并采用多模判决机制对执行体的输出结果进行一致性判决, 检测并定位攻击行为。

为实现拟态防御的动态调度和多模判决等机制, 本文提出了一种基于软件定义网络^[8]的实现架构。SDN 是一种数控分离的新型网络体系结构, 数据平面只负责基本的数据转发, 而将网络管理功能集中于 SDN 控制平面。这种集中管理控制的架构, 可以实现网络流量的动态控制, 为拟态防御思想的实现提供了有利条件。本文将选调、判决等机制作为应用(APP), 设在 SDN 控制器

上, 通过下发流表项实现动态调度等功能, 增强网络安全性能。

2 相关工作

本节主要介绍防御技术的相关工作, 防御技术主要介绍 MTD^[6]和 N 变体系统^[9]这 2 种防御思想, 然后简要介绍利用 SDN 进行流量管理控制的相关研究。

MTD 由美国网络与信息技术研究与发展计划(NITRD, federal networking and information technology research and development)^[6]提出, 旨在通过引入多样性和动态性, 增加系统对外呈现的不确定性。根据应用的场景不同, MTD 技术可分为基于终端级和基于网络级。基于终端级的, 如指令随机化(ISR, instruction set randomization)技术^[10]、代码随机化(code randomization)技术^[11], 以及地址空间随机化(ASLR, address space layout randomization)技术^[12]等。基于网络级的移动目标防御技术主要是将网络设备、功能的网络配置动态化。例如, Dunlop 等^[13]提出的 MT6D (moving target IPv6 defense), 基于 IPv6 巨大的地址空间, 动态轮转网络和传输层的地址(IP 跳变), 使攻击者难以锁定目标攻击。Jafarian 等^[14]提出了 RHM, 采用基于地址空间和 IP 地址的分层跳转方案, 后又在文献[15]中提出采用主动自适应地址跳变技术, 该技术通过监测攻击行为, 自动执行地址跳转。

N 变体系统^[9]的核心思想是采用多个异构体执行相同的输入, 通过检测其输出结果的一致性判断系统是否遭受攻击, 其原理是一种攻击方法无法同时对多个变体有效, 且造成相同的错误。Mayo 等^[16]提出的异构冗余模型采用了同样的思想, 并通过投票机制(大数判决)裁决最终的有效结果和异常变体, 实现入侵容忍和攻击检测。

拟态安全防御的动态异构冗余模型则兼有上述 MTD 的动态性和 N 变体的异构冗余特性, 其一般模型如图 1 所示。其中 $\{E_1, E_2, \dots, E_N\}$ 为 N 个异构执行体空间, 调度器动态地从中选取 M 个 $\{A_1, A_2, \dots, A_M\}$ 提供服务, 对输入进行处理, 由判决器对多个结果进行裁决, 输出最可信的结果,

注1: <http://www.netmarketshare.com/>

注2: <https://www.srgresearch.com>

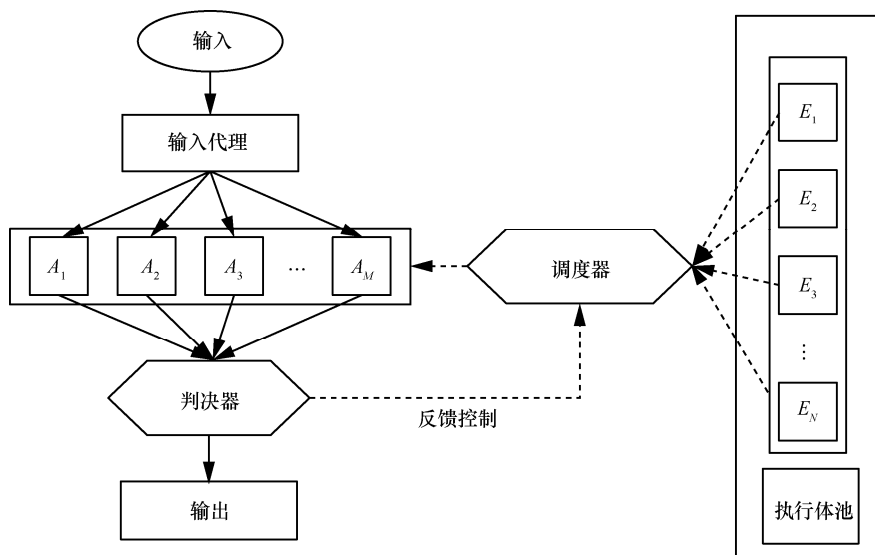


图1 动态异构冗余模型

定位输出异常的执行体，并通告调度器，降低其选取概率权值或直接予以清洗，实现反馈控制。对于动态异构冗余模型，若取消其动态性，则拟态防御模型退化为 N 变体模型；若将上述 M 取值设为 1，并取消判决机制，则退化为移动目标防御模型。因此，上述 2 种防御思想某种程度上可视为拟态防御的特例。

目前，基于拟态防御思想进行安全防护的研究日益增加^[17,18]，然而，拟态较为复杂的安全机制设计，使其在实际场景中不如上述 2 个防御思想应用广泛。针对这一问题，本文提出基于软件定义网络技术，实现拟态防御的简易部署安全架构。

SDN 集中管理控制的优势为网络流量动态管理提供了有利条件，如应用最为广泛的流量均衡^[19]。文献[20]通过引导流量至新规划的路径利用 SDN 实现链路快速恢复，增加网络健壮性。文献[21]将流量分多路径进行分段传输，增加攻击者截获重组数据的难度，从而增加数据的可靠性。文献[22]借助 SDN 实现动态多蜜罐技术。文献[23]基于 SDN 网络，实现透明无感的移动目标防御。文献[24]基于 SDN，考虑了网络功能的状态信息，实现网络功能的动态迁移。

本文利用软件定义网络集中管理控制的优势，提出一种可应用于多种防御场景、易部署、非侵入的拟态防御实现方法和架构。

3 基于软件定义网络的拟态防御架构

基于软件定义网络的拟态防御整体架构如图 2 所示。首先，本文利用网络功能虚拟化等技术构建异构体空间，这种硬件资源虚拟化和软硬解耦的设计有利于对异构体空间进行拓展、清洗等管理操作。其次，与图 1 对应可以看出，本文将输入代理（负责分流）交由 SDN 数据平面完成，而将调度、判决等控制相关功能置于 SDN 控制器上，而控制平面上的功能模块分为基本模块和拓展模块，实现集中管理控制。

首先，通过这种架构设计，可以看出执行体池中的执行体根据应用场景部署为需要进行保护的信息系统，如域名系统（DNS, domain name system）服务器或 Web 服务器等，得益于 SDN 提供的可定义的开放编程接口，对于具体不同的应用场景和网络状态，只需修改 SDN 控制器上运行的应用中相关代码实现调整部署。其次，这种冗余设计本身无需对原有的信息系统以及客户端做任何修改：从客户端的角度上看，其不会感知到冗余架构的信息系统；从每个执行体（信息系统）的角度上看，其不会感知到其余冗余执行体以及 SDN（控制器）的操作处理（选调、裁决等），即每个执行体都独立运行，相互之间不会感知和影响，实现了透明无感，是一种非侵入、易部署的设计。

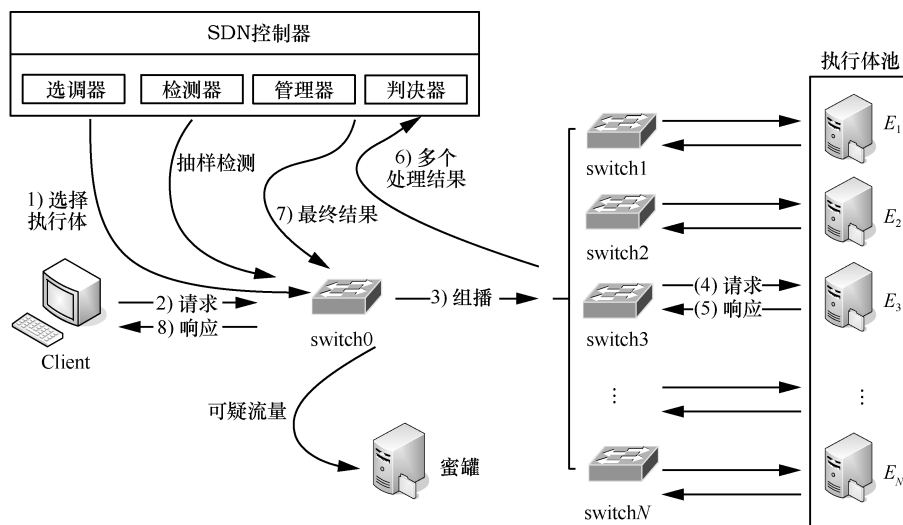


图 2 基于 SDN 的拟态防御实现架构

本节首先对架构的各个模块进行介绍，然后分析架构本身的安全性。

3.1 基本模块

作为拟态防御最核心的功能模块，基本模块主要包含选调器和判决器，负责异构执行体的动态调度和输出结果的多模判决，下面分别进行介绍。

3.1.1 选调器

选调器主要工作为动态调度和控制交换机（图 1 中 switch0）进行分流。选调器主要通过下发 OFPT_FLOW_MOD（OpenFlow 流表修改消息）实现网络流量分流。例如，某时刻执行体 1~3 ($A_1 \sim A_3$) 被选取出，则向 switch0 下发的 OFPT_FLOW_MOD 消息中 2 个关键的 match（匹配字段）和 actions（行为字段）设置如程序 1 所示（本文采用 Ryu 控制器），当 switch0 安装相应流表项后，switch0 就会将匹配的数据分组（程序 1 中只采用 in_port（入端口）作为匹配字段，实际中可增用 IP 地址等）复制 3 份，分别发往 switch1~switch3，最终交由 $A_1 \sim A_3$ 进行处理。

程序 1

```

1) function output(x)
2)   output to x // output 的功能是将网络数据分组转发给交换机的端口 x
3) end function
4) match = (in_port == from_client) //流表规则中的匹配字段，匹配内容为：从客户方向进入交换机端口的网络请求

```

5) actions = [output(switch1), output(switch2), output(switch3)] //流表规则中的行为字段，操作为：将网络请求分别分发给交换机 switch1, switch2, switch3

6) if match == True //如果满足上述匹配内容，则执行如下操作

7) then actions //将网络请求发往设定的交换机 switch1, switch2, switch3

8) end if

选调器的调度方法一般借助判决器实现反馈控制。例如，某执行体出现异常（输出结果与大多数结果不一致或者拓展模块检测出异常等），则选调器可以降低选取它的概率等。具体的策略则可根据实际应用场景进行定制设置。例如，对于某些网络核心服务，安全需求较高时，当发现执行体异常，可以不再选取（概率为 0），清洗后方可使用；而其他一些场景，则可以采用较温和的方法，根据其出现异常的次数，逐渐降低其选调概率^[17]，在保证安全性能的情形下，增加利用率，降低维护成本（清洗操作开销大）。

而触发选调器的机制一般为定时触发机制（即周期性选取）和异常事件触发（即判决器等发现不一致时）。

3.1.2 判决器

判决器主要对 switch1~switchM 上报的输出结果进行裁决，然后将结果用 OFPT_PACKET_OUT 消息下发给 switch0 返回用户，并向选调器

和管理模块通告异常。判决器一般采用投票机制,即大数判决,半数以上一致结果判定为正确结果。这种裁决机制是基于假设:攻击者难以对多数异构执行体成功实施攻击,并造成相同的错误输出^[1]。

而判决2个结果是否一致的方法一般通过比对数据分组中某些数据的一致性,需要根据不同的应用场景进行区别设置,如在DNS查询服务中,某个域名可能对应多个IP地址,此时DNS服务器返回的报文中这多个IP地址的顺序可能不同,因此可以采用IP集合比对的方法;若是在异构SDN控制器场景下,则可比对OFPT_FLOW_MOD中的match和actions这2个关键字段^[18]。

由于处理效能或网络时延等影响,可能存在多个输入的输出结果乱序问题,对此也可根据报文中序号ID等标记进行区分输出结果对应的输入,如DNS报文中的transaction_id、ping包中的序列号等。

3.2 拓展模块

拓展模块是除上述基本模块的可选功能拓展,如检测器和管理器等,下面分别进行介绍。

1) 检测器:攻击检测模块负责对网络流量进行抽样分析检测,并向选调器和判决器汇报异常,而后选调器可以调整选调策略,但判决器可以不考虑异常执行体的结果。在移动目标防御和拟态防御模型的情形下,除了常规的检测,还可利用动态冗余模型的优势对攻击者的探测等攻击行为进行检测:若某时刻发现存在处于非活跃态(未被选取)执行体的相关流量,则可判定这些数据为可疑流量^[25],然后着重进行考察。显然,通过这样的针对性检测机制,可以提高检测效率和准确性。除此之外,利用蜜罐技术,将这些可疑流量引导至蜜罐中,从而进一步分析攻击者的行为特征等。

2) 管理器:主要负责执行体的管理,如清洗执行体、新增或删除执行体等。清洗执行体,如通过远程控制清除其状态数据,并重置至初始状态,而触发清洗操作的可以是检测器或者判决器发现异常执行体。另外,还可从系统安全和效能状态的角度进行考察,决定新增或删除执行体,

得益于网络功能虚拟化(NFV, network function virtualization)、Docker等虚拟化技术,可较便捷、快速地新建镜像实例或删除。

3.3 架构安全性分析

通过3.1节和3.2节的设计,本文在软件定义网络下完成了拟态防御实现设计。可以看出,在本文中SDN控制器是实现架构的核心组件,选调、裁决和管理等逻辑功能都集中于SDN控制层、存在单点问题等。因此,架构本身,尤其是SDN控制器的安全问题也需要分析。

首先,SDN控制器本身作为SDN网络中最重要的部分,它集中管理着其覆盖网络的所有视图信息,负责完成网络的基础逻辑功能,同时在本架构中SDN控制器承担着架构中的核心功能,如选调器、判决器和管理器等功能的实现,极易成为攻击的首选目标,因此,架构存在SDN控制器单点失效的安全隐患。

其次,由选调器调度的功能实现可以看出,架构依靠选调器动态下发流表规则至交换机完成网络请求的分发,因此,控制器下发流表的正确性也需要得到保证,否则攻击者可以通过篡改SDN控制器下发的流表规则将流量引至已被攻陷的执行体上,从而完成攻击。因此,架构还存在流表规则篡改攻击的安全威胁。

针对SDN控制器的安全问题,首先,采用安全增强型的控制器,如FortNOX^[26]、SE-Floodlight^[27]和Rosemary^[28]等;其次,为解决上述的单点失效和流表篡改问题,可以采用多控制器的分布式SDN控制层设计,如ONOS^[29]、拜占庭容错NOS^[30],以及文献[31]提出的MNOS,将动态异构冗余思想引入控制层,可以有效解决上述问题。通过上述的安全措施,在增加一定开销代价下,可以有效保障架构中SDN控制器的可用性和可靠性,从而提升架构整体本身的安全性能。

4 关键机制

本节主要介绍地址隐藏机制和动态切换机制,分别为提升架构安全性能和工作效率。

4.1 地址隐藏机制

如前文所述,拟态防御的异构冗余可以增强

系统的入侵容忍性能，然而，其仍然无法应对半数以上执行体被攻击的场景，因此，在面对高级持续性威胁（APT, advanced persistent threat）类攻击时，攻击者可能不惜代价逐一探测并实施攻击，成功渗透。因此，本文提出地址隐藏机制，通过隐藏执行体的真实 IP 地址，进一步保障系统安全。

地址隐藏的目标是对外呈现统一的虚拟 IP 地址，而使用户或潜在的攻击者无法获取执行体的真实地址。具体地，本文利用 OpenFlow 交换机的数据修改功能实现，以 switch1 为例，假设对外暴露的虚拟 IP 地址设置为 10.0.0.2，而执行体 E_1 的真实地址为 10.0.0.101，则在 switch1 中下发的流表项的逻辑如程序 2 所示。可见，switch1 对来源于 switch0 输入的操作是首先修改 IP 地址为执行体 E_1 真实 IP 地址（mod_nw_dst 功能为修改目的网络地址），然后交由执行体 E_1 处理；而对执行体 E_1 的处理结果的操作是修改源 IP 地址为虚拟地址（mod_nw_src 功能为修改源网络地址），然后交由控制器的判决器进行综合裁决。

程序 2

```

1) function mod_nw_dst(x)
2)    modify the destination IP address to be x
//mod_nw_dst 功能为修改报文的目的 IP 地址
3) end function
4) function mod_nw_src(x)
5)    modify the source IP address to be x
//mod_nw_src 功能为修改报文的源 IP 地址
6) end function
7) match1 = (in_port == from_switch0) //流
表规则中的匹配字段，匹配内容为：接收到来自
交换机 switch0 发来的网络请求
8) actions1 = [mod_nw_dst(10.0.0.101), out-
put( $E_1$ )] //流表规则中的行为字段，操作为：首
先修改报文的目的 IP 地址为 10.0.0.101 ( $E_1$  的 IP
地址)，然后转发给  $E_1$ 
9) match2 = (in_port ==  $E_1$ ) //流表规则中的
匹配字段，匹配内容为：接收到来自  $E_1$  的执行结
果，即响应
10) actions2 = [mod_nw_src(10.0.0.2), out-
put(controller)] //流表规则中的行为字段，操作

```

为：首先修改报文的源 IP 地址为 10.0.0.2（虚拟地址），然后转发给 SDN controller 进行裁决等处理

```

11) if match1 == True //如果满足上述
match1，则执行如下操作
12)    then actions1 //执行 action1
13) end if
14) if match2 == True //如果满足上述
match2，则执行如下操作
15)    then actions2 //执行 action2
16) end if

```

通过上述机制，可实现对用户和执行体本身的透明无感：用户将错误地认为处理他们请求的是 IP 地址为 10.0.0.100 的执行体，同时执行体也无需做任何更改。同时，假若在 switch1 和执行体以外的网络位置有任何涉及执行体真实 IP 地址的报文出现，即可认为其可疑（因为执行体对外呈现的都是虚拟 IP 地址），然后将这些可疑流量引至蜜罐，进行攻击行为分析，之后调整相应的防御策略，进一步增强安全性。

4.2 动态切换机制

由于一般的裁决机制都需要对所有执行体的处理结果进行综合判决，因此，判决器需要知道当前活跃（被选取的）执行体的数量，如典型的大数判决。在一般情况下，该数据固定且可以从选调器获取，然而，对于动态切换过程中收到的处理结果的数量却无法确定。例如， t_1 时刻前，活跃执行体数量为 3， t_1 时刻，选调器决定新选取 5 个执行体，并向 switch0 下发流表项，更改转发规则，假设 t_2 时刻，switch0 收到该指令，并完成流表安装工作。显然，对于 $t_1 \sim t_2$ 时间内的输入，仍然是由旧的 3 个执行体处理，而 t_2 以后的输入为新执行体处理，然而这些信息判决器无法获得：控制器到 switch0 以及安装流表的时延和具体从哪个输入开始切换的都无法确定。因此，当判决器收到 3 份处理结果时，它无法确定是否还需要等待剩余 2 份，因此可能一直等待，而实际上可能该输入本身就由旧的 3 个执行体处理，从而导致处理效率降低，时延增加。因此，本文提出一种切换机制，可以有效解决这一问题。

本文的思路是借助 SDN 控制器实现切换时转发和输入的标记工作，具体步骤如下。

1) 在 t_1 时刻, 选调器决定选取新执行体, 并向 switch0 下发流表项, 与程序 1 不同, 这里 actions 字段中的 output 设置为 controller, 即将收到的数据上报给控制器, 而不是直接给 switch1~switchN。

2) t_2 时刻, 指令到达 switch0 并完成流表安装, 在 t_3 时刻, 控制器收到由 switch0 上报的第一个数据 (输入), 并记录其序号 ID, 然后将该输入以及后续收到的输入都发往新执行体对应的交换机。

3) 假设在 t_4 时刻, 判决器收到步骤 2) 中记录序号 ID 的处理结果, 即为第一个由新执行体处理的输入。

4) 最后, 选调器才向 switch0 下发流表项, 将输入直接 output 至执行体对应的交换机, 完成切换过程。

经上述步骤, 判决器将通过时间 t_4 确定具体的数量: 在 t_4 时刻以前的输入, 是由旧执行体处理的, 而 t_4 时刻以后的, 由新执行体处理, 之后即可从选调器获取新、旧执行体的数量。

5 实验评估

本节从安全性能和时延开销的角度, 对架构进行测试评估。本文在华为服务器 (RH2288H V3, Intel Xeon E5-2600 v3 系列处理器) 上搭建实验环境, 采用 mininet(2.2.0) 搭建实验拓扑, 共有 8 个主机 (其中 7 个用作执行体, 1 个作为用户), 用户的 IP 地址为 10.0.0.1, 执行体的真实地址设为 10.0.0.101~10.0.0.107, 对外呈现的虚拟 IP 地址为 10.0.0.2。本文在用户上用 ping 请求 (ping 虚拟地址) 作为输入, 而执行体的 ping 响应报文作为输出。

5.1 入侵容忍能力验证

首先验证架构的入侵容忍能力。在实验设置上, 采用 3 个执行体 (E_1, E_2, E_3), 将 E_1 的 ping 包响应的数据域进行主动修改 (模拟执行体被入侵后, 报文被修改), 而 E_2, E_3 的响应不做处理, 即实现 3 个执行体容忍 1 个错误/入侵。

图 3 是 Ryu 控制器上接收到 switch1~switch3 上报的各执行体的处理结果。可见, 由 switch1 上报的结果 (from switch1 框) 与 switch2~switch3 (from switch2 和 from switch3 框) 不同, 此时, 判决器采取大数判决机制, 判决占多数的结果作

为最终结果 (final answer), 从而实现对 E_1 的错误/入侵容忍。



图3 判决器收到的处理结果

5.2 攻击成功概率仿真

下面对架构的安全性能进行仿真评估。假设一个调度周期内, 攻击者对执行体 E_i 成功实施攻击的概率为 P_a^i , 由于采用异构配置, 因而假设 $P_a^1 = P_a^2 = \dots = P_a^N = P_a$, 则成功攻击系统的概率可计算为 $\sum_{l=(M+1)/2}^M C_M^l P_a^l (1-P_a)^{M-l}$, 即至少成功攻击过半 (大数判决)。图 4 为攻击成功概率与 P_a 和选取出的执行体数量的关系。可见, 执行体数量越多, 系统架构的安全性能越好。

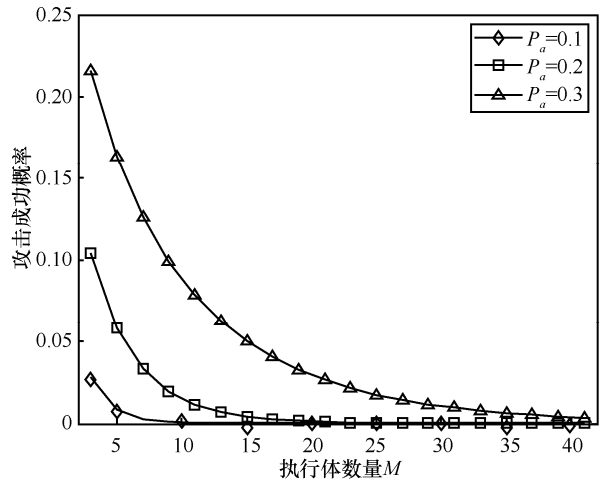


图4 攻击成功概率与执行体数量的关系

5.3 代价分析

由于引入了冗余执行体, 并增设了控制器环节, 涉及选调、判决等模块, 因此本文对传统单

一执行体和本文架构的处理时延进行了对比分析评估。为更好描述和分析引入的额外时延开销，给出本节涉及的相关符号和定义，如表 1 所示。

表 1 本节涉及的相关符号和定义

符号	含义
T_{td_ss}	两交换机间的传输时延
T_{td_cs}	控制器与交换机间的传输时延
T_{td_ce}	执行体到控制器间总时延
T_{pd_s}	交换机修改报文的处理时延
T_{pd_c}	控制器的处理时延
T_{pd_e}	执行体的处理时延
T_{add}	架构增加的时延

因此，传统单执行体的时延可表示为（这里忽略主机到交换机的时延）

$$T_{original} = 2T_{td_ss} + T_{pd_e} \quad (1)$$

而由架构的工作流程可以看出，本文架构的时延可表示为（按处理流程顺序）

$$T_{Mimic} = T_{td_ss} + T_{pd_s} + T'_{td_ce} + T_{pd_c} + T_{td_cs} \quad (2)$$

其中， T'_{ce} 表示本文拟态架构下多执行体到控制器的总时延，可以表示为 $T'_{td_ce} = \max(T_{td_ce}^1, T_{td_ce}^2, \dots, T_{td_ce}^M)$ ，因为判决器需要等待所有报文进行判决，所以这部分时延由 M 个执行体中时延最大的决定，下文简称为“max 效应”。而每个 $T_{td_ce}^i$ 都可表示为 $T_{pd_s} + T_{pd_e} + T_{td_cs}$ 。

因此，额外增加的时延可表示为

$$T_{add} = T_{pd_s} + T'_{td_ce} + T_{pd_c} + T_{td_cs} - T_{td_ss} \quad (3)$$

考虑到本实验中交换机间的传输时延以及交换机修改报文时延无法避免（从改进的角度上看），且经过测试这部分时延约为 0.01 ms，相比而言较小，将其忽略，则式(3)近似为

$$T_{add} = T'_{td_ce} + T_{pd_c} + T_{td_cs} \quad (4)$$

可见，时延的增加主要源于选调、判决等引入的处理时延，控制器与交换机间传输时延，以及由于 max 效应带来的时延。

最终的时延测试结果如图 5 所示。其中，Normal 是指传统的单一执行体的情况，M-1~M-7 表示本文的拟态架构分别采用 1~7 个执行体的情形。

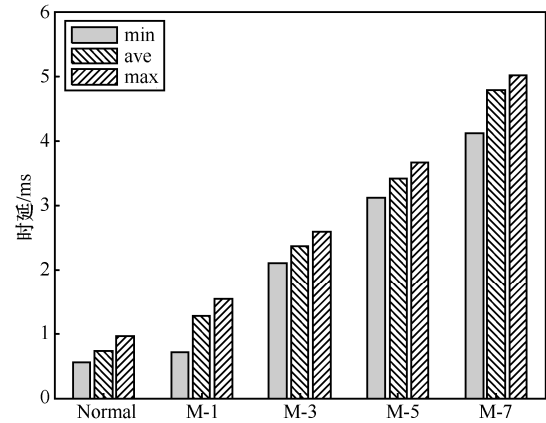


图 5 ping 包时延

显然，M-1 本身并不属于拟态架构，因为只有 1 个执行体（没有冗余空间），本文主要是将其与 Normal 进行对比。此时的时延增加主要源于 T_{td_cs} ，因为其他时延基本相同，而且由于没有选调和判决， T_{pd_c} 也可忽略，此时延约为 0.542 ms。

而将 M-3~M-7 与 M-1 进行对比，则可以看出由 max 效应和控制器选调判决等带来的时延（因为 T_{td_cs} 抵消）。此时的时延大约与执行体数量呈线性增加的趋势，M-7 相比 M-1 的时延增加约为 278%，增幅较大。

上述的时延开销，如 T_{td_cs} 本身由架构导致，无法进行改进， T_{pd_c} 也只能对选调判决机制优化，因此，可以进行改进的是 T'_{td_ce} ，即由 max 效应带来的时延。对此可以采用“先斩后奏”的策略——判决器在接收到一个应答后先将该应答返回用户，然后在后续报文到达后进行第二次判决。显然，若采用这种策略，用户的等待时延可以相应降低，甚至可以比原始的时延还低，因为此时的时延为 $\min(T_{td_ce}^1, T_{td_ce}^2, \dots, T_{td_ce}^M)$ 。当然，上述策略也可以修改为：判决器接收少数应答后先进行一次简易判决，全票则先发给用户，否则等待后续完整的大数判决结果。总体来说，这种策略在不影响架构安全性的情况下，降低了用户的等待时延，当然用户为此付出的代价是安全性的降低，可以认为是效率与安全性的“置换”。

最后，本文对切换机制进行了测试。如图 6 所示，在第 50、100、150 个 ping 包请求时进行切换，分别用不同线型表示不同执行体数量配置下的时延。可见，在每次切换瞬间，时延都会陡增。增幅主要源于：1) 采用了更多的执行体，

max 效应增加; 2) 由于切换期间的 ping 包请求由控制器负责转发分流, 因而增加了其瞬时负载, 处理时延也相应增大。但在切换后的平均第 5 个请求时, 时延就会恢复、下降, 即切换完成后, 控制器负载恢复正常。并且从各配置稳态时的时延也可以看出, 时延随执行体数量增加而增加 (max 效应), 这与图 5 结果相符。

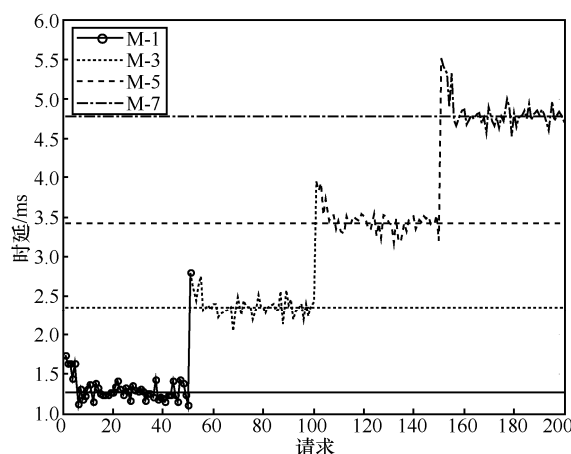


图6 切换时延

6 结束语

邬江兴院士提出的“改变游戏规则”的拟态安全防御思想, 以其动态异构冗余模型有效增加系统的安全性能, 如入侵容忍能力、自身的攻击检测能力等。本文在分析已有的防御机制的基础上, 利用软件定义网络的集中管理控制优势, 提出了一种实现拟态防御思想的安全架构。该架构在 SDN 控制器上实现执行体的动态调度和输出的裁决, 利用 SDN 交换机实现分流和报文修改功能。另外, 为进一步提升性能, 本文提出并实现了地址隐藏机制和动态快速切换机制。

当然, 架构也存在一定的不足, 如实验测试中代价分析部分所述, 由于采用冗余执行体带来的 max 效应, 会增加一定的时延。对此, 可以采用“先斩后奏”机制降低时延。后续的可能研究思路主要包括应用该架构在某些实际场景中, 如 DNS 服务器; 优化判决机制, 进一步降低时延等。

参考文献:

[1] VOAS J, GHOSH A, CHARRON F, et al. Reducing uncertainty about common-mode failures[C]//IEEE Symposium Software Reliability Engineering. 1997: 308-319.

[2] LEVITIN G. Optimal structure of fault-tolerant software systems[J]. Reliability Engineering & System Safety, 2005, 89(3):286-295.

[3] AROMS E. NIST special publication 800-94 guide to intrusion detection and prevention systems (IDPS)[M]. South Carolina: CreateSpace, 2012.

[4] SPITZNER L. Honeypots: catching the insider threat[C]//IEEE Computer Security Applications Conference. 2003:170-179.

[5] JANA S, PORTER D E, SHMATIKOV V. TxBBox: building secure, efficient sandboxes with system transactions[C]//IEEE Symposium on Security and Privacy, IEEE Computer Society. 2011: 329-344.

[6] ZHUANG R, DELOACH S A, OU X. Towards a theory of moving target defense[C]//ACM Workshop on Moving Target Defense. 2014: 31-40.

[7] 邬江兴. 网络空间拟态防御研究[J]. 信息安全学报, 2016, 1(4):1-10.

WU J X. Research on cyber mimic defense[J]. Journal of Cyber Security, 2016, 1(4): 1-10.

[8] MCKEOWN N, ANDERSON T. OpenFlow: enabling innovation in campus networks[J]. ACM Sigcomm Computer Communication Review, 2008, 38(2):69-74.

[9] COX B, EVANS D, FILIPI A, et al. 2006. N-variant systems: a secretless framework for security through diversity[C]//Usenix Security Symposium. 2006:9.

[10] KC G S, KEROMYTIS A D, PREVELAKIS V. Countering code-injection attacks with instruction-set randomization[C]//ACM Conference on Computer and Communications Security. 2003: 272-280.

[11] HOMESCU A, BRUNTHALER S, LARSEN P, et al. Librando: transparent code randomization for just-in-time compilers[C]//ACM Sigsac Conference on Computer & Communications Security. 2013:993-1004.

[12] BHATKAR S, DUVARNEY D C, SEKAR R. Address obfuscation: an efficient approach to combat a board range of memory error exploits[C]// Conference on Usenix Security Symposium. 2003: 105-120.

[13] DUNLOP M, GROAT S, URBANSKI W, et al. Mt6d: a moving target IPv6 defense[C]//IEEE Military Communications Conference. 2011: 1321-1326.

[14] JAFARIAN J H, AL-SHAER E, DUAN Q. An effective address mutation approach for disrupting reconnaissance attacks[J]. IEEE Transactions on Information Forensics and Security, 2015, 10(12): 2562-2577.

[15] JAFARIAN J H, AL-SHAER E, DUAN Q. Adversary-aware IP address randomization for proactive agility against sophisticated attackers[C]//2015 IEEE Conference on Computer Communications (Infocom). 2015: 738-746.

[16] MAYO J R, TORGERSON M D, WALKER A M, et al. The theory of diversity and redundancy in information system security[R]. LDRD Final Report, 2010.

[17] 王祺鹏, 扈红超, 程国振. 一种基于拟态安全防御的 DNS 框架设计[J]. 电子学报.

WANG Z P, HU H C, CHENG G Z. A DNS architecture based on mimic security defense [J]. Chinese Journal of Electronics.

[18] HU H C, WANG Z P, CHENG G Z, et al. MNOS: a mimic network operating system for software defined networks[J]. IET Information Security, 2017.

- [19] WANG H, SCHMITT J. Load balancing-towards balanced delay guarantees in NFV/SDN[C]//IEEE Conference on NFV-SDN 2016: 240-245.
- [20] CASCONI C, SANVITO D, POLLINI L, et al. Fast failure detection and recovery in SDN with stateful data plane[J]. International Journal of Network Management, 2016.
- [21] PANG J, XU G, FU X. SDN-based data center networking with collaboration of multipath TCP and segment routing[J]. IEEE Access, 2017, (99):1-1.
- [22] HAN W, ZHAO Z, AHN G J. HoneyMix: toward SDN-based intelligent honeynet[C]//ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization. 2016:1-6.
- [23] JAFARIAN J H, AL-SHAER E, DUAN Q. Openflow random host mutation: transparent moving target defense using software defined networking[C]//The Workshop on Hot Topics in Software Defined Networks. 2012:127-132.
- [24] GEMBER-JACOBSON A, VISWANATHAN R, PRAKASH C, et al. OpenNF: enabling innovation in network function control[C]//ACM Conference on Sigcomm. 2015:163-174.
- [25] JAFARIAN J H, AL-SHAER E, DUAN Q. Adversary-aware IP address randomization for proactive agility against sophisticated attackers[C]//Computer Communications(IEEE Infocom). 2015: 738-746.
- [26] PORRAS P, SHIN S, YEGNESWARAN V, et al. A security enforcement kernel for OpenFlow networks[C]//The First Workshop on Hot Topics in Software Defined Networks. 2012:121-126.
- [27] CHEUNG S, FONG M, PORRAS P, et al. Securing the software-defined network control layer[C]//Network and Distributed System Security Symposium. 2015.
- [28] SHIN S, SONG Y, LEE T, et al. Rosemary: a robust, secure, and high-performance network operating system[C]//2014 ACM SIG-SAC Conference on Computer and Communications Security. 2014:78-89.
- [29] BERDE P, HART J, HART J, et al. ONOS: towards an open, distributed SDN OS[C]//The Workshop on Hot Topics in Software Defined Networking. 2010:1-6.
- [30] LI H, LI P, GUO S, et al. Byzantine-resilient secure software-

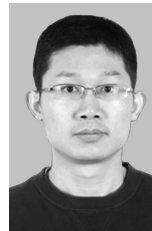
defined networks with multiple controllers in cloud[J]. IEEE Transactions on Cloud Computing, 2015, 2(4):436-447.

- [31] HU H C, WANG Z P, CHENG G Z, et al. MNOS: a mimic network operating system for software defined networks[J]. IET Information Security, 2017.

作者简介:



王祺鹏(1993-), 男, 湖北黄冈人, 国家数字交换系统工程技术研究中心硕士生, 主要研究方向为拟态安全防御。



扈红超(1982-), 男, 河南商丘人, 国家数字交换系统工程技术研究中心副研究员, 主要研究方向为网络安全防御和新型网络体系结构。



程国振(1986-), 男, 山东菏泽人, 国家数字交换系统工程技术研究中心助理研究员, 主要研究方向为主动防御技术和 SDN 安全。

张传浩(1979-), 男, 河南郑州人, 铁道警察学院讲师, 主要研究方向为网络安全。