

异构容错控制平面的安全性分析

吴奇, 陈鸿昶, 陈福才

(国家数字交换系统工程技术研究中心, 河南 郑州 450001)

摘要: 随着软件定义网络的大规模应用, 软件定义网络的安全性显得愈发重要。基于异构思想的容错控制平面作为一种重要的防御思路, 近年来越来越引起研究者的注意。但是现有容错控制平面的研究中忽视了异构原件中的同构漏洞问题, 这大大降低了容错控制架构对软件定义网络的安全收益。从异构原件中的同构漏洞出发, 首先分析了同构漏洞对控制平面的安全影响, 然后以此为基础对容错控制平面的容忍能力进行量化, 构造出一个最大化容忍能力的控制平面布局方法。实验仿真证明了所提方法可以有效降低控制平面的故障概率, 攻击者在对基于所提方法构造的控制平面进行攻击时, 需要花费更多的攻击成本才可以瘫痪控制平面。

关键词: 软件定义网络; 同构漏洞; 异构控制器; 容忍能力

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2018095

Security analysis in heterogeneous fault-tolerant control plane

WU Qi, CHEN Hongchang, CHEN Fucui

National Digital Switching System Engineering and Technological R & D Center, Zhengzhou 450001, China

Abstract: With the large-scale application of software-defined networks, the security of software-defined networks becomes more and more important. As an important defense idea, the fault-tolerant control plane based on heterogeneity has attracted more and more researchers' attention in recent years. However, the existing researches ignore the problem of common vulnerability in heterogeneous variants, which greatly reduces the security benefits of the fault-tolerant control architecture for software-defined networks. Addressing this problem, the common vulnerability was taken in heterogeneous variants into considerations. First, the tolerance capability of the fault-tolerant control plane was quantified. Then a control plane deployment method was constructed which was able to maximize the tolerance capability. The simulations show that the proposed method can effectively reduce the failure probability of the control plane. When the attackers attack the control plane constructed based on the proposed method, they pay more attack cost to compromise the control plane.

Key words: software-defined network, common vulnerability, heterogeneous variant, tolerance capability

1 引言

软件定义网络 (SDN, software defined net-

work) 将控制与转发分离, 形成应用层、控制层、数据层的 3 层结构, 通过南向可编程接口开放了网络, 促进了网络创新, 简化了网络管理^[1]。近

收稿日期: 2018-09-15; 修回日期: 2018-10-20

通信作者: 吴奇, wqstudy@outlook.com

基金项目: 国家重点研发计划资金资助项目 (No.2016YFB0800101); 国家自然科学基金创新群体基金资助项目 (No.61521003)

Foundation Items: The National Key R&D Program of China (No.2016YFB0800101), The Foundation for Innovative Research Groups of the National Natural Science Foundation of China (No.61521003)

些年,随着 SDN 在现实社会中大规模应用,其安全性越来越受到研究者的关注。有些研究者尝试通过增强 SDN 中控制器的安全能力进而提高整个 SDN 的安全性,该类方法通常在 SDN 的内核中增加一个安全内核,如 FortNOX^[2]和 ROSEMARY^[3]中的安全内核,通过权限管理、角色划分的机制保证 SDN 的内核层不被攻击者攻击或篡改。但该类机制无法有效应对 SDN 中的单点失效问题。于是有研究者提出了容错控制平面,该类设计可以通过在控制平面部署多个控制器避免单一控制器的损坏对 SDN 带来的安全隐患,如 kandoo^[4]、CoVisor^[5]等。一旦控制器出现故障,该控制器的备份元件就替换原有的故障控制器,以此完成控制平面的修复。

虽然容错控制平面给 SDN 带来更高的顽健性,但由于容错控制平面中控制器是由结构相同的原件组成的,一旦攻击者利用漏洞使控制平面中一个控制器发生故障,如消息阻塞、控制器宕机等,其他所有的控制器很可能由于该漏洞而出现故障。如图 1 所示,若攻击者已经探知控制平面的控制器为 Floodlight,那么无论容错控制平面利用多少个备份 Floodlight 控制器进行错误修复,攻击者仍可以利用 Exit 宕机攻击瘫痪 Floodlight 控制平面^[6-7]。一旦攻击者攻陷了全部的控制器,控制平面将无法处理请求消息,整个 SDN 就会处于崩溃的状态。

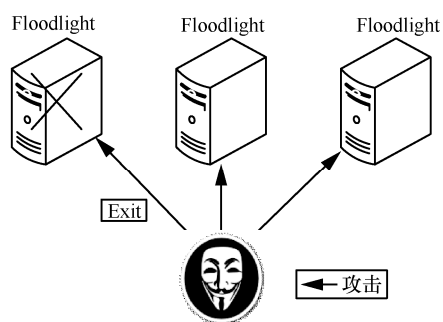


图 1 同构控制平面的安全风险

为了解决这种问题,有研究者尝试使用结构相异的异构控制器对容错控制平面进行优化,进而避免上述的同构错误问题。但实际上,由于很多第三方公用库及开源工具被广泛应用,异构原件间也有可能存在同构漏洞。例如,若控制平面利用 Windows 和 Linux 构造运行环境,

攻击者可以利用 2 种操作系统间的共性漏洞(如 CVE-2018-8897 等^[8])瘫痪控制平面;若控制平面利用 Floodlight 和 ONOS 来构造,攻击者可以利用发布订阅机制中的权限漏洞终止部分控制器的功能,进而瘫痪控制平面^[9]。

故对控制平面而言,如何在异构控制器间存在同构漏洞的前提下,最大限度上发挥容错机制对 SDN 控制层的保护能力是容错控制平面必须解决的问题。本文以此为出发点,设计了一种异构容错控制平面布局方法。首先,本文利用探测次数对单个控制器及整个控制平面的安全能力进行量化;然后,为了最大化控制平面的安全能力,提出了一种基于相似度的异构容错控制平面的布局方法;最后,本文对所提算法进行了仿真,仿真结果证明所提算法可以有效降低控制平面的故障概率,攻击者在对该平面进行攻击时,其需要花费更多的攻击成本才可以瘫痪控制平面。

2 相关工作

安全性问题一直是 SDN 中的重要问题,目前已经有对 SDN 的安全问题进行了深入研究。

有些研究者尝试对控制平面中每一个控制器内部结构中的安全性问题进行研究,Gude 等^[10]在开源控制器 NOX 的基础上设计了一种新的安全内核 FortNOX^[2],其可以利用角色的数据源认证模块(role-based source authentication module)对每个流规则进行签名,并为候选流规则指定相应的特权类别,进而避免流冲突等可能导致控制器故障的问题。之后,他们对开源控制器 Floodlight^[11]进行了类似的安全扩展,设计了 Floodlight 控制器的安全增强版本 SE-Floodlight^[12],SE-Floodlight 控制器继承了 FortNOX 的设计思路,并进行了扩充和完善。除了上述基于已有开源控制器的研究外,还有研究者设计了全新的安全控制器,如 Rosemary^[3]和 PANE^[13],这些控制器在设计和开发之初,便将安全性作为核心问题之一进行考虑,通过突破已有控制器在系统架构、编程语言和预留接口等方面的限制,以此提高控制平面的安全性。

上述安全控制器可以一定程度上解决控制器

内部的安全问题,但是其仍然无法解决 SDN 中的典型安全问题:单点失效问题,即一旦控制器出现错误,整个 SDN 将陷入异常。为了解决这个问题,针对容错控制平面的安全研究应运而生。为了避免单点失效,早期的 SDN 控制平面研究主要关注于分布式控制平面,如 Onix^[14]、OpenDaylight^[15]和 ONOS^[16],这些机制可以通过在控制平面部署多个控制器来避免单点失效问题。之后有研究者提出了层次化控制平面,Yeganeh 等^[4]设计了一种双层控制平面,其核心思路在于顶层控制器管理底层控制器的工作状态,底层控制器负责对数据层直接进行管理,该机制可以保证当一个底层控制器出现故障时,顶层控制器可以迅速发现故障并进行修复。CoVisor^[5]通过在控制层和数据层之间加入一个中间层,将控制层的多个控制器组合成一个控制器,避免了单个控制器故障对整个网络的影响。近些年,还有研究者尝试利用容错控制平面^[17-20]进一步提高控制平面的安全能力,该机制的核心思路为在控制平面部署不同类型的控制器,并通过在控制器间运行拜占庭协议、大数判决等机制,避免多个控制器同时故障的情形。

3 安全性模型

3.1 控制器的故障概率

对于一个控制器而言,攻击者需要经历 2 个阶段(探测弱点、实施攻击),才可以成功实施对控制器的攻击。一般来说,攻击者对控制器探测的次数越多,越有可能发现控制平面的弱点^[21],故利用攻击者对控制器的探测数表示控制器的安全能力,可以表示为

$$P_j = 1 - e^{-b_j n} \quad (1)$$

其中, P_j 表示控制器发生故障的概率,当控制器故障率 $P_j \geq \phi$ 时,控制器看作被攻击者攻陷。 n 表示攻击者对控制器的探测次数, b_j 表示控制器的弱点系数, j 表示类型为 j 的控制器。 b_j 的值由两方面决定。

其一由控制器自身的安全能力决定。一般来说,控制器自身的漏洞数 V_j 越多,攻击者瘫痪控制器所需要的探测次数越少, b_j 越大。结合

式(1), b_j 和控制器自身的漏洞数 V_j 正相关。

$$b_j \propto V_j \quad (2)$$

其二由攻击者的先验知识决定。由于现有的异构设计难以做到完全的相异性,攻击者在攻击一个控制器时获得的知识可能会指导之后的攻击过程。这里利用收益参数 ρ_j 定义攻击者的先验知识。

$$\rho_j = \max \left(\frac{V_{ij}}{V_i} \right), i \in \mathcal{G} \quad (3)$$

其中, V_i 表示控制器 i 中的全部漏洞, V_{ij} 表示控制器 i 和控制器 j 间的共性漏洞。 $\mathcal{G}$ 表示在控制器 j 前被攻陷的控制器集合。式(3)表明攻击者攻击控制器 j 时,可以从已经攻陷的控制器集合 $\mathcal{G}$ 中学习到的最多攻击知识。考虑到 $\rho_j \in [0,1]$, 设定 b_j 和探测收益参数 e^{ρ_j} 正相关。

$$b_j \propto e^{\rho_j} \quad (4)$$

结合式(2)和式(4), b_j 可以定义为

$$b_j = \alpha e^{\rho_j} V_j \quad (5)$$

将式(5)代入式(1),可以计算出控制器故障概率和攻击者探测次数的关系。

$$P_j = 1 - e^{-\alpha e^{\rho_j} V_j n_j} \quad (6)$$

3.2 控制平面的故障概率及容忍能力

由各个控制器的安全性模型可以构造出控制平面的故障概率。假设容错平面 C 是由 K 个控制器组成的,只有当攻击者攻陷全部的 K 个控制器时,控制平面才发生故障,故对于容错控制平面而言,其故障概率可以表示为

$$P_C = \prod_{j \in K} P_j \quad (7)$$

其中, P_C 表示控制平面 C 的故障概率。一旦 K 个控制器全部发生故障时,控制平面的容错机制将完全失效。对于攻击者而言,此时需要进行的探测次数可以表示为

$$Cost = \sum_{j \in K} \frac{-\log(1 - \phi)}{\alpha e^{\rho_j} V_j} \quad (8)$$

其中, $Cost$ 表示攻击者突破控制平面容忍能力的探测次数。由于攻击过程中攻击者先验知识获取存在先后关系,导致攻击者在瘫痪控制平面时采取不同的攻击次序所需要的花销是不同的,为有

效地表示控制平面的容忍能力，规定攻击者瘫痪控制平面所需要的最少探测花销就是控制平面的容忍能力，可以定义为

$$N_{\text{tol}} = \min \text{Cost} \quad (9)$$

4 控制平面布局问题

对容错控制平面而言，若控制平面的可选控制器池 C_M 包含的控制器数量 M 和控制平面需要的控制器数量 K 相同，即 $M=K$ 时，由于控制平面无法改变其中的控制器，故其安全能力是固定的，此时研究控制平面中的控制器选择问题是无意义的。但当控制平面的可选控制器池的大小 M 大于控制平面的控制器数 K 时，即 $M > K$ 时，控制平面可以通过不断选择控制器改变自身的容忍能力，此时研究控制平面中的控制器选择问题是有益的。因此，本文着重对 $M > K$ 情形下控制平面中异构控制器的选择问题进行分析，主要研究“如何从包含 M 个控制器的可选控制器池中选择出最优的 K 个控制器，使控制平面的容忍能力最大”的问题。该问题可以表示为

$$\max \min \sum_{j \in C} \frac{-\log(1-\phi)}{\alpha e^{\rho_j} V_j} \quad (10)$$

s.t.

$$M > K \quad (11)$$

$$0 < \phi, \xi < 1 \quad (12)$$

在该模型中，目标函数（如式(10)所示）是寻找具有最大化容忍能力的控制平面 C ，约束（如式(11)所示）表示可选控制器池 M 的大小大于控制平面的控制器数量 K ，约束（如式(12)所示）表示模型中参数的取值范围在阈值内。

5 问题求解

若采用枚举法对上述模型进行求解，需要首先计算出从 M 个可选控制器池中选择 K 个控制器的全部可能，共需要遍历 C_M^K 次；然后分析出每一种控制平面的容忍能力，需要遍历 A_K^K 次。在这个过程中所需要的计算量为 $C_M^K A_K^K$ ，该过程的计算复杂度为 $O(M^K)$ 。

若 M 、 K 都比较小，可以利用枚举法计算出最优的控制平面。但当 M 、 K 都比较大时，枚举出全部情况的成本极高，本文借鉴模拟退火算法，

设计了一种基于相似度的异构容错控制平面的布局方法进行求解。该算法可以通过相似度指标，减少对相似的控制平面解进行分析，进而加快求出最优解的速度。

首先，定义控制平面的相似度指标，可以被计算为

$$\text{sim} = \sum_{j=1}^M C'_j C''_j \quad (13)$$

其中，控制平面 C' 和 C'' 表示控制平面中控制器的组成向量，其向量长度等于 M ， C'_j 表示控制平面 C' 中的第 j 个控制器，若该控制器被部署于控制平面 C' 中，则 C'_j 的值为 1，反之为 0。

从式(13)可以发现， sim 值越大，控制平面 C' 和 C'' 的控制器组成越接近。

基于相似度指标可以在对控制平面的解进行遍历时，通过调整 2 个解之间的相似度，主动调节控制平面的遍历速度。假设 C^{i-1} 和 C^i 是已经计算出的控制平面第 $i-1$ 和 i 个解，则利用这 2 个解，控制平面第 $i+1$ 解与第 i 个解之间的不相似度需要满足

$$\text{dsim}_{i,i+1} = \begin{cases} \left\lceil \frac{N_{\text{tol}}^{i-1}}{N_{\text{tol}}^i} \right\rceil + 1, N_{\text{tol}}^{i-1} > N_{\text{tol}}^i \\ 1, N_{\text{tol}}^{i-1} \leq N_{\text{tol}}^i \end{cases} \quad (14)$$

其中， $\text{dsim}_{i,i+1} = K - \text{sim}_{i,i+1}$ 表示控制平面第 i 和 $i+1$ 个解之间的不相似度。从式(14)中可以看出，控制平面第 $i-1$ 个解的容忍能力比第 i 个解的容忍能力大，表明第 i 个解更差，故控制平面的第 $i+1$ 个解必须有较大改变。相对地，控制平面第 i 个解的容忍能力比第 $i-1$ 个解的容忍能力大，表明第 i 个解更优，此时控制平面的第 $i+1$ 个解只需要满足最低限度的不相似度即可。

图 2 展示了控制平面解的遍历过程，在图中假定依据历史信息已经得到 $\text{dsim}_{i-1,i}$ 为 3，则解 C^{i-1} 切换到解 C^i 需要变换 3 个元素，解 C^{i-1} 中容忍能力最低的 C_2 、 C_4 、 C_6 被解 C^i 中的 C_7 、 C_8 、 C_9 取代，利用解 C^{i-1} 和 C^i 计算 $\text{dsim}_{i,i+1}$ 为 1 后，解 C^i 切换到解 C^{i+1} 需要变换 1 个元素，解 C^i 中容忍能力最低的 C_3 被解 C^{i+1} 中的 C_2 取代。

考虑到上述过程可以使解陷入局部最优解，本文借鉴模拟退火算法中的 Metropolis 准则接受控制平面的新解。

$$P(\Delta N_{\text{tol}}) = \begin{cases} 1, \Delta N_{\text{tol}} > 0 \\ \exp\left(\frac{\Delta N_{\text{tol}}}{T}\right), \Delta N_{\text{tol}} \leq 0 \end{cases} \quad (15)$$

其中, $\Delta N_{\text{tol}} = N_{\text{tol}}^i - N_{\text{tol}}^{i-1}$, T 为模拟退火算法的温度, 其更新机制为 $T_{i+1} = \gamma T_i$ 。式(15)表示若控制平面第 i 个解的容错能力大于第 $i-1$ 个解的容忍能力, 所提方法将以 1 的概率接受新解, 否则, 将以 $P(\Delta N_{\text{tol}})$ 的概率接受新解。

上述过程不断进行, 直到 $|N_{\text{tol}}^i - N_{\text{tol}}^{i-1}|$ 的差值小于设定的阈值 ε 。所提方法的伪代码如算法 1 所示。

算法 1 基于相似度的异构容错控制平面的布局方法

输入 可选控制器池 C_{pool} , 控制平面的控制器数量 K , 参数 α 、 γ , 初始温度 T , 最大迭代次数 i_{max}

输出 最优控制平面 C_{final}

1) 从可选控制器池中随机选择一个控制平面的解 C^0 , 并计算 C^0 的容忍能力 N_{tol}^0 , $i=0$, $dsim_0=1$, $T_i=T$

2) while $|\Delta N_{\text{tol}}| > \varepsilon$

3) $i = i + 1$

4) $T_{i+1} = \gamma T_i$

5) 计算 $dsim_{i-1,i}$ 及控制平面的新解 C^i , 容忍能力 N_{tol}^i

6) if $\Delta N_{\text{tol}} > 0$

7) $C^{i-1} \leftarrow C^i$

8) else

9) if $P(\Delta N_{\text{tol}}) \geq \text{rand}(1)$

10) $C^{i-1} \leftarrow C^i$

11) end if

12) end if

13) if $(i \geq i_{\text{max}})$

14) break

15) end if

16) end while

17) 输出最优控制平面 C_{final}

6 仿真实验

本节对所提方法进行实验评估。为合理地对所提方法进行评价, 本文利用枚举法 (Optimal)、随机法 (Random) 和最差法 (Worst) 与所提方法进行对比, 枚举法是指遍历全部的解空间寻找到容忍能力最大的控制平面; 随机法是指随机从可选控制器池中选取控制平面, 为避免随机误差, 运算 10 000 次取平均值; 最差法是指从可选控制器池中选取容忍能力最差的控制平面。

考虑到控制器又被称为网络操作系统 (network operation system) [10], 借鉴操作系统中漏洞构造实验所需要的漏洞数据集。操作系统中漏洞统计数据来自 CVE (common vulnerabilities and exposures) [7], 其中包含 11 种操作系统间的漏洞表, 如表 1 所示。其中, OB 表示 OPENBSD, NB 表示 NETBSD, FB 表示 FREEBSD, W03 表示 WINDOWS SERVER 2003, W08 表示 WINDOWS SERVER 2008, W12 表示 WINDOWS SERVER 2012, UB 表示 UBUNTU, DE 表示 DEBIAN, RE 表示 REDHAT, SO 表示 SOLARIS, OS 表示 OPENSOLARIS。

表 1 是对角线上的数据表示控制器自身的漏洞, 其他的数据表示两两控制器间的漏洞, 即共性漏洞。实验所使用的可选控制器池包含 11 种控制器, 即可选控制器池大小为 $M = 11$ 。

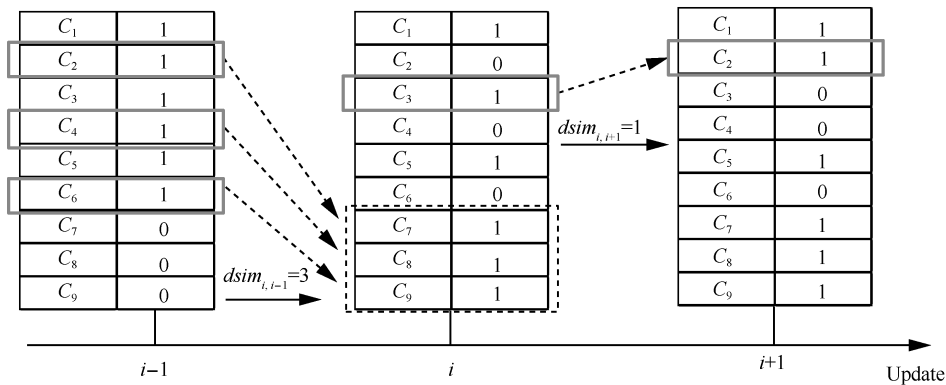


图 2 控制平面解的遍历过程

表 1 操作系统中的漏洞统计

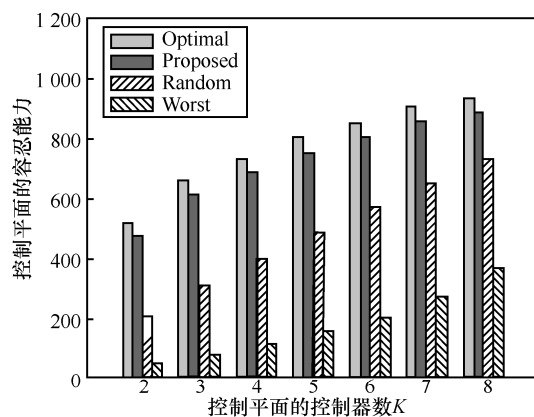
	OB	NB	FB	W03	W08	W12	UB	DE	RE	SO	OS
OB	149	45	58	2	1	0	3	2	10	13	1
NB	45	139	56	1	1	0	0	4	8	16	0
FB	58	56	254	3	2	0	6	9	22	22	0
W03	2	1	3	412	344	85	0	0	1	7	0
W08	1	1	2	344	846	385	0	0	0	0	0
W12	0	0	0	85	385	467	0	0	0	0	0
UB	3	0	6	0	0	0	840	909	176	86	1
DE	2	4	9	0	0	0	303	995	220	76	1
RE	10	8	22	1	0	0	179	220	1518	69	1
SO	13	16	22	7	0	0	86	73	69	365	27
OS	1	0	0	0	0	0	1	1	1	57	100

6.1 控制平面容忍能力

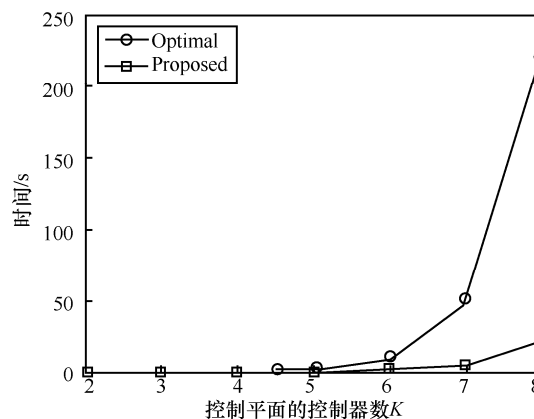
参数 K 是所提算法中最为重要的参数，实验中不断改变 K 的值，进而观察 4 种方法下控制平面的容忍能力变化。规定 K 的取值范围为 $[2, 8]$ ，实验结果如图 3 所示。

图 3 给出控制平面控制器数量 K 变化时所提方法的容忍能力和枚举法、随机法及最差法的对比，在图 3(a)中，利用最差法构造的控制平面的安全性是最差的，随机法虽然优于最差法，但和所提方法及枚举法的差距仍然很大。在图 3(a)中，枚举法略优于所提方法，这是由于所提方法没有搜索全部的解，它寻找出的控制平面的容忍能力接近但并不一定是最优的控制平面。本文分析了枚举法和所提方法的计算时间，如图 3(b)所示，显然，所提方法搜索时间远小于枚举法的所需要的搜索时间，控制平面中的控制器数 K 越大，所提算法相对于枚举法搜索时间收益越明显。

在图 3(a)观察到，随着 K 越接近 M ，几种方法的安全收益越接近，这是由于可选控制器池的空间越小，控制平面可以改变的空间越小。结果表明，所提方法在 M 远大于 K 时才可以充分发挥作用， M 比 K 大得越多，所提算法的安全增益越明显。



(a) 控制平面的容忍能力和控制器数量的关系



(b) 枚举法和启发式算法的计算时间

图 3 控制平面的控制器数量 K 变化时所提方法和几种方法对比

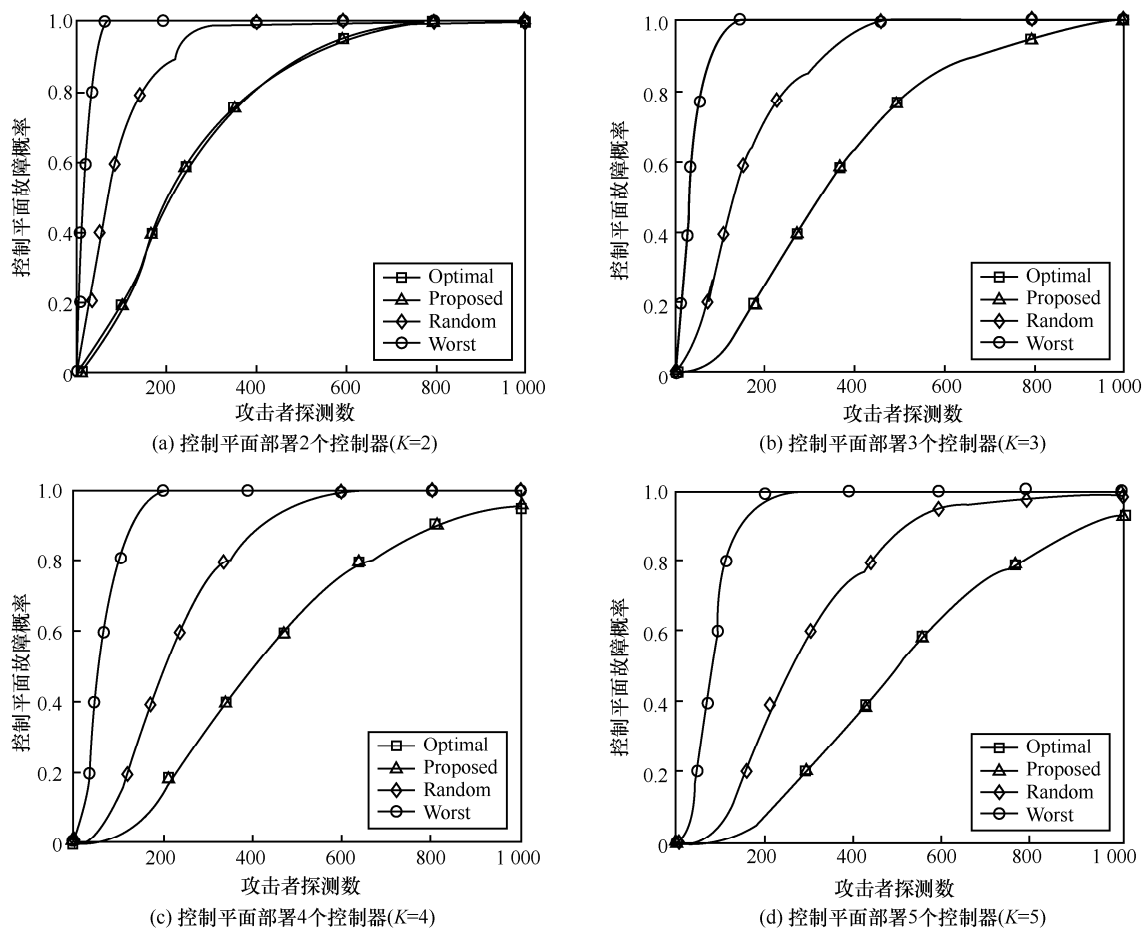


图4 不同方法下控制平面故障概率变化的对比

6.2 控制平面故障概率

上述场景中都认为攻击者遵循最优攻击路径对控制平面进行攻击, 实际中, 攻击者通常难以知道最优攻击路径, 故在接下来的仿真中, 假定攻击者可以随机地对控制平面中的控制器发动恶意探测, 并且每一次攻击者只能针对一个控制器进行攻击。分别验证了 $K \in [2, 5]$ 时, 控制平面故障概率的变化, 具体的仿真结果如图4所示。

图4表示了控制平面部署不同数量的控制器时, 控制平面的故障概率随着探测次数增加的变化情况。通过4幅图的横向对比可以发现, 随着控制器数量增多, 控制平面故障概率的提升速度会明显变慢, 但不同方法配置的控制平面故障概率的变化速度不同, 显然, 在攻击者探测数量增多时, 所提方法与枚举法构造出的控制平面的故障概率增长速度接近, 它们的性能远远优于随机法和最差法, 这表明所提方法有效地降低了控制平面的故障概率。

7 结束语

本文首先利用控制器中的漏洞对控制平面的容忍能力和故障概率进行定义, 分析了同构漏洞对控制平面安全性的影响; 然后为了提高控制平面的容错能力, 提出了一种基于相似度的异构容错控制平面的布局方法。最后, 对所提模型及算法进行验证, 实验结果表明, 所提算法可以有效提高控制平面的容错能力, 降低攻击者攻击对控制平面安全能力的影响。

参考文献:

- [1] SCOTT-HAYWARD S, NATARAJAN S, SEZER S. A survey of security in software defined networks[J]. IEEE Communications Surveys & Tutorials, 2016, 18(1):623-654.
- [2] PORRAS P, SHIN S, YEGNESWARAN V, et al. A security enforcement kernel for OpenFlow networks[C]//International Workshop on Hot Topics in Software Defined Networks. ACM, 2012:121-126.
- [3] SHIN S, SONG Y, LEE T, et al. Rosemary: a robust, secure, and high-performance network operating system[C]//Conference on

- Computer and Communications Security. ACM, 2014:78-89.
- [4] YEGANEH S H, GANJALI Y. Kandoo: a framework for efficient and scalable offloading of control applications[C]// The Workshop on Hot Topics in Software Defined Networks. ACM, 2012:19-24.
- [5] JIN X, GOSSELS J, Rexford J, et al. CoVisor: a compositional hypervisor for software-defined networks[C]// Usenix Conference on Networked Systems Design and Implementation. USENIX Association, 2015:87-101.
- [6] LENG J, ZHOU Y, ZHANG J, et al. An inference attack model for flow table capacity and usage: exploiting the vulnerability of flow table overflow in software-defined network[J]. Water Air & Soil Pollution, 2015, 85(3):1413-1418.
- [7] 卢振平, 陈福才, 程国振. 软件定义网络中控制器调度时间机制设计与实现[J]. 网络与信息安全学报, 2018(1):36-44.
- LU Z P, CHEN F C, CHENG G Z. Design and implementation of the controller scheduling-time in SDN[J]. Chinese Journal of Network and Information Security, 2018(1):36-44.
- [8] [EB/OL]. <http://cve.mitre.org/>
- [9] YOON C, LEE S, KANG H, et al. Flow wars: systemizing the attack surface and defenses in software-defined networks[J]. IEEE/ACM Transactions on Networking, 2017, (99):1-17.
- [10] GUDE N, KOPONEN T, PETTIT J, et al. NOX: towards an operating system for networks[J]. ACM Sigcomm Computer Communication Review, 2008, 38(3):105-110.
- [11] Floodlight documentation[EB/OL]. <http://www.projectfloodlight.org/display/floodlightcontroller/Floodlight+Documentation>
- [12] CHEUNG S, FONG M, PORRAS P, et al. Securing the software-defined network control layer[C]// Proceedings of the Network and Distributed System Security Symposium, 2015.
- [13] FERGUSON A D, GUHA A, LIANG C, et al. Participatory networking: an API for application control of SDNs[C]//ACM SIGCOMM. 2013:327-338.
- [14] KOPONEN T, CASADO M, GUDE N, et al. Onix: a distributed control platform for large-scale production networks[C]//Usenix Conference on Operating Systems Design and Implementation. 2010:351-364.
- [15] MEDVED J, VARGA R, TKACIK A, et al. OpenDaylight: towards a model-driven SDN controller architecture[C]// IEEE, International Symposium on a World of Wireless, Mobile and Multimedia Networks. IEEE, 2014:1-6.
- [16] BERDE P, HART J, HART J, et al. ONOS: towards an open, distributed SDN OS[C]// The Work-shop on Hot Topics in Software Defined Networking. ACM, 2014:1-6.
- [17] Castro M, Liskov B. Practical byzantine fault tolerance and proactive recovery[J]. ACM Transactions on Computer Systems, 2002, 20(4):398-461.
- [18] BOTELHO F, BESSANI A, RAMOS F M V, et al. On the design of practical fault-tolerant SDN controllers [C]// European Workshop on Software-Defined Networks, 2014: 73-78.
- [19] Li H, Li P, Guo S, et al. Byzantine-resilient secure software-defined networks with multiple controllers[J]. IEEE Transactions on Cloud Computing, 2015, 2(4):436-447.
- [20] 王祺鹏, 扈红超, 程国振, 等. 软件定义网络下的拟态防御实现架构[J]. 网络与信息安全学报, 2017, 3(10):52-61.
- WANG Z P, HU H C, CHENG G Z, et al. Implementation architecture of mimic security defense based on SDN[J]. Chinese Journal of Network and Information Security, 2017, 3(10):52-61.
- [21] KELLER E, et al. Timing-based reconnaissance and defense in software-defined networks[C]// Conference on Computer Security Applications. ACM, 2016:89-100.

[作者简介]



吴奇 (1991-), 男, 江苏徐州人, 国家数字交换系统工程技术研究中心博士生, 主要研究方向为网络安全。

陈鸿昶 (1964-), 男, 河南郑州人, 国家数字交换系统工程技术研究中心研究员、博士生导师, 主要研究方向为网络安全方向、大数据。

陈福才 (1974-), 男, 江西高安人, 国家数字交换系统工程技术研究中心研究员、硕士生导师, 主要研究方向为网络安全方向、大数据。