全要素SDN指纹攻击及其模糊混淆防御机制研究

王 涛,陈鸿昶
(中国人民解放军战略支援部队信息工程大学,河南郑州 450003)

摘 要: 软件定义网络(Software-Defined Networking,SDN)的"三层两接口"架构使攻击者可以通过分析数据包往返时延分布规律推测网络类型、控制器类型及关键流规则等指纹信息.目前SDN指纹攻击及其防御研究基本处于空白状态,为此,本文系统构建了全要素SDN指纹攻击链,并在双重时间维度分别设计了概率加扰和控制器混淆调度防御机制,通过渐变概率加扰与最优混淆调度协同提升SDN指纹信息隐藏度.实验结果表明该机制能够在有效隐藏SDN指纹信息的同时减少对网络性能的影响.

关键词: SDN指纹攻击;往返时延;模糊混淆;指纹信息隐藏;概率加扰;动态混淆调度
中图分类号: TP309.1 **文献标识码:** A **文章编号:** 0372-2112 (2020)06-1213-07
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2020.06.024

Research on a Full-Factor SDN Fingerprint Attack and Its Fuzzy Confusion Defense Mechanism

WANG Tao, CHEN Hong-chang
(*PLA Strategic Support Force Information Engineering University, Zhengzhou, Henan* 450003, *China*)

Abstract: The "three-layer two-interface" architecture of software-defined networking (SDN) enables attackers to infer fingerprint information such as network type, controller type, and key flow rules by analyzing the round-trip time distribution of packets. Currently SDN fingerprint attack and its defense research are not mature, so this paper constructs a full-factor SDN fingerprint attack chain. Then, the probabilistic scrambling mechanism and controller dynamic confusion scheduling mechanism are designed in the dual time dimension respectively. More specifically, the gradient probabilistic scrambling and optimal confusion scheduling synergistically promote the information hiding degree of SDN fingerprint. The experimental results show that the mechanism can effectively hide the SDN fingerprint information while reducing the impact on network performance.

Key words: SDN fingerprint attack; round-trip time; fuzzy confusion; fingerprint information hiding degree; probabilistic scrambling mechanism; controller dynamic confusion scheduling mechanism

1 引言

SDN[1]作为一种新型网络体系架构受到研究人员的广泛关注.SDN将控制平面与数据平面分离,并通过定义标准的接口来简化网络运维管理.但是,控制平面与数据平面解耦式的结构为攻击者发动面向SDN的指纹攻击提供可能.当交换机流表内存在与数据包相匹配的流规则时,数据包直接由交换机硬件高速率转发;然而,当交换机流表内不存在与数据包相匹配的流规则时,交换机触发table-miss事件通知控制器安装相应流规则后再由交换机硬件完成转发.由于交换机硬件的转发处理速度比软件定义的控制器生成策略速度快数个数量级,因此在上述两种数据包转发过程中转发时延存在明显差异.攻击者可以利用此类属性推测获得网络类型、控制器类型,甚至流规则等关键信息,为后续发动DoS攻击等更具针对性和威胁性的攻击奠定基础.目前,国内外针对SDN指纹攻击的研究还处于初步阶段,相关攻击及防御技术远未成熟.现有为数不多的防御方案[2]为了防御部分类型的SDN指纹攻击将所有接收到的需要转发的数据包统一延迟,这会显著降低网络性能,不利于实

收稿日期:2019-07-15;修回日期:2019-12-18;责任编辑:覃怀银
基金项目:国家重点研发计划课题(No.2018YFB0804004);国家自然科学基金创新研究群体项目(No.61521003)

际部署. 因此,本文提出了基于概率加扰和控制器混淆调度的 SDN 指纹攻击防御机制,使系统在有效隐藏 SDN 指纹信息的同时减少对网络性能的影响.

2 全要素 SDN 指纹攻击模型

目前 SDN 指纹攻击远未完善,相关研究论文数量少且研究点分散,为了完善思路突出重点,本章将在相关研究工作[3~5]的基础上系统构建全要素 SDN 指纹攻击链并着重论述网络类型、控制器类型和关键流规则等指纹信息的提取及利用过程.

2.1 网络类型指纹信息

准确识别网络类型信息是进行 SDN 指纹攻击的第一步. 攻击者可以利用数据包往返时延(RTT)推测获得网络类型. 考虑到 RTT 在很大程度上取决于主机的地理位置以及实时网络条件等其他因素,因此,为了消除这些无关因素干扰,可以测量由攻击者发出的两个探测包的 RTT 之间的差异 ΔRTT,该指标不依赖于攻击者主机的网络位置,主要由 SDN 控制器与交换机的交互开销和网络抖动构成. 在网络正常时,网络抖动所带来的影响可以相较于交互开销可以忽略[6],即:

$$\Delta \mathrm{RTT} = \max_{\forall k}\delta_k^1 - \max_{\forall k}\delta_k^2 \tag{1}$$

其中,δ_k^i 表示 SDN 控制器和 OpenFlow 交换机 k 之间可能的通信对第 i 个数据包所引入的延迟. 如果两个数据包都没有导致任何规则安装或网络类型为普通传统网络,那么 $\Delta\mathrm{RTT}\approx 0$. 如果其中任意一个数据包触发规则安装则证明该网络类型为 SDN 网络,即 $\max_{\forall k}\delta_k^1 \gg 0$ 或 $\max_{\forall k}\delta_k^2 \gg 0$,那么 $|\Delta\mathrm{RTT}| \gg 0$. 因此,在不同网络类型情况下,ΔRTT 指标具有显著差异性.

2.2 SDN 控制器类型指纹信息

在识别目标网络类型为 SDN 后,攻击者为进一步对目标网络发动高效精准攻击,通常将 SDN 控制器类型信息作为指纹攻击的重要突破点. SDN 控制器由于使用了不同类型的编程语言、函数库和框架导致不同控制器执行速度存在一定差异. 通过测量目标控制器的响应时间,并将其与预先创建的不同控制器的处理时间数据库进行比较,即可得出结论. 为了创建处理时数据库,攻击者可以重复发送 n 个 ping 包,每两个 ping 包的时间间隔大于空闲超时值. 按照上述方式,每次 ping 都会导致交换机向控制器发送 Packet-In 消息,这样即可得出 n 个 ping 包的平均时间 T_{pavg}. 然后,攻击者再次测量存在流规则的情况下 n 个数据包的平均 RTT 值 $\mathrm{RTT}_{\mathrm{avg}}$. 这样即可得出当前控制器的处理时间 T_{pavg}-$\mathrm{RTT}_{\mathrm{avg}}$. 针对目前所有控制器重复以上过程即可得到控制器处理时间数据库(controller;processing time(T_p)). 最后,通过比较测量时延与数据库时延差异(Compare(RTT'-$\mathrm{RTT}_{\mathrm{avg}}$,processing time($T_p$)))即可推测出控制器类型,如算法 1 所示.

算法 1 识别控制器类型指纹

1:Calculate idle-timeout and $\mathrm{RTT}_{\mathrm{avg}}$
2:for $i=1$ to m do
3: wait period > idle-timeout seconds
4: send a ping and save ping time
5:end for
6:Calculate the average of saved ping-time values RTT′
7:Compare RTT′-$\mathrm{RTT}_{\mathrm{avg}}$ to the processing-time entries

2.3 关键流规则指纹信息

当攻击者发动指纹攻击成功探测到交换机中的流规则信息后,可以更好地理解数据包转发逻辑及网络连通图. 为识别关键流规则指纹信息,攻击者同时向目标网络发送时间探测流和测试数据流. 专门构造的时间探测流能够触发控制器与交换机的交互,其往返时间取决于控制器. 测试数据包是特殊虚假构造的数据包,根据目标流规则的属性不断调整头部字段值,然后观察测试数据包对时间探测流往返时延的变化,直到通过分布规律可以探测出相应的流规则. 为表述关键流规则指纹攻击具体过程,假设目标网络中有 4 台主机 $h1\sim h4$,MAC 地址分别为 00:00:00:00:00:01 至 00:00:00:00:00:04,流规则示例如图 1 所示.

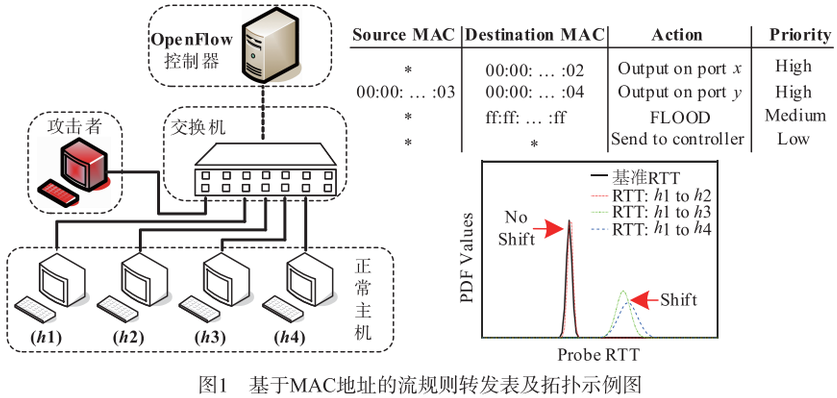


图1 基于MAC地址的流规则转发表及拓扑示例图

攻击者构建不同目的 MAC 地址的测试包(分别从 $h1$ 到 $h2$、$h3$ 和 $h4$),并统计对应的时间探测流的 RTT. 如图可知,目的 MAC 地址为 $h2$ 的测试数据流所对应的 RTT 相较于基准 RTT 分布并没有显著偏移,这表明交换机内存在与之相匹配的流规则而没有将测试数据包转发到控制器;然而,目的 MAC 地址为 $h3$ 和 $h4$ 的测试数据流存在显著分布偏移,表明测试数据包没有匹配的流规则需要控制平面进一步处理. 由此可以推断出,交换机流表内存在 MAC 地址由 $h1$ 到 $h2$ 的流规则映射.

3　概率加扰与控制器动态混淆调度机制

3.1　概率加扰机制

由于指纹攻击者是利用时延分布差异识别重要信息,因此通过可变概率对时延加扰可以从一定程度上增加攻击难度. 具体而言,在单轮防御时间窗内,SDN 控制器可根据不同流属性按照一定概率添加随机扰动以迷惑攻击者. 相比于交换机在转发操作之前延迟每一个数据包和随机删除流规则并在接收相应的数据包输入事件时重新安装等方案,通过制定概率加扰策略使得控制器既能有效干扰 SDN 指纹识别也能尽最大限度保证网络服务质量.

如图 2 所示,概率加扰机制主要由监控器、Hash 表、策略生成模块和概率加扰执行代理等四个模块实现. 其中,监控器负责监听并收集数据平面状态信息,并将该信息存入 Hash 表;Hash 表将提取流的源、目的地址映射为索引,如果表中没有匹配索引,则新建条目并将其值列表初始化为 0,如果在表中已经存在相关索引,则更新对应值列表;策略生成模块中概率决策组件决定流中具体数据包的延迟概率,即完成对流中哪些数据包进行延迟操作的决策 Delay($Packet_i$),而随机时延扰动组件负责确定需要延迟的数据包扰动值 Time($Packet_i$),最终策略生成模块将 < Delay($Packet_i$),Time($Packet_i$) > 策略组合发送至概率加扰执行代理;概率加扰执行代理负责将加扰策略转化为数据平面可执行的指令,根据概率干扰策略标记不同数据包,然后通过为组表(group table)定义新的动作桶(action buckets)选择逻辑实现对不同数据包执行不同延时操作,从而达到混淆时延分布的效果. 该方案具体流程如算法 2 所示.

算法 2　概率加扰

Input:Hash table H;Average RTT rtt
Output:Scrambling packet with delay time dt($packet_i$)
1:while TRUE do
2:　$packet_i$ ← receive a packet
3:　index = hash(extractHeader($packet_i$))
4:　if the index of $packet_i$ is not in the Hash table H then
5:　　H.add(index)
6:　　H(index).Counter ← 0 and H(index).Tag ← 0
7:　　for NumTag = 0 to m do
8:　　　setDelayTag(Flow(index).$Packet_{NumTag}$, genProbability(NumTag))
9:　　if the label of Flow(index).$Packet_{NumTag}$ is Delayed then
10:　　　delay(Flow(index).$Packet_{NumTag}$, random(0.5,1) * rtt) to proxy
11:　　end if
12:　end for
13:end while

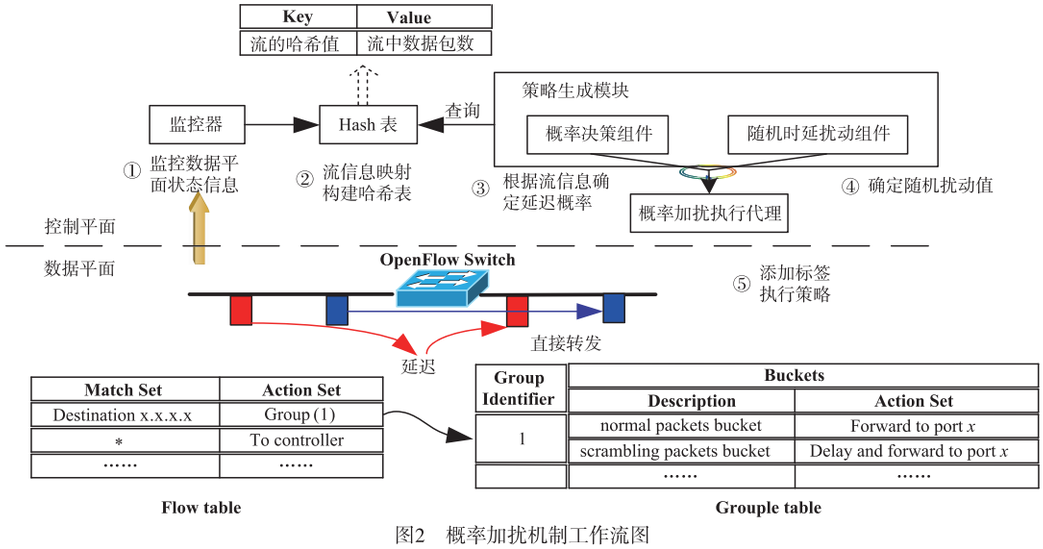


图2　概率加扰机制工作流图

一般而言,SDN 指纹攻击者为了追求攻击效率仅在某伪造流中发送少量虚假探测数据包,通过多次统

计不同伪造流中的数据包时延差异来识别指纹信息. 基于该指纹攻击特性,本文设计了一种随着流中数据包数量渐变的概率模型,该模型以较高的干扰概率对流中初始数量的数据包实施干扰,而对后续数据包以弹性渐变的概率施加干扰. 该模型如式(2)所示.

$$P_c = \text{genProbability}(c) = \theta \cdot (\alpha \cdot \beta^{\gamma \cdot c} \cdot \cos\gamma c + \delta^c + \varepsilon) \tag{2}$$

其中,P_c 表示流中数据包 packet_c所对应的加扰概率值,c 表示数据包的计数值,$\theta,\alpha,\beta,\gamma,\delta,\varepsilon$ 则为调整渐变概率曲线的非负系数.

3.2 基于移动目标防御的动态混淆调度机制

在系统完全静态的条件下采用概率加扰机制能够防御有限时间约束下的 SDN 指纹攻击,但是如果指纹攻击者拥有无限时间资源不断探测目标系统且目标系统一直保持固定静止的状态,攻击者便有可能突破概率加扰屏障,获得目标有效信息. 为了解决此类无限时间可持续型指纹攻击问题,系统需要综合考量性能与代价,主动变化其指纹信息以进一步迷惑攻击者.

如图 3 所示,该机制引入中间动态调度层来实现控制器动态混淆调度. 中间动态调度层主要由数据代理模块、评估模块和调度模块三部分构成. 数据代理模块主要负责代理控制平面与数据平面的状态及指令等信息的交互. 评估模块在接收到数据代理模块统计的控制平面和数据平面的状态信息后,分别评估攻击者成功发动攻击时所造成的损耗与控制器动态切换时产生的代价. 调度模块中调度时间决策组件在攻击损耗与调度代价评估值的基础上可以计算确定最优控制器切换点从而保证单位时间内综合代价最低,而调度执行组件则根据以最优切换时刻为基础的控制器动态混淆调度算法从备份控制器池中选择与当前主控制器不同类型的控制器从而达到主动变换指纹信息的目的. 具体算法流程如算法 3 所示.

算法 3 控制器动态混淆调度机制

Input:The dataset (Successful attack interval time series sample) D
Output:The list of scheduling time series ST_{list}
1:while TRUE do
2: Colletct state information INF from master controller and data plane
3: Estimate scheduling cost C_s and attack loss L_a based on Step2
4: Fit a distribution $F(t)$ based on D
5: if Cannot match any suitable distribution then
6: Scheduling time $T \leftarrow \text{meanValue}(D)$
7: else
8: $T \leftarrow \text{deriveTime}(D, C_s, L_a)$
9: end if
10: The elapsed time since the last controller scheduling $t_{\text{elapse}} \leftarrow 0$
11: while $t_{\text{elapse}} < T$ do
12: if $\text{Percentage}_{\text{Flow}(\text{Counter}<3)} > \text{Threshold}$ in INF(HashTable) then
13: break
14: else
15: Update t_{elapse}
16: end if
17: end while
18: Update $\text{ST}_{\text{list}} \leftarrow \text{ST}_{\text{list}} \cup \min(t_{\text{elapse}}, T)$
19: Start the controller scheduling based on ST_{list}
20:end while

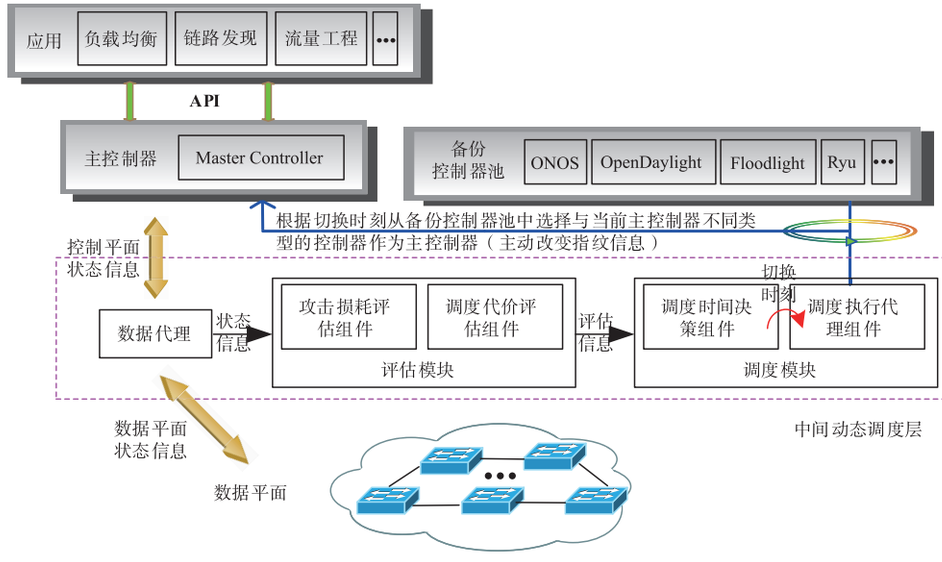


图3 控制器动态混淆调度架构图

为了求解最优切换点,我们构建如下最优调度时间模型:令 T^i_{smart}表示控制器执行完第 i 次调度后出现学习型攻击者所需要的探测时间,$F(t)$ 表示其概率分布函数,$f(t)$ 为该分布的概率密度函数,两者关系如式

(3)所示.

$$F(t) = \int_0^t f(t)\,\mathrm{d}t \tag{3}$$

令 T^i_{defense}表示中间动态调度层所计算的控制器在执行第 $i-1$ 次调度与第 i 次调度之间的理论时间间隔,而 T^i_{actual}表示实际时间间隔. 分析算法 3 可知,流状态轻量判决条件直接影响调度时间,因此,控制器第 i 次实际调度时间如式(4)所示.

$$T^i_{\text{actual}} = \begin{cases} T^i_{\text{defense}}, & \text{if } T^i_{\text{smart}} > T^i_{\text{defense}} \\ T^i_{\text{smart}}, & \text{if } T^i_{\text{smart}} < T^i_{\text{defense}} \end{cases} \tag{4}$$

在满足式(4)第一个不等式的情况下,防御者在出现学习型攻击者之前就主动变化指纹信息,因此防御者在第 i 次调度时的代价仅包括调度代价 C_s;而在满足第二个不等式时意味着防御者在未执行调度前就出现学习型攻击者,这样防御者需要额外承担一次攻击损失后再立即进行调度. 因此,防御者在执行第 i 次调度时的代价如式(5)所示.

$$\text{Cost}_i = \begin{cases} C^i_s, & \text{if } T^i_{\text{smart}} > T^i_{\text{defense}} \\ L^i_a + C^i_s, & \text{if } T^i_{\text{smart}} < T^i_{\text{defense}} \end{cases} \tag{5}$$

分别对控制器第 i 次实际调度时间 T^i_{actual}和执行第 i 次调度时的代价Cost_i 求期望,得到式(6)和(7).

$$\begin{aligned} E(T^i_{\text{actual}}) &= \int_0^{T^i_{\text{defense}}} t\cdot f(t)\,\mathrm{d}t + \int_{T^i_{\text{defense}}}^{\infty} T^i_{\text{defense}}\cdot f(t)\,\mathrm{d}t \\ &= \int_0^{T^i_{\text{defense}}} t\cdot f(t)\,\mathrm{d}t + T^i_{\text{defense}}\cdot(1-F(T^i_{\text{defense}})) \end{aligned} \tag{6}$$

$$\begin{aligned} E(\text{Cost}_i) &= C^i_s P(T^i_{\text{smart}} > T^i_{\text{defense}}) + (L^i_a + C^i_s) P(T^i_{\text{smart}} \leqslant T^i_{\text{defense}}) \\ &= C^i_s (P(T^i_{\text{smart}} > T^i_{\text{defense}}) + P(T^i_{\text{smart}} \leqslant T^i_{\text{defense}})) \\ &\quad + L^i_a \cdot P(T^i_{\text{smart}} \leqslant T^i_{\text{defense}}) \\ &= C^i_s + L^i_a \cdot F(T^i_{\text{defense}}) \end{aligned} \tag{7}$$

控制器动态调度会不可避免地暂时影响网络整体性能和服务质量,该代价主要包括两部分:①控制器与底层交换机重新建立映射时,部分到达数据平面的流会因为控制器动态调度而丢失,从而产生调度代价;②当控制器动态调度时,概率加扰机制中哈希表内的值列表会被刷新(仅保留索引信息)并初始化为 0,这样概率加扰机制会对刷新操作所涉及的流产生性能损失. 因此,控制器动态调度引入的代价如式(8)所示.

$$C^i_s = \omega_s \cdot \kappa \cdot \mu\lambda + \sum_{x\in\Theta}\sum_{y=1}^{100} \text{delay}\cdot P^{\text{Flow}_x}_y \tag{8}$$

式(8)中 ω_s 表示调度代价权重系数,目标网络中交换机数目由 κ 表示,μ 为单个交换机所连接的客户数量,λ 代表用户流的到达速率参数(即用户产生流满足参数为 λ 的泊松分布),Θ 表示控制器调度前概率加扰机制中哈希表内索引集合,x 为流索引,y 表示流内数据包索引,delay 代表数据包延迟期望值,$P^{\text{Flow}_x}_y$ 对应 x 流第 y 个数据包的干扰概率. 同理,攻击损失 L^i_a 定义如式(9).

$$L^i_a = \omega_a \cdot \kappa \cdot \mu\lambda + \sum_{x\in\Theta}\sum_{y\in\Phi} \text{delay}\cdot P^{\text{Flow}_x}_y \tag{9}$$

其中,ω_a 表示攻击损失权重系数,Φ 表示控制器调度前概率加扰机制中哈希表内值列表中计数值集合.

基于以上分析,本文将控制器第 i 次实际调度时间期望$E(T^i_{\text{actual}})$和执行第 i 次调度时的期望代价$E(\text{Cost}_i)$的比值定义为单位时间代价,并以此为指标衡量控制器动态调度效果. 单位时间代价如式(10)所示.

$$\begin{aligned} \xi(T^i_{\text{defense}}) &= \frac{E(\text{Cost}_i)}{E(T^i_{\text{actual}})} \\ &= \frac{C^i_s + L^i_a F(T^i_{\text{defense}})}{\int_0^{T^i_{\text{defense}}} t\cdot f(t)\,\mathrm{d}t + T^i_{\text{defense}}(1-F(T^i_{\text{defense}}))} \end{aligned} \tag{10}$$

为了获得最优控制器动态调度效果,需令调度产生的单位时间代价最小化,因此继续对 $\xi(T^i_{\text{defense}})$求导得到最优时间 T^i_{defense},即:

$$\min\xi(T^i_{\text{defense}}) \Rightarrow \frac{\partial\xi(T^i_{\text{defense}})}{\partial T^i_{\text{defense}}} = 0 \tag{11}$$

4　实验仿真

4.1　概率加扰实验

概率加扰实验环境如图 4 所示,为对比测试概率加扰机制防御效果,首先在控制器不开启概率加扰机制的情况下,令攻击者构造并发送含有不同目的地址的探测 ping 包,其中,向每个目的地址“背靠背”的发送 2 次(即每个流中包含 2 个探测数据包),然后分别统计每个流中第一个探测数据包(FirstPacket,记为 FP)和第二个探测数据包(SecondPacket,记为 SP)的往返时间. 类似地,在控制器开启概率加扰机制时重复上述步骤(参数 $\theta=0.78$,$\alpha=0.2$,$\beta=0.93$,$\gamma=0.4$,$\delta=0.91$,$\varepsilon=0.1$). 两种情况下的往返时延结果如图 5 所示. 分析图 5 可以发现,未部署概率加扰机制时,由于控制器与交换机交互,不同流的第一个数据包和第二数据包存在明显的时延差异;而部署概率加扰机制后,这种时延差异明显模糊化.

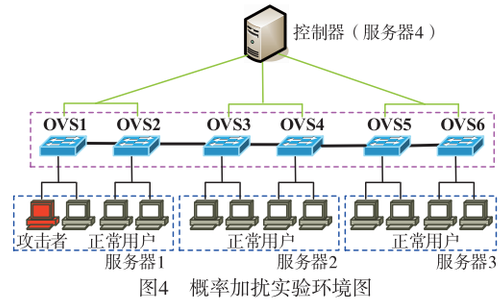
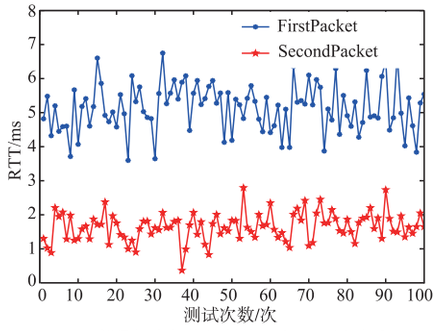
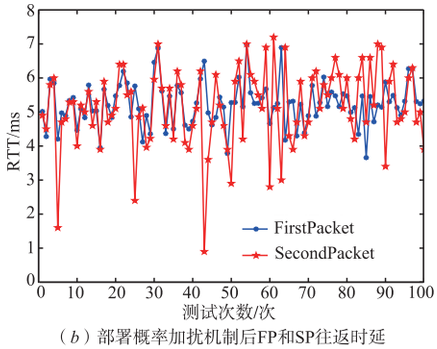


图4　概率加扰实验环境图



（a）部署概率加扰机制前FP和SP往返时延



（b）部署概率加扰机制后FP和SP往返时延

图5 部署概率加扰机制前后往返时延对比图

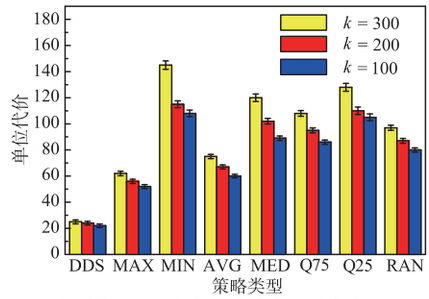
4.2 控制器动态混淆调度实验

为了测试控制器动态混淆调度机制有效性，选择OpenDaylight、Floodlight 和 Ryu 等三种不同类型的控制器组成备份控制器池，每种控制器均部署概率加扰机制；选择公开真实攻击数据集作为输入样本[7]；构建三种不同规模的实验拓扑（分别包含 100、200 和 300 个交换机），每个控制器连接两个用户主机；用户主机按照泊松分布发送数据流. 具体实验参数如表 2 所示.

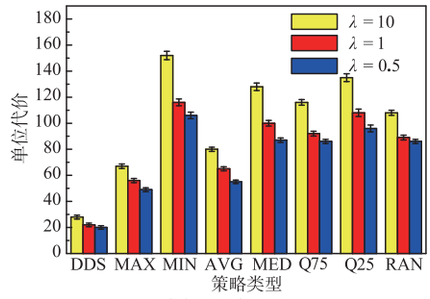
表 2 实验参数设置

符号	含义	数值设置
ω_s	调度代价权重系数	2
ω_a	攻击损失权重系数	1
κ	目标网络中交换机数目	100 ~ 300
μ	单个交换机所连接的主机数量	2
λ	流到达速率参数（泊松分布）	0.5,1,10

为了横向对比控制器动态混淆调度机制的有效性，本文选择了七种常见控制器动态调度策略作为对照实验. 具体而言，这七种控制器动态调度策略分别以攻击时间间隔分布的最大值、最小值、平均值、中位值、上四分位值、下四分位值和随机值作为切换时间点. 上述七种控制器动态调度策略分别简记为符号 MAX，MIN，AVG，MED，Q75，Q25 和 RAN. 根据以上不同的动态调度策略和 3.2 节提出的控制动态混淆机制得到如图 6 所示的单位代价结果.



（a）不同实验拓扑下单位防御代价对比（λ=1）



（b）不同λ下单位防御代价对比（k=200）

图6 不同策略下单位代价对比图

从图 6 可以发现，相较于其他调度策略，DDS 具有最优单位代价值，而且随着拓扑规模扩大（100 ~ 300）及流到达速率升高（0.5 ~ 10），单位代价变化能够相对稳定在合理范围内，不会出现剧烈变化，这是由于 DDS 能够在综合考虑调度代价及攻击损耗的基础上确定最优切换时间点，所以调度效果最好. 而其他策略由于具有盲目性和偶然性，会不同程度的影响防御代价.

为了进一步衡量控制器动态混淆调度机制的防御效果，在交换机数量为 200、流到达速率为 1 的实验拓扑下，以学习型攻击者的攻击方式按照样本攻击时间间隔发动指纹攻击，统计部署控制器动态混淆调度机制前后 RTT 分布曲线，实验结果如图 7 所示. 图中曲线表明，在未部署控制器动态混淆调度机制之前（仅存在概率加扰机制），学习型攻击者通过学习型攻击方式可以明显得到双峰值 RTT 分布曲线（集中在区间[2,3]和[6,7]），而且其 EER 值为 0.67%，表明学习型攻击者完全能可以通过统计结果推测网络类型，甚至是控

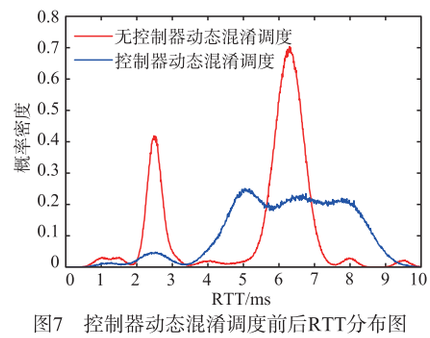


图7 控制器动态混淆调度前后RTT分布图

制器类型. 而当部署控制器动态调度机制之后,攻击者同样采用上述攻击方式探测指纹信息,由于控制器类型一直处于动态混淆过程,不同类型控制器的响应时间相互交织杂糅,使其无明显分布规律,而且其对应 EER 值均接近 50%,因此,攻击者不能提取控制器类型等指纹信息,有效提升指纹信息隐藏度.

5　结论

本文以全要素 SDN 指纹攻击链为基础提出了概率加扰和控制器动态混淆调度防御机制. 通过渐变概率加扰和控制器混淆调度机制综合平衡防御收益与代价,使得系统在有效防御 SDN 指纹攻击的同时将防御开销控制在合理范围内.

参考文献

[1] Mckeown N, Anderson T, Balakrishnan H, et al. Openflow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69 – 74.

[2] Bifulco R, Cui H, Karame G O, et al. Fingerprinting software-defined networks[A]. Proceedings of the IEEE International Conference on Network Protocols[C]. Singapore: IEEE, 2016. 453 – 459.

[3] Shin S, Gu G. Attacking software-defined networks: a first feasibility study[A]. Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking[C]. Hong Kong: ACM, 2013. 1 – 2.

[4] Azzouni A, Braham O, Trang N T M, et al. Fingerprinting OpenFlow controllers: The first step to attack an SDN control plane[A]. Proceedings of the Global Communications Conference[C]. Singapore: IEEE, 2017. 1 – 6.

[5] Sonchack J, Aviv A J, KellerE. Timing SDN control planes to infer network configurations[A]. Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization[C]. New Orleans: ACM, 2016. 19 – 22.

[6] Karame G O, Danev B, Bannwart C, et al. On the security of end-to-end measurements based on packet-pair dispersions[J]. IEEE Transactions on Information Forensics and Security, 2013, 8(1): 149 – 162.

[7] Wang A, Mohaisen A, Chang W, et al. Delving into internet DDoS attacks by botnets: characterization and analysis[A]. Proceedings of the IEEE/IFIP International Conference on Dependable Systems & Networks[C]. Rio De Janeiro: IEEE, 2015. 379 – 390.

作者简介

王　涛　男,1993 年生于山东临朐. 现为中国人民解放军战略支援部队信息工程大学博士研究生. 主要研究方向为 SDN 安全.
E-mail: wangtaogenuine@163.com

陈鸿昶　男,1964 年生于河南新密. 现为中国人民解放军战略支援部队信息工程大学教授、博士生导师. 主要研究方向为新型网络体系架构和网络安全.
E-mail: chc@mail.ndsc.com.cn