一种抗网络侦察的网络指纹在线混淆机制

王彬沣,邢长友,丁 科,许 博

(中国人民解放军陆军工程大学 指挥控制工程学院,江苏 南京 210007)

摘 要: 网络指纹探测是实施网络攻击之前的重要情报获取工作。但作为一种主动对抗指纹探测行为的典型方法,现有的网络指纹混淆技术仍存在部署复杂性高、对端系统不透明,以及对网络性能影响大等问题。为此,基于可编程数据平面技术,提出了一种抗网络指纹探测的分组在线混淆机制P4FO。P4FO利用可编程交换机的灵活分组处理能力,以端系统透明的方式在线混淆网络指纹信息。在分析探测流响应速率特征的基础上,实现了"识别—重构"相结合的网络指纹两阶段混淆方案,支持融主动探测流识别、虚假指纹定制,以及在线指纹混淆等能力为一体,并能够缓解高速网络环境中可编程交换机的资源约束。基于真实网络流量数据集的实验表明,P4FO在对抗网络指纹探测能力方面优于当前主流方法,为网络设备指纹保护提供了一种更为高效的解决途径。

关键词: 网络侦察;网络指纹混淆;网络欺骗防御;可编程协议无关报文处理(P4)

DOI: 10.11907/rjdk.241102 开放科学(资源服务)标识码(OSID):

中图分类号: TP393.08 **文献标识码:** A **文章编号:** 1672-7800(2024)005-0137-09

An Online Network Fingerprint Obfuscation Mechanism Against Network Reconnaissance

WANG Binfeng, XING Changyou, DING Ke, XU Bo

(*Command and Control Engineering College*, *Army Engineering University of PLA*, *Nanjing* 210007, *China*)

Abstract: Network fingerprinting detection is a crucial intelligence-gathering step prior to conducting network attacks. However, existing network fingerprint obfuscation techniques, which are typical countermeasures against fingerprint detection activities, still face issues like high deployment complexity, non-transparency to end systems, and significant impact on network performance. Addressing these concerns, we propose a packet-based online obfuscation mechanism for resisting network fingerprint detection, named P4FO (P4-based fingerprint obfuscation mechanism), leveraging programmable data plane technology. P4FO utilizes the flexible packet processing capabilities of programmable switches to obfuscate network fingerprint information online in a manner transparent to end systems. Building upon analyzing response rate characteristics of probing flows, the mechanism implements a two-phase fingerprint obfuscation scheme combining "recognition-reconstruction", which integrates capabilities of active probing flow recognition, false fingerprint customization, and online fingerprint obfuscation, and it can alleviate resource constraints of programmable switches in high-speed network environments. Experiments based on real network traffic datasets show that P4FO outperforms current mainstream methods in combating network fingerprint detection, offering a more effective solution for the protection of network device fingerprints.

Key Words: network reconnaissance; network fingerprint obfuscation; network deception defense; programmable protocol-independent packet processing(P4)

收稿日期: 2024-03-19

基金项目: 国家自然科学基金面上项目(62172432)

作者简介: 王彬沣(1995-),男,中国人民解放军陆军工程大学指挥控制工程学院硕士研究生,研究方向为网络主动防御;邢长友(1982-),男,博士,中国人民解放军陆军工程大学指挥控制工程学院教授、硕士生导师,研究方向为网络主动防御、软件定义网络和网络性能测量;丁科(1978-),男,博士,中国人民解放军陆军工程大学指挥控制工程学院讲师,研究方向为网络虚拟化技术;许博(1980-),男,博士,中国人民解放军陆军工程大学指挥控制工程学院副教授、硕士生导师,研究方向为网络性能测量、网络流量识别和软件定义网络。本文通讯作者:邢长友。

0 引言

网络指纹探测是一种典型的网络侦察方法，攻击者能够利用该方法侦察目标设备的操作系统、所运行服务等信息，在获取目标情报、制定攻击计划等方面具有重要意义。网络指纹信息主要来自设备底层实现TCP/IP协议栈过程中引入的差异性，例如IP生存时间（TTL）初始值、TCP默认窗口大小、选项字段类型及顺序等。在网络指纹探测过程中，攻击者能够利用Nmap等主动探测工具向目标设备发送指纹探测分组，或者利用p0f等被动探测工具分析网络流量特征，寻找目标分组中的关键字段，匹配指纹库得到目标系统的网络配置信息、操作系统属性、服务信息等。指纹探测往往是攻击者发起网络攻击的第一步，现有研究表明，攻击者利用这些差异性能够在较短时间内准确识别设备及服务类型等[1]。

为了有效对抗攻击者的指纹探测，研究人员引入欺骗防御的思想，通过在网络中策略性地对网络指纹特征进行混淆以误导和迷惑攻击者，达到对抗指纹探测行为的目的[2]。具体而言，网络指纹特征混淆主要消除网络实体的系统、通信协议、应用等指纹信息，使得攻击者在进行指纹探测时获得错误的指纹信息，进而难以基于该信息有效描述探测对象和发起攻击行为[3]。

然而，尽管当前存在一些通过主动修改协议字段、部署蜜罐等方法进行指纹特征混淆的方法，但大多数存在实现方法复杂、支持的变换灵活性不足，以及难以支持高速网络环境中的线速变换等[4]。尤其是被动指纹识别通过被动监听网络流量识别网络资产的指纹特征，不需要向网络目标发送探测数据，导致防御更加困难。以Netfilter为例，尽管它可以在Linux内核修改分组以实现网络指纹混淆，但其配置规则复杂，处理效率相对较低。在蜜罐技术中，虽然通过模拟虚假目标系统可以有效地隐藏真实系统的指纹，但这种技术维护成本高且消耗计算资源大，且存在蜜罐被攻击者识破的风险。在基于OpenFlow的软件定义的网络环境中，尽管可以通过控制平面定制规则实现指纹混淆，但受限于OpenFlow协议约束，其可变换的指纹特征有限。因此，这些混淆方法虽然潜在有效，但在可操作性、资源消耗、安全性等方面还有待提高和优化。

近年来，可编程数据平面技术的发展为网络指纹混淆带来了新的机遇。通过P4（Programming Protocol Independent Packet Processors）等可编程架构设计[5]，可编程数据平面使得网络设备的行为可以根据需求灵活编程。在可编程数据平面中，网络交换机可以根据定义规则执行复杂的分组处理逻辑，包括修改分组首部字段数据、增加新的首部字段等。因此，这一灵活数据处理能力为开展网络指纹混淆带来了便利性，可以针对攻击者的指纹探测行为，在数据平面修改分组首部字段，从而实现定制化混淆网络指纹特征的目标。

基于这一思想，结合网络欺骗防御的基本思想与可编程数据平面技术，本文实现了一种抗网络指纹探测的分组在线混淆机制P4FO（P4-based Fingerprint Obfuscation），支持以可编程的方式线速混淆网络指纹信息，以及混淆指纹的策略化定制等。本文主要贡献如下：①针对主/被动探测带来的网络指纹信息泄露风险，提出了一种网络指纹的透明混淆架构，该架构利用可编程交换机的灵活分组处理能力，支持以端系统透明的方式策略性混淆网络指纹信息，达到欺骗攻击者指纹探测行为的目标；②基于P4可编程交换机实现了网络指纹抗探测机制P4FO，P4FO针对高速流量处理需求和可编程交换机的资源约束，在分析探测流响应速率特征的基础上，设计了“识别—重构”相结合的网络指纹两阶段混淆方案，支持融主动探测流识别、虚假指纹定制，以及在线指纹混淆为一体，并能够缓解高速网络环境中流状态存储的压力。测试结果表明，P4FO能够根据用户定制策略在线混淆网络指纹信息，有效对抗典型的网络指纹探测行为。

1 相关工作

由于大多数安全漏洞都依赖于特定的操作系统或应用服务，因此攻击者在确定攻击目标后，需要了解目标系统的类型、应用版本和配置，以便制定进一步的攻击策略[6]。因此，网络指纹混淆技术通过构建虚假网络指纹欺骗攻击者的探测行为，误导攻击者形成错误认知。为了阻止攻击者识别操作系统指纹，Albanese等[7]使用Linux系统内核模块Netfilter抓取和修改IP、TCP首部及负载的相关信息，降低攻击者的识别准确率。Fan等[8]利用软件定义网络技术，构建了包含诱饵、捕获器、协调器等模块的蜜罐框架，同时实现了更加隐蔽的流量重定向功能。Luo等[9]提出一种MTD和蜜罐相结合的方法以抵御网络指纹探测，动态为设备分配虚拟IP地址，增加攻击者发现设备真实IP地址的时间。Segofensiva等[10]提出的轻量化指纹混淆方法OSfooler，可以在Linux用户端拦截针对用户主机的指纹探测攻击，并自动生成虚假响应报文。Liu等[11]基于可编程数据平面的网络架构，通过在数据平面中实现加密和解密功能，保护IoT设备之间的通信免受窃听攻击。李凌书等[12]通过调整云端资源的容器特性遵循匿名化规则，并构造虚假的云资源视图以提高攻击者网络侦查与嗅探难度。与此相似的是，Givehchian等[13]通过定期改变设备的硬件缺陷实现对移动设备蓝牙物理层指纹的混淆。

然而，尽管已经有不少指纹混淆相关研究，但现有方法大多在端设备上实现，部署复杂性较高，难以适应高度动态的网络环境变化，且对网络通信的性能影响较大，如何以端系统透明的方式高效进行指纹混淆仍面临较多挑战。鉴于此，本文探讨如何利用可编程数据平面技术，建

立一种端系统透明的网络指纹混淆架构,以应对设备多样性和协议栈差异性带来的挑战,并且支持在高速网络环境下为不同设备提供低负载的网络指纹定制混淆能力。

2　基于可编程数据平面的网络指纹在线混淆机制 P4FO

通常情况下,攻击者可以采用主动或者被动等方式探测目标主机的指纹信息,而网络指纹混淆则是策略性地生成虚假指纹信息,诱使攻击者的探测行为获得错误结果。在主动指纹探测过程中,攻击者向目标主机发送探测分组,根据接收到的响应分组推断目标主机特征。而在被动指纹探测过程中,攻击者被动监听目标主机的通信流量,分析分组首部字段,进而推断目标主机特征。与之相对应,网络指纹在线混淆则以在线方式修改分组首部的信息,以达到欺骗攻击者的目的。图 1 给出了这一机制的基本思路,网络指纹混淆设备串接在受保护网络与外部网络之间,按需混淆流经分组所携带的指纹信息,使攻击者难以发现设备的真实指纹信息,实现主机和典型应用等核心资产的特征隐藏。

为此,本文提出并实现了一种抗网络指纹探测的分组在线混淆机制 P4FO,包括主动探测行为分析、指纹库管理、虚假指纹构建等核心组件,图 2 展示了 P4FO 的整体架构。其中,主动探测行为分析组件主要根据收集的常见扫

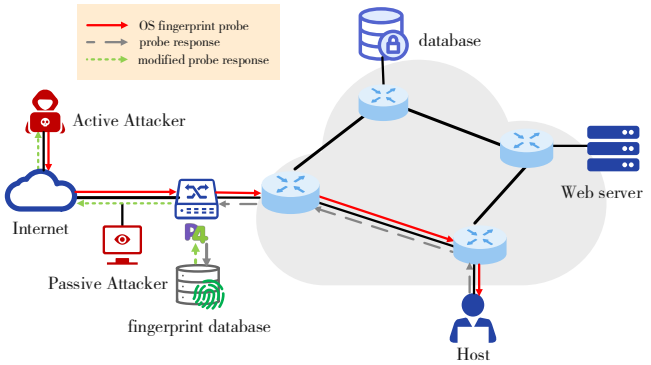

Fig. 1　Network fingerprint detection and obfuscation mechanism

图 1　网络指纹探测及混淆机制

描探测手段,形成探测行为特征库,对收到的分组进行过滤,识别探测分组。指纹库管理组件主要负责维护和更新不同类型指纹探测行为的规则集合,为后续指纹混淆提供支撑。虚假指纹构建组件针对不同类型的探测行为进行指纹混淆处理。由于对网络资产的指纹识别主要是基于服务 Banner 信息和 TCP/IP 指纹特征,因而要实现指纹混淆,必须要能够结合指纹探测所使用的指纹库,策略性地修改分组中特定的字段信息,达到消除原始指纹特征、呈现虚假指纹特征的目标。为此,网络指纹混淆过程会选择要呈现的虚假指纹信息,并基于这些信息重构分组首部字段,以确保构造的分组具有防御者所期望呈现的虚假指纹信息,从而达到迷惑攻击者的效果。

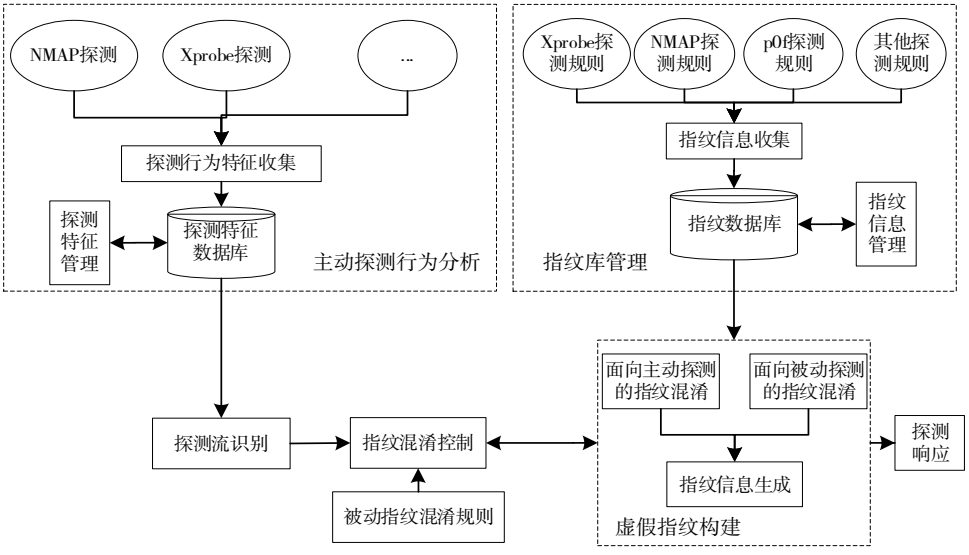

Fig. 2　P4FO network fingerprint obfuscation architecture

图 2　P4FO 网络指纹混淆架构

为了提高网络指纹混淆效率,本文借鉴 NetBeacon 中的状态管理思想,提出了一种“识别—重构”相结合的网络指纹两阶段混淆方案[14]。控制平面下发包含探测流特征的流表项至 P4 可编程交换机,以支持探测流的在线识别。当分组进入配置了特定探测流量检测机制的 P4 可编程交换机时,该交换机将根据控制平面所下发的规则判断是否存在网络探测行为。考虑到网络中正常流量占绝大多数,这一识别方法能够快速找到只包含少数主动探测分组的恶意流量。为了加快虚假响应报文构造,将探测流首部索引存储在交换机寄存器中,用作配置以探测流响应报文为匹配规则的流表策略。最后,指纹构造模块利用之前收集和维护的操作系统指纹数据库,完成重构响应流中分组首部的字段,实现对攻击者的指纹混淆。

图 3 给出了 P4FO 进行网络指纹混淆的核心逻辑:控制平面首先下发指纹混淆策略至 P4 交换机;对于正常流经 P4 交换机的网络业务流量,虚假指纹构造模块主要根

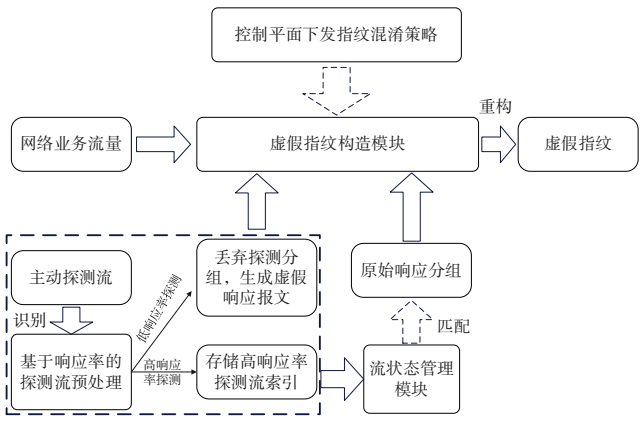

Fig. 3 P4FO network fingerprint obfuscation logic

图 3 P4FO 网络指纹混淆逻辑

据策略混淆网络指纹，以应对被动网络指纹探测。对于攻击者主动发送的探测流，P4FO 在对其进行识别的基础上，根据响应率对探测流作不同的处理。对于高响应率探测，P4 交换机存储该探测流的索引信息，并由流状态管理模块在响应分组中检测与该索引相匹配的分组，一旦找到则根据混淆策略重构分组首部信息，形成虚假指纹；而对于低响应率探测，考虑到存储探测流索引需要耗费交换机的存储资源，且目标主机在接收到这些探测分组后响应的概率很低，因此 P4FO 将直接丢弃这类探测分组，由 P4 交换机根据混淆策略生成具有特定虚假信息的响应分组并回复给攻击者。

结合上述处理逻辑，本文对 P4FO 的功能进行了分解设计，主要包括 3 个核心部分：数据平面处理逻辑、流状态管理模块、指纹构造模块。

2.1 数据平面处理逻辑

数据平面处理逻辑主要根据网络指纹混淆策略，对分组的指纹信息进行处理，在具体结构上由解析器（Parser）、匹配动作表单元（Match Action Unit，MAU）以及逆解析器（Deparser）3 部分配合实现。当分组进入 P4 交换机后，其处理逻辑如图 4 所示。

对于每一个到达 P4 交换机的分组，解析器负责处理按照事先定义的多个 header 结构体，逐层提取其首部并进行解析。

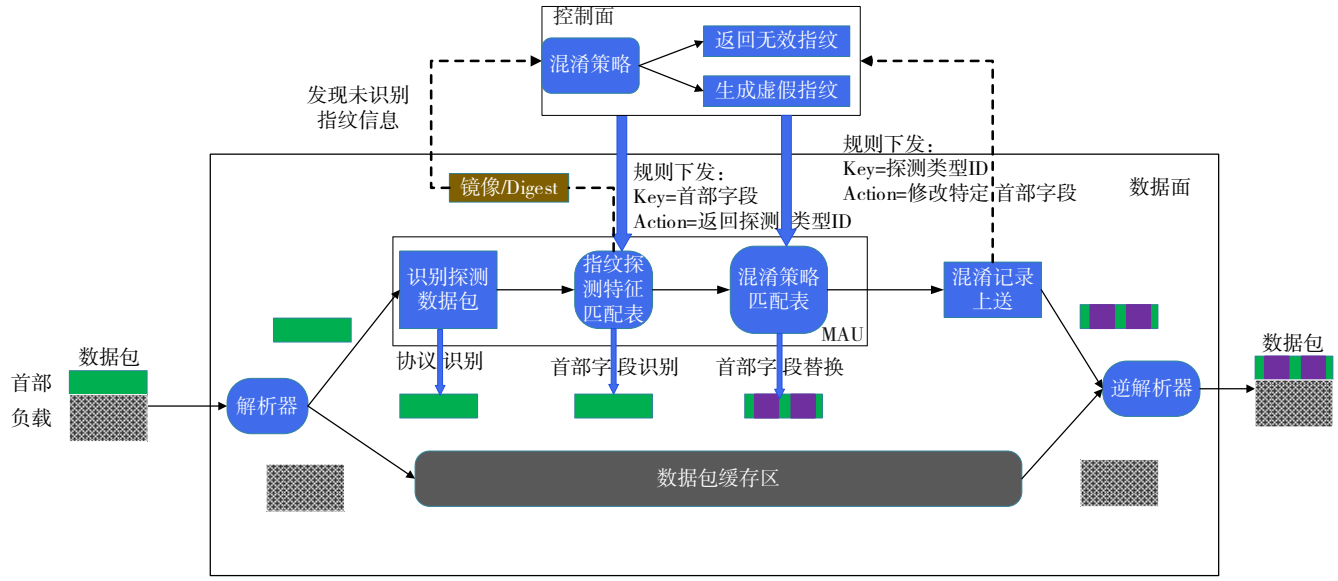

Fig. 4 Data plane processing logic

图 4 数据平面处理逻辑

解析完成后，将相应字段在多个 MAU 中进行匹配，并根据匹配结果，结合预先制定的策略执行指纹混淆操作。由于 P4 交换机的规则容量相对有限，难以容纳所有的处理规则，因此控制器需要有针对性地下发规则。当匹配失败时，需要将分组发送到控制平面，由其决定对该分组启用何种混淆策略，并及时下发新的处理规则，替换掉那些长时间未发生匹配的老化分类规则。在指纹混淆阶段，P4 交换机将维护一张混淆策略匹配表，通过匹配指纹探测特征匹配表输出的探测类型，执行相应的混淆操作，例如首部替换为特定的虚假信息或者完全随机的虚假信息等。

逆解析器部分的操作是解析器部分操作的镜像过程，负责将混淆后的分组首部字段重新与分组载荷部分组合，并将组合后的分组从指定端口转发。

由于 P4 可编程交换机的存储资源有限，一旦发生寄存器空间占满，则交换机内存储的流状态将不再可靠。假设每个规则需要大约 113 字节的存储空间（包括 4 字节源 IP 地址、4 字节目的 IP 地址、2 字节源端口、2 字节目的端口、1 字节协议类型、40 字节的 IP 选项、40 字节的 TCP 选项，以及平均 20 字节的动作参数），在一个拥有 120 MB SRAM 的 Tofino 1 可编程交换机中，可以存储大约 1 113 452 条规则。而对于 IPv6 分组，其更大的地址空间意味着单条规则占用更多存储空间，从而总的规则存储数量会相对减少。考虑到 P4 编程架构内的额外因素，如元数据、状态跟踪组件（例如计数器等），以及特定于交换机架

构的附加开销,实际所需存储空间可能会增加。

为了降低存储开销,本文通过区分高响应率和低响应率的探测流,减少存入交换机的探测流索引总量。本质上,针对主动指纹探测行为,真实系统在响应攻击者不同探测流过程中存在差异性:针对部分探测流的响应率较高,而另外一些探测流的响应率则较低或者不响应。如图5所示,通过对扫描工具Nmap公开的操作系统指纹库中5 702种系统的响应特征进行统计分析后可以观察到,T2-T7类型探测流的最低响应率为21.3%,这表明该类型探测流得到的目标系统反馈较为有限,从而获取的设备指纹信息也相对较少。鉴于此,本方案专注于对Nmap中响应率相对较高的探测流进行状态维护,从而显著降低了对有限状态存储空间的竞争。此外,方案设计也具备一定可扩展能力,旨在适应与Nmap报文特征相似的其他探测工具。如Xprobe2,其核心机制依旧是通过向目标主机发送多种探测分组,并通过分析其响应以推测主机信息。通常,这些探测分组的响应率差异显著,标准的ICMP ping请求得到较高响应率,但特殊定制的ICMP分组或不常用的TCP/UDP端口探测的响应率可能较低。

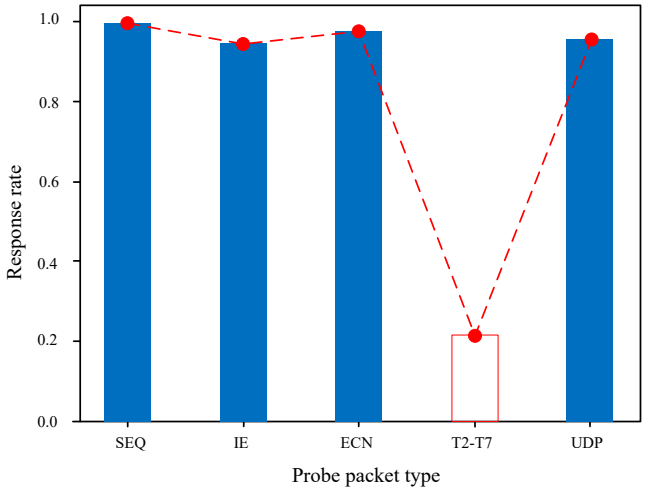

Fig. 5　Nmap probe flow response rate statistics

图5　Nmap探测流响应率统计

高响应率探测报文是数据平面处理逻辑的重要部分,如图6中SYN Probes部分所示,针对高响应率指纹探测包,采用布隆过滤器(Bloom Filter)存储分组首部信息的五元组(源IP地址、目的IP地址、源端口号、目的端口号、协议类型)。流状态存储模块使用布隆过滤器存储来自网络外部的探测分组与响应报文地址信息,从而将之后进入交换机的响应报文与指纹探测流库中的探测流形成唯一映射,最后将确定了所属探测流类型的响应报文交给指纹构造模块重构首部。反之,低响应率的探测报文首部特征无需存储在布隆过滤器中,在经过指纹构造模块处理后直接由交换机的重定向(Redirect)操作响应攻击者。

对于收到的pkt_in分组,数据平面处理逻辑需要查询指纹探测规则表。查询每个规则的复杂度为1,而最坏情况下需要遍历所有规则表,因此其时间复杂度为$O(n)$,n为规则表的数量。而布隆过滤器的插入和查询时间复杂度都为$O(1)$,空间复杂度为$O(m)$,m为布隆过滤器采用的bit类型数组长度。因此,数据平面处理逻辑的时间复杂度为$O(n)$,空间复杂度为$O(m)$。

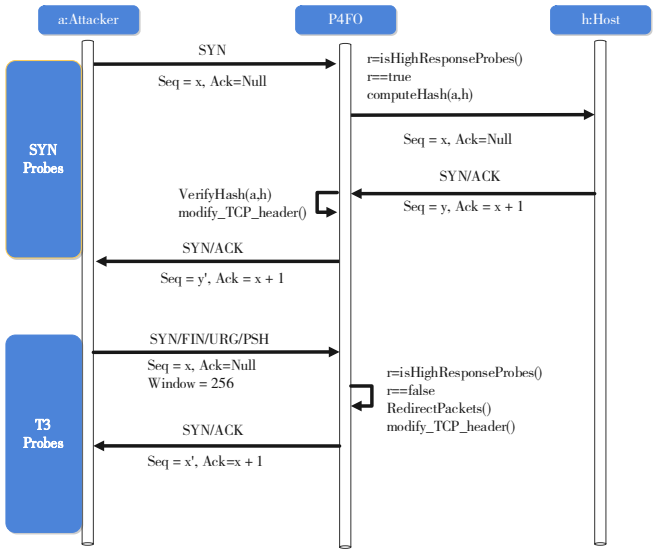

Fig. 6　Handling probing packets with different response rates

图6　针对不同响应率的探测报文

算法1　数据平面处理模块

输入:数据报文pkt_in

输出:数据报文pkt_out

```
1   if pkt_in.probeResponse ==high then //高响应率探测流
2      compute_flow_hash(reg_pos,5-tuple) //分组5元组计算流哈希值
3      bloom _filter.write(reg_pos,1) //存储流哈希值索引
4   else if pkt_in.probeResponse ==low then//低响应率探测流
5      Redirect pkt_in as probe response
6   if pkt_in is probeResponse then //探测流响应报文
7      compute_flow_hash(reg_pos,5-tuple) //分组5元组计算流哈希值
8     if pkt.TIMEOUT//流中最后一个分组超时
9      InitRegs(reg_pos)//初始化布隆过滤器
10   Modify pkt_in header//根据预定义的欺骗策略修改响应报文
```

此外,也可以通过匹配动作表处理高响应率探测流,每当检测匹配探测流,数据平面通过digest通知控制平面,并由控制平面下发相应的流表项。但这种方式的缺点是需要等到表规则在数据平面中生效,与控制平面通信过程中会产生较大时延。与上述匹配操作表的方式相比,P4FO将探测流匹配、分组处理模块实现均卸载至数据平面,避免了来自控制平面的通信时延,确保流量能够线速率转发。通过降低与控制平面的交互需求,P4FO可以显著提高交换机的处理响应速度,从而使指纹混淆机制更加高效。

2.2　探测流状态存储模块

流状态存储模块负责监听来自交换机输入端口的分组,解析分组的首部信息,并根据首部信息判断当前分组的所属流类别,包括何种探测类型、探测机制等。流状态存储模块还负责维护探测流特征库,为了实现线速率流量

分析，需要采用有利于频繁查找和更新操作的高效存储结构。该模块基于布隆过滤器在 P4 交换机中实现一个探测流存储表。具体而言，该模块中初始化状态的布隆过滤器里所有比特位置为 0，当 P4 交换机将输入流的特定字段与系统指纹特征库匹配成功时，将提取分组首部的五元组信息（即源/目的 IP 地址、源/目的端口号及传输层协议），再通过 k 个相互独立的哈希函数 hash1、hash2…hashk，将五元组映射到探测流检测域的对应比特位，并将该位置设为 1。

然而，基于哈希的布隆过滤器结构存在存储冲突问题（见图 7），这种情况会导致数据存储和查找准确率降低，故本文采取必要措施以缓解假阳性率。首先，相较于正常的流量，探测流通常由短流构成，因此本文通过规则匹配的模式区分正常流与探测流，当探测流响应报文输入交换机时，布隆过滤器中探测流的存储索引将重置为初始状态，从而更有效地利用存储资源；其次，当发生冲突时，P4FO 允许新流使用已被占用的寄存器位，如果寄存器中旧流最后一个分组的到达时间已经超过了设定的超时阈值，则允许将新流覆盖存储在寄存器中；最后，寄存器中的一些流信息需要定期删除，以防止冲突发生。

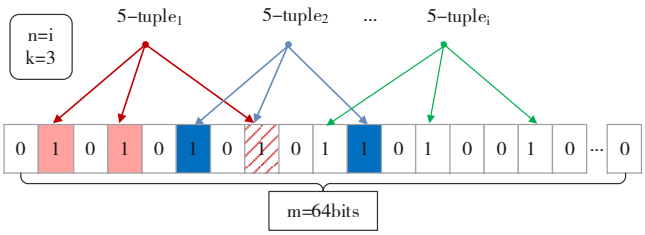

Fig. 7 Probe flow storage table based on Bloom filter

图 7 基于布隆过滤器的探测流存储表

2.3 虚假指纹构造模块

除 IP 和 TCP 首部中固定长度字段所包含的信息外，可变长度的 TCP 选项亦包含对网络指纹识别至关重要的信息。例如，指纹识别工具 p0f[15]、Nmap[16]通过对 TCP 选项的长度、数量和顺序细粒度解析可以大幅提高设备识别准确率。相对而言，在 TCP 选项类型、长度和数量固定，仅在随机化选项顺序的前提下进行操作，系统识别准确率平均下降 10%[17]。

然而，在 P4 可编程交换机上实现 TCP Option 字段操作存在一定困难，因为当前版本的 P416 语言规范并不支持对于可变长字段的修改和查找操作。受 Sherry 使用 header stack 结构存储全部 TCP 选项成员启发，本文扩展使用 header union（联合体）结构管理不同索引位置上的 TCP 选项成员，利用 header union 中多种元素共享存储资源和只能选择其中一个元素的特性，将 7 种常见 TCP 选项类型存储在 header stack 结构中[18]。

虚假指纹构造模块主要根据控制平面下发的混淆策略生成新指纹欺骗攻击者。虚假指纹构造模块主要处理 3 类分组的指纹信息：高响应率探测流的响应分组、低响应率探测流的响应分组和正常业务流量分组。

高响应率探测流的响应分组与低响应率探测流的响应分组均是主动探测式指纹探测行为。以对抗 Nmap 中基于 TCP 初始序列号探测的指纹扫描为例，形成虚假指纹的流程主要分为 3 个阶段：

（1）在分组首次抵达交换机时，先由解析器负责提取分组首部的关键指纹信息，其中包括 IP、TCP 和 UDP 协议的相关字段。P4FO 截取分组首部中含原始指纹特征的信息如 TCP 数据偏移，并将其存储在交换机的元数据（metadata）中待后续使用。

（2）在匹配动作表阶段，针对控制平面下发的混淆规则，交换机对 TCP 首部中的序列号、确认号、窗口大小及 TCP 选项字段进行修改和重排序。并且，在 TCP 选项存在变动时，将新的数据偏移量记录于元数据中。TCP 选项处理逻辑如图 8 所示，根据在 header stack 结构中存储的 header union 类型，TCP 选项从有效位通过 setValid 操作进行修改重排。

（3）在分组首部重构阶段，首先执行校验过程，目的是识别和恢复新指纹中未修改字段的原始值，此校验过程严格依照元数据所保留的原始 TCP 首部信息进行；然后，P4FO 采用恢复后的数据作为依据，对 TCP 和 IP 首部的长度字段进行更新，并对校验和字段重新计算。整个过程中，确保分组在传输过程中保持其结构稳定性和完整性。

3 实验与评估

3.1 实验设置

如图 9 所示，本文选择了两种典型的操作系统指纹探测攻击者：主动探测式和被动监听式。同时，由于指纹识别工具各自的特征提取算法差异和采用不同标准的特征集，因此其结果可能会因所用特征选取方法的差异而有所不同。

为了评估本文提出的指纹混淆机制在实际场景中的性能表现，研究使用网络模拟工具 Mininet 构建一种模拟真实网络环境的拓扑结构[19]。本文指纹混淆机制使用了基于 P4 语言实现的 BMv2 软件交换机，用以创建一个灵活且可编程的实验平台[20]。针对主动攻击场景，选择了已被广泛使用的指纹扫描工具 Nmap-7.94；针对被动攻击场景，选用了被动监听工具 p0f v3.09b 和 PRADS，以模拟攻击者监听并分析网络流量的行为。

为了创建接近实际应用中的网络流量模式，本文采用了 CIC-IDS 2017 数据集以构建流量场景。该数据集提供了广泛的流量类型覆盖，包括从防火墙到交换机、路由器等各类网络设备生成的流量，这种多样化的流量构成确保了实验结果能够更好地反映指纹混淆机制应对复杂网络环境时的性能。

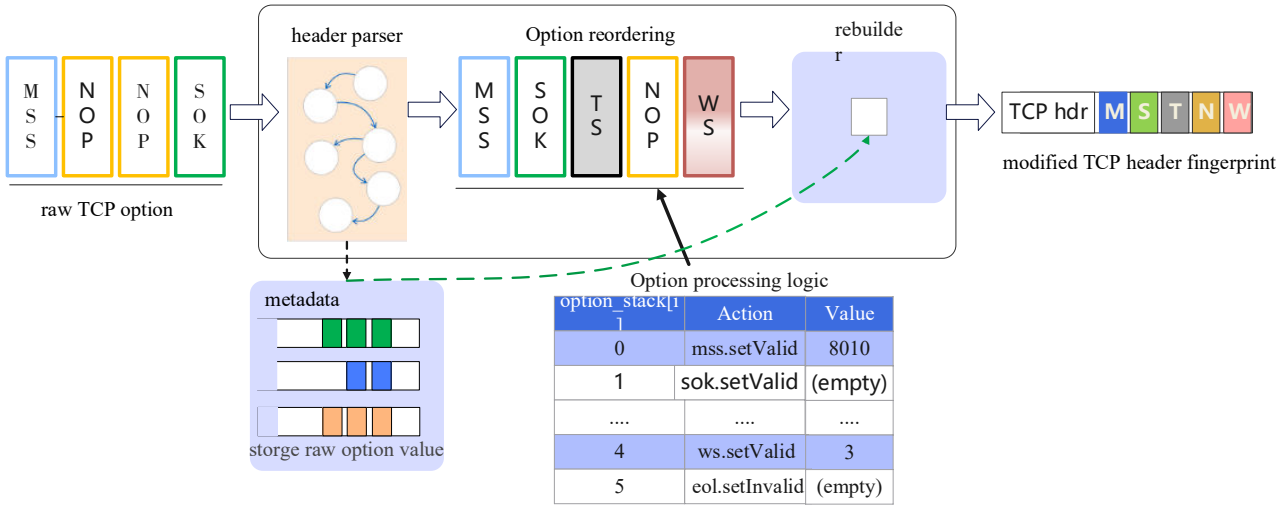

Fig. 8　TCP option fingerprint construction

图 8　TCP 选项指纹构造

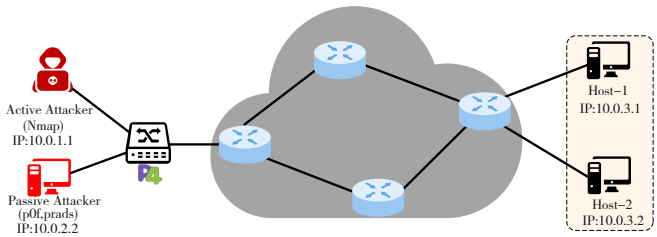

Fig. 9　Experimental network topology diagram

图 9　实验网络拓扑图

为了测试本文指纹混淆机制的效果,Host-1、Host-2 采用 tcpreplay 工具重放 CIC-IDS 2017 数据集中记录的流量模式,并在位于网络边缘的被动监测点部署了两种流量分析工具,即 p0f 和 PRADS,它们构成了被动指纹攻击者的角色。这些工具被用来捕获流经 Host-1 和 Host-2 的网络流量,并分析和推断该主机的设备类型及其操作系统等关键属性。

3.2　防御效果评估

通过对 Host-1 的 SYN 和 ACK 数据包实施混淆防御,并观察其对攻击者设备识别成功率的影响。如图 10 所示,采用指纹混淆机制处理后的 Host-1,即使在 60 s 的连续设备流量监听之后,攻击者的成功率仍然维持在 30%~50%。相比之下,对未经混淆处理的设备 Host-2 进行同样的攻击时,攻击者能够以高达 97% 的准确率正确识别设备类型。还可以观察到,在攻击者监听 TCP 会话的早期阶段,攻击者成功率增长得很快。这揭示了在会话初期阶段实施指纹混淆的关键价值及其高效性,因为这段时间内泄漏了最多的设备特征信息。综上所述,P4FO 指纹混淆机制使得攻击者对设备类型的推断变得不可靠,即便是使用了性能较优的工具如 PRADS,其成功率也仅降低到未设置混淆防御的一半。

随着观察时间的延长,攻击者有更多的机会收集目标设备的指纹特征,进而逐渐提高成功识别目标系统的概率。这是因为,与需要和目标设备建立连接的主动指纹扫描器如 Nmap 不同,p0f 和 PRADS 等被动指纹扫描器通过分析网络流量中抓取的数据包识别设备特征,从而很难被目标设备通过修改响应分组信息轻易混淆。

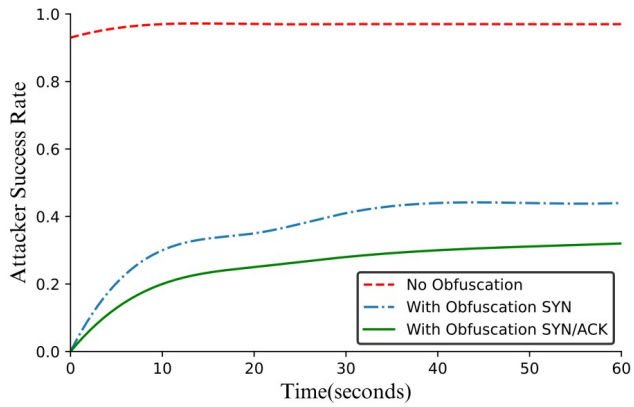

Fig. 10　Passive fingerprint attack obfuscation effect

图 10　被动指纹攻击混淆效果

为了衡量防御主动指纹攻击的效果,本文选择了 Mininet 中的主机 Host-1 作为被攻击目标。此外,实验中使用主动指纹扫描器 Nmap 的探测结果作为衡量指纹混淆成功率的标准。图 11 展示了 8 种基于 P4FO 构造的虚假系统指纹欺骗 Nmap 探测结果,实验结果显示,在 P4FO 的防御机制下,虚假指纹的平均欺骗成功率在 0.91 以上,标准偏差为 0.02。这表明,超过 91% 的攻击者对 P4FO 构建的虚假系统指纹认定为真实;同时,极低的标准偏差 0.02 表明,在不同指纹实验中其较高成功率的稳定性十分显著。

3.3　性能评估

本文将讨论基于 P4 可编程交换机的实验设置,以及因应用指纹变换而引入的额外延迟性能分析。实验中,网络外部主机使用 Hping3 工具的 TCP 模式,通过具有指纹防御机制的 P4 交换机对内网中的 Host-1 和 Host-2 进行时延测量,该过程涉及发送 TCP SYN 标志的包到目标主机的指定端口并观察响应。图 12 显示了部署被动指纹混淆防御

时带来的延迟开销，实验对比无保护、本文P4FO指纹混淆机制、随机周期指纹跳变策略的往返时延（Round-Trip Time，RTT）测试结果。其中，当不部署任何防御机制时，平均RTT值为4.46 ms，而使用了P4FO防御机制及随机周期指纹跳变策略带来的额外延迟开销平均在3.9~4.5 ms。这是由于规则下发、策略更新频率及交换机处理能力影响，故导致了传输时延的增加。由此可见，P4FO能够在不引入显著额外性能消耗的前提下，满足针对异构设备的实时指纹防御要求。

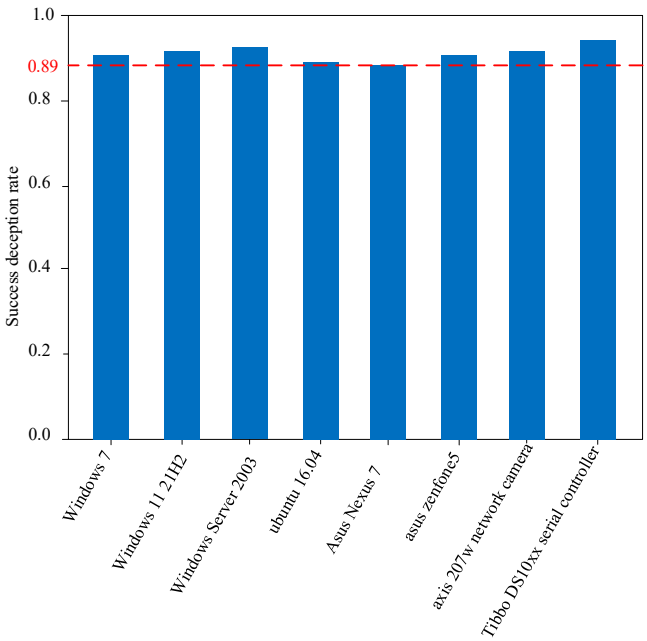

Fig. 11 Active fingerprint defense results

图11 主动指纹防御结果

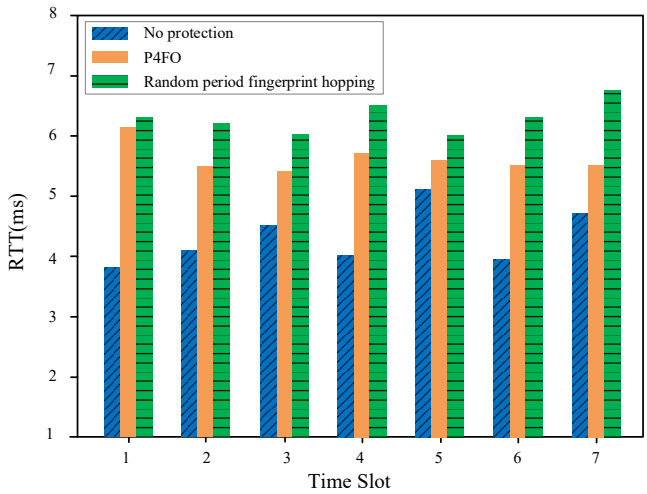

Fig. 12 Passive fingerprint defense delay measurement

图12 被动指纹防御延迟测量

3.4 与已有研究比较

指纹混淆方法比较如表1所示，可以看出，OSfuscate[21]、IPMorph[22]和Massimiliano的方法中，防御机制仅能实现指纹库中较少种类的指纹混淆，这是因为上述3种指纹混淆机制的原理都是基于修改原始端主机探测响应报文。然而，如果目标主机自身屏蔽了这些探测报文，或者网络防火墙设置了相应的流量拦截规则，则不会产生相应的探测响应报文，从而无法构造完整的虚假响应报文，将导致指纹混淆失败。OSfooler的方法中，防御者在收到攻击者扫描请求时完全不依赖于主机的响应报文，通过直接构造响应报文的方式，达到欺骗防御者的效果。蜜罐[6]的方法构造复杂的TCP/IP堆栈指纹，检测到探测攻击时将使用适当的指纹信息进行响应。通过对比分析可知，本文防御机制能够在有效抵御探测攻击时，实时动态地防御目标网络中的系统指纹特征。

Table 1 Fingerprint obfuscation method comparison

表1 指纹混淆方法比较

防御机制	兼容特定系统	指纹库覆盖率	端系统透明	扩展性
OSfuscate	Linux	<60%		
IPMorph	Linux	≈60%		
Massimiliano	Linux	≈60%		
OSfooler	Linux	>90%		√
蜜罐	不限	>90%	√	√
P4FO	不限	>90%	√	√

4 结语

针对攻击者通过网络指纹探测方法获取大量网络情报的问题，本文提出了一种基于P4可编程交换机的网络指纹在线混淆机制P4FO。P4FO充分利用了数据平面可编程带来的分组灵活处理能力，在不修改端系统的情况下策略性地混淆网络指纹信息，有效对抗主动/被动等不同方式的网络指纹探测手段。相较于现有在端系统增加代理等方式的实现，P4FO基于串接到网络边界位置的P4交换机实现，对端系统完全透明，因此能够支持更多的被保护设备类型。并且，P4FO采用两阶段“识别—重构”的方式整合探测流识别、虚假指纹定制和指纹混淆功能，能够在P4交换机资源约束下开展高效的指纹混淆操作，为保护网络指纹信息、避免指纹特征泄露提供了一种解决方案。

在未来工作中，将围绕提升网络指纹定制化能力和策略适应性开展研究，进一步增强系统通用性和对抗未知威胁的能力。此外，考虑到机器学习技术被越来越多地应用于指纹探测，将研究采用对抗学习等手段生成指纹混淆策略，并交由P4交换机执行具体的指纹混淆动作，进而实现更加智能化的网络指纹混淆机制。

参考文献：

[1] ZHU M, ANWAR A H, WAN Z, et al. A survey of defensive deception: approaches using game theory and machine learning[J]. IEEE Communications Surveys & Tutorials, 2021, 23(4): 2460-2493.

[2] ZHANG L, THING V L L. Three decades of deception techniques in active

cyber defense-retrospect and outlook[J]. Computers & Security, 2021, 106:102288.

[3] WU Y H, YANG W L, DU Z H, et al. Similar traffic strategy against website fingerprinting attacks[J]. Software Guide, 2023, 22(8):172-177.
巫咏辉,杨蔚林,杜中辉,等. 抵御网站指纹攻击的相似流量策略[J]. 软件导刊,2023,22(8):172-177.

[4] GUO H Y, SHEN Y. A distributed honeypot deployment strategy based on cooperative game[J]. Software Guide, 2022, 21(9):129-134.
郭昊月,沈勇. 一种基于合作博弈的分布式蜜罐部署策略[J]. 软件导刊,2022,21(9):129-134.

[5] BOSSHART P, DALY D, GIBB G, et al. P4: programming protocol-independent packet processors[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 87-95.

[6] TONG Q, GUO Y F, HUO S M, et al. Progress in diversity research for proactive defense [J]. Journal of Information Security, 2022, 7 (3): 119-133.
仝青,郭云飞,霍树民,等. 面向主动防御的多样性研究进展[J]. 信息安全学报,2022,7(3): 119-133.

[7] ALBANESE M, BATTISTA E, JAJODIA S. A deception based approach for defeating OS and service fingerprinting[C]//2015 IEEE Conference on Communications and Network Security, 2015: 317-325.

[8] FAN W J, DU Z H, SMITH-CREASEY M, et al. Honeydoc: an efficient honeypot architecture enabling all-round design[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(3): 683-697.

[9] LUO X, YAN Q, WANG M, et al. Using MTD and SDN-based honeypots to defend DDoS attacks in IoT[C]//2019 Computing, Communications and IoT Applications (ComComAp), 2019: 392-395.

[10] SEGOFENSIVA J. OSfooler-NG[EB/OL]. https://githubfast.com/segofensiva/OSfooler-ng, 2013.

[11] LIU G, QUAN W, CHENG N, et al. Softwarized iot network immunity against eavesdropping with programmable data planes[J]. IEEE Internet Things, 2021, 8(8): 6578-6590.

[12] LI L S, WU J X, LIU W Y. A method for network deception based on anonymous container fingerprints in a SaaS cloud environment[J]. Journal of Information Security, 2022, 7(2): 72-86.
李凌书, 邬江兴, 刘文彦. SaaS 云环境下基于容器指纹匿名的网络欺骗方法[J]. 信息安全学报, 2022, 7(2): 72-86.

[13] GIVEHCHIAN H, BHASKAR N, REDDING A, et al. Practical obfuscation of BLE physical-layer fingerprints on mobile devices [C]//2024 IEEE Symposium on Security and Privacy, 2023: 73-73.

[14] ZHOU G, LIU Z, FU C, et al. An efficient design of intelligent network data plane[C]//Anaheim: 32nd USENIX Security Symposium (USENIX Security 23), 2023.

[15] ZALEWSKI M. p0f: 3.0.9b[EB/OL]. https://www.kali.org/tools/p0f/.

[16] Nmap-7.94[EB/OL]. https://www.kali.org/tools/nmap/.

[17] HOLLAND J, SCHMITT P, FEAMSTER N, et al. New directions in automated traffic analysis[C]//Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2021: 3366-3383.

[18] BAI S, KIM H, REXFORD J. Passive OS fingerprinting on commodity switches[C]//2022 IEEE 8th International Conference on Network Softwarization(NetSoft), 2022: 264-268.

[19] DE OLIVEIRA R L S, SCHWEITZER C M, SHINODA A A, et al. Using mininet for emulation and prototyping software-defined networks[C]//Bogota: 2014 IEEE Colombian Conference on Communications and Computing, 2014.

[20] NetworksBarefoot. P4-BMv2[EB/OL]. https://github.com/p4lang/behavioral-model.

[21] CRENSHAW A. OSfuscate: change your windows OS TCP/IP fingerprint to confuse P0f, NetworkMiner, ettercap, Nmap and other OS detection tools[EB/OL]. https://www.irongeek.com/i.php? page=security/osfuscate-change-your-windows-os-tcp-ip-fingerprint-to-confuse-p0f-net-workminer-ettercap-nmap-and-other-os-detection-tools.

[22] PRIGENT G, VICHOT F, HARROUET F. IpMorph: fingerprinting spoofing unification[J]. Journal in Computer Virology, 2010, 6(4):329-342.

(责任编辑:孙　娟)