

基于时隙的多重冗余流指纹模型

刘科显¹, 关建峰¹, 张婉澂¹, 何志凯², 闫迪嘉³

(1. 北京邮电大学计算机学院(国家示范性软件学院), 北京 100876;
2. 中国航空综合技术研究所, 北京 100028; 3. 北京工业大学信息学部, 北京 100124)

摘要: 随着互联网的日益广泛应用, 各种网络安全问题频频暴露, 以“打补丁”形式为主的安全增强模式难以有效防范日益增长的安全风险, 网络安全领域研究者认为未来互联网体系架构应当将安全作为一种基本属性, 实现网络架构对安全的内生支持。为了支持内生安全的数据可信, 在流水印/流指纹机制的研究基础之上, 设计实现了一种基于时隙的多重冗余流指纹模型。该模型使用3个时隙间隔, 通过对指定时隙内的数据包进行延迟等操作, 可在相邻比特操作不产生冲突的情况下实现指纹嵌入, 并且引入冗余编码来提高指纹健壮性, 同时考虑到网络中抖动或攻击者的恶意扰乱等行为, 将延迟干扰、垃圾数据包干扰以及丢包干扰对模型的影响进行了理论分析, 结果表明在给定网络流中数据的包分布时, 随着冗余位的增加, 指纹模型的鲁棒性明显提高。为了减少时间和空间消耗, 提高数据包操作的效率和精度, 基于内核设计和实现流指纹原型系统, 并对其效率和鲁棒性进行评估, 实验表明该模型具有很高的鲁棒性。最后, 展示了所提模型的应用场景, 基于该流指纹功能模型, 能有效检测中间人攻击, 预防网络身份欺骗。

关键词: 流水印; 流指纹; 内生安全; 多重冗余

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2023006

Multiple redundant flow fingerprint model based on time slots

LIU Kexian¹, GUAN Jianfeng¹, ZHANG Wancheng¹, HE Zhikai², YAN Dijia³

1. School of Computer Science (National Pilot Software Engineering School,
Beijing University of Posts and Telecommunications, Beijing 100876, China

2. AVIC China Aero-poly Technology Establishment, Beijing 100028, China

3. Faculty of Information Technology Beijing University of Technology, Beijing 100124, China

Abstract: With the increasingly widespread use of the Internet, various network security problems are frequently exposed, while the “patching” style security enhancement mechanisms cannot effectively prevent the growing security risks. The researchers in the field of network security believe that the future Internet architecture should take

收稿日期: 2022-04-13; 修回日期: 2022-11-02

通信作者: 关建峰, jfguan@bupt.edu.cn

基金项目: 国家重点研发计划 (2018YFE0206800)

Foundation Item: The National Key R&D Program of China (2018YFE0206800)

引用格式: 刘科显, 关建峰, 张婉澂, 等. 基于时隙的多重冗余流指纹模型[J]. 网络与信息安全学报, 2023, 9(1): 115-129.

Citation Format: LIU K X, GUAN J F, ZHANG W C, et al. Multiple redundant flow fingerprint model based on time slots[J]. Chinese Journal of Network and Information Security, 2023, 9(1): 115-129.

security as a basic attribute to provide the native security support which is also called as endogenous safety and security. In order to support the data trustworthiness of endogenous security, a time-slot based multiple redundant flow fingerprint model was designed and implemented based on the research of the watermark (or fingerprint) mechanism. The proposed model used only three time slot intervals and operated the packets within the specified time slots, so that the fingerprint can be embedded without conflicting with the adjacent bit operations. Redundant coding was introduced to improve the fingerprint robustness, and the behaviors such as jitter or malicious disruptions by attackers in the network were considered. Furthermore, the impacts of delayed interference, spam packet interference and packet loss interference were analyzed. The analytical results show that the robustness of the fingerprint model improves with increasing redundant bits when the packet distribution in the network stream is given. Besides, in order to reduce the consumption of time and space and improve the efficiency and accuracy of packet operations, a flow fingerprinting prototype system was designed and implemented based on the kernel, and its efficiency and robustness were evaluated. The experimental result show that the model has high robustness. Additionally, the application scenario of the model was elaborated, which can effectively detect man-in-the-middle attacks and prevent network identity spoofing with the help of the flow fingerprinting model.

Keywords: flow watermark, flow fingerprint, endogenous safety and security, multiple redundancy

0 引言

互联网原始设计基于完全可信场景，其核心 TCP/IP 簇在设计之初并不包含安全机制。随着互联网的不断发展，这种可信场景变得不再有效，各种网络安全问题频频暴露，使得互联网安全成为人们所关注的焦点。网络安全领域研究者认为，未来互联网体系架构应当将安全作为一种基本属性存在，而不像现在互联网“外挂”的附加功能存在。

为了缓解上述问题，文献[1]提出了内生安全问题，并为其定义：“如果一个软硬件实体的暗功能和副作用能被某种因素触发而影响到实体服务功能的可信性，则称这些副作用和暗功能为内生安全问题”。综合当前围绕网络安全的相关研究，可以看出内生安全主要围绕身份真实、行为可控、组网安全、数据可信等方面开展，期望达到安全机制的“内嵌原生”。

在网络数据安全方面的预防手段，一般采用被动方式来实现。2001年，Wang等^[2]提出了“流水印”，作为一种主动流量分析手段，在不影响流正常传输的情况下，在流上游以某种隐蔽方式为流打上标识，流下游则根据观察到的标识推测流在传输过程的变化情况。这种不修改流量内容而通过带外标记的主动分析方式，为保障网络数据安全开拓了新思路。为确保内生网络流量安全可

信，本文在研究流水印及其扩展的流指纹机制的基础上，提出了一种改进型的基于时隙的多重冗余流指纹模型，设计实现了一个基于内核的流指纹原型系统，并阐述了该流指纹模型的应用场景。本文主要贡献如下。

1) 基于时隙的多重冗余流指纹模型设计。为提高基于时隙的流水印的鲁棒性，设计了一种冗余编码机制，对每条流都嵌入唯一的指纹信息，提升流指纹应对恶意干扰的能力。

2) 流指纹原型系统的设计与实现。设计了一套基于 Netfilter 的流指纹原型系统，在系统内核进行指纹嵌入、提取和编解码，基于高精度定时器来实现高效实时的指纹嵌入与提取能力。

3) 流指纹系统的评估和应用。搭建原型系统，对流指纹的性能进行测试，分析对比了流指纹系统的性能，并阐述了其应用场景。

1 相关工作

流指纹是在流水印的基础上发展而来的，水印信息仅表明了流被标记与否，而指纹信息则与流本身紧密相关且唯一，除了区别多个不同流以外，还可以根据所需场景关联到流的身份以协助网络安全审计。根据流标记的嵌入位置（或称载波特征），流水印或流指纹机制可划分为三大类：基于数据包内容、基于流速率、基于时间。典型流水印/流指纹载波特征如表1所示。

表1 典型流水印/流指纹载波特征
Table 1 Typical flow watermark and fingerprint carrier characteristics

载波类型	相关工作	功能特征
数据包载荷	Wang 2001 ^[2]	将水印信息比特注入需要发送的信息中，如数分组的载荷或头部，这类方法不适应加密流量且安全性较低
数据包速率	Yu 2007 ^[3] , JIA ^[4] , Huang 2011 ^[5] , Zhang ^[6] , Xu 2018 ^[7]	在整个流的维持时间内进行分段选取，根据水印比特信息调整选取时间段内的流速率，以此来标识流
数据包间隔	Wang 2002 ^[8] , Wang 2003 ^[9] , Houmansadr 2009 ^[10] , Houmansadr 2013 ^[11] , Soltani 2019 ^[12] , Rezaei 2021 ^[13]	通过改变一个或多个分组先后到达或离开某个网络节点的时间差以嵌入水印信息
数据包丢包	YAO ^[14] , Iacovazzi 2018 ^[15-16]	借鉴了网络中自然的丢包行为，在嵌入水印时删除几个特定的数据包，从而改变了数据包间隔的延迟。
基于时隙	Pyun 2017 ^[17] , Houmansadr 2011 ^[18] , Rezaei 2017 ^[19] , Luo 2017 ^[20] , Mo2021 ^[21] , Wang 2007 ^[22] , Lu 2009 ^[23]	随机选定流的一段时间内的数据分组，再按照相同的时隙进行分组，通过调整不同时隙的数据分组的发送时间以嵌入水印信息
基于时隙质心	Hou 2019 ^[24] , Liu 2018 ^[25] , Wang 2009 ^[26] , Wang 2010 ^[27] , Liu 2010 ^[28]	在时隙方法的基础上进一步改进，通过选取分好组后的几个时隙，改变其中部分的数据分组的偏移时间以嵌入水印

基于数据包内容的嵌入出现在流水印的早期研究中，Wang 等早在 2001 年就提出了困倦水印追踪方案，此方案是直接将水印的信息以比特的方式注入数据包中，此手段已无法适应多数流量加密的网络环境中。

基于流速率的方法是在流的整个维持时间内进行分段，微调时间段内流的速率以嵌入水印。文献[3]在 2007 年将直序扩频（DSSS, direct sequence spread spectrum）机制引入水印中，该方法在流的发送端将原始信号进行扩频，再通过微调流速进行水印的嵌入，接收端则进行过滤、降噪等操作提取出水印信息，以此来判断流量是否关联。但这种方法只适用于流速较稳定的环境。文献[4]在此基础上通过对流的时序进行均方自相关而检测水印的存在。文献[5]在 2011 年对 DSSS 水印方法进行了改进，不同于之前的短随机码，作者使用长随机码作为嵌入水印信息，但这种方法仍然是基于流速率的微调，难以避免其固有缺陷。文献[6]在 2010 年提出了一种抗多流攻击的扩展频谱水印方案来跟踪多个网络流，该方案采用多个正交的伪随机码来扩展不同的水印位，具有良好的隐蔽性和健壮性。文献[7]为提高基于流量的 DSSS 流水印技术的抗多流攻击能力，提出了一种基于伪码多样化的扩频流水印技术，消除了水印流之间的依赖性以抵御多流攻击。

基于时间的方法是国内外研究者对水印

方法研究的焦点。此类方法是通过在某个网络节点，调整数据包到达或者离开的时间来嵌入水印信息。基于时间的方法可分基于数据包间隔、基于时隙、基于时隙质心三大类。

数据包间隔就是一对分组先后到达或离开某个网络节点的时间差，基于数据包间隔的水印方法就是通过改变一个或多个分组间隔以嵌入水印信息。文献[8]在 2002 年提出了一种基于数据包间隔的水印方法，此方法不会因为数据流的加密而受到影响，因为分组的时序不会因为数据加密而改变，随后文献[9]进行了改进和细化，主要解决时间扰乱等问题，可以嵌入更多的水印信息，更细的粒度去操作流。文献[10]提出 RAINBOW 方案，该方案基于之前的数据包间隔方案，通过操作很少的数据包就可以嵌入更多比特的水印信息，调节幅度轻微，不易被攻击者察觉。文献[11]通过对 RAINBOW 进一步改进，提出了“流指纹”的概念，除了标记不同流之外还添加了如源、目的地址等额外信息。文献[12]基于包间延迟设计了一种流指纹方案，该方案由发送端生成一个加密的指纹序列，发送端嵌入指纹，接收端根据数据包到达时间提取出指纹，与共享指纹对比，达到了流跟踪的效果。文献[13]通过深度学习的方法生成指纹的延迟序列，使其类似自然的网络抖动，增强了其隐蔽性。文献[15]提出了一种基于隐马尔可夫模型的流量跟踪技术，该技术也是通过

操作包间隔时间而对流嵌入不可见的水印,使用状态转移概率均值作为水印载体,提高了水印的隐蔽性。文献[16]提出了一种基于逆流水印的隐藏服务器识别技术,通过丢包来改变数据包之间的间隔以嵌入指纹,对传统水印技术的攻击具有鲁棒性。同样地,文献[14]通过对网络中数据包进行丢弃,以此来间接修改数据包间延迟以嵌入水印,该方案具有良好隐蔽性。尽管分组间隔方法操作起来相对简单和隐蔽,但是容易受到网络扰动的影响。

基于时隙的方案随机选取一段时间内的数据组,再按照相同的时隙将此段时间的数据组进行分隔,通过调整指定时隙内所有数据包的发送时间以嵌入水印信息。文献[17]在2007年提出了基于时隙的水印(IBW, interval-based watermarking)方法,这种方法针对以往IPD方法无法抵抗数据分组的延迟、丢包等方面有良好的表现。文献[18]提出了对于丢包隐蔽且健壮的弹性水印方法,该方法采用不同的模式操作时隙以嵌入水印信息来抵御多流攻击。文献[19]在2017年引入了纠错码技术进行两阶段编码以增强指纹的健壮性。文献[20]设计了一种基于时隙的流指纹技术,使得每一条流嵌入的指纹唯一且与流本身紧密相联,可有效抵御多流攻击。文献[21]设计了四进制的流水印技术,通过分析流水印在传输过程中的失真情况,判断网络拥塞程度。基于时隙的水印技术对网络的自然扰动以及中间人攻击有良好的表现。

不同于基于时隙的方法,基于时隙质心方法选取部分时隙,通过调整时隙内部分数据分组的偏移时间来嵌入水印信息。文献[23]于2007年提出基于时隙质心的水印(ICBW, interval centroid-based watermark)方法,该方法利用均匀分布原理,通过调整两组时隙内数据分组的时间以嵌入一个水印信息位,可应对流混合、流分割、聚合等扰动。文献[24]基于时隙质心方法提出了一种DSSS改进方法,利用流到达时间的统计值作为特征,提高了不同流速场景的适用性。针对现有基于质心的水印技术适用性不足的问题,文献[25]提出了一种基于区间质心的水印技术,该方法具

有更强的隐蔽性,提高了边缘计算中对多流攻击的抵抗能力。文献[22]利用时间间隔的内部交换和自适应质心量化框架设计了一种高度隐蔽的水印嵌入方法,具有很强的抗网络抖动能力。文献[26-27]也都应用了时隙质心方法,其中文献[27]提出了基于双时隙质心的水印方法来抵御多流攻击,但是此方法时空开销较大,对数据流的追踪效率会明显下降。

通过上述分析可以发现:基于流速的方法在流速率较大、持续时间较长的目标流量中表现较好,基于数据包间隔的方法易受数据包丢失、乱序、重传和时间扰动等多种网络事件影响,而基于时隙的方法将流划分成时隙,对指定时隙内的所有数据包施加延时操作,能够缓解多种扰动的影响。因此,本文设计了基于时隙的多重冗余流指纹模型,并基于内核开发了原型系统,实验结果表明该方法具有较高的效率和较强的鲁棒性。

2 指纹模型设计与实现

2.1 模型设计

基于时隙的多重冗余流指纹模型主要包括流指纹的生成、嵌入和提取。

模型采用随机函数生成指纹,指纹为二进制序列,并把当前系统的时间作为随机种子。

指纹嵌入考虑了冗余编码以抵抗网络中自然的抖动、延迟和丢包,增强其鲁棒性。每一比特的指纹都连续重复 n 次,如当 n 为3时,0重复为000,1重复为111,由于数据流在传输过程会遭受自然的抖动或恶意的攻击,所以在接收端提取指纹时,认为连续的两个相同的比特即最终结果,如001、100、000都解码为0,110、011、111都解码为1。除了考虑每一比特的冗余外,为了防止重复比特之间的干扰,还可以考虑将整个比特重复 r 次,如指纹比特若为1001,当 r 为3时,则编码为1001 1001 1001。

本文流指纹的嵌入只使用3个时隙间隔即可在相邻比特操作不产生冲突的情况下进行指纹的嵌入,即只需要操作3个时隙间隔中的第一个或第二个就能嵌入不同指纹信息。流指纹嵌入过程如图1所示。

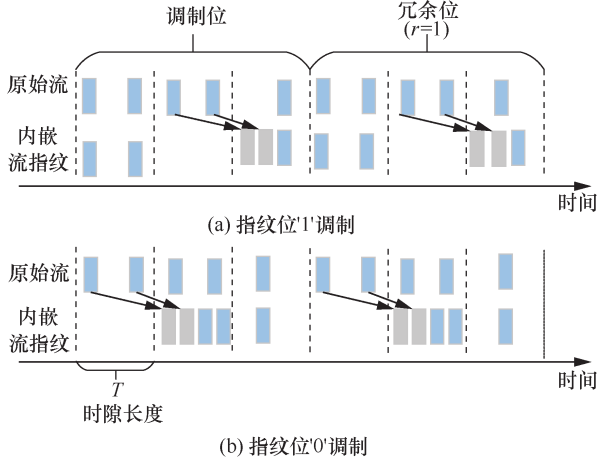


图 1 流指纹嵌入过程
Figure 1 The flow fingerprint embedding process

在嵌入端，用随机函数生成指纹序列后，对每一位指纹比特进行处理。在流的时间偏移量起始处，将流的一部分划分为连续的多个相同的时间间隔，依次每 3 个连续的时间间隔 T 对应一位指纹比特，操作每 3 个时间间隔的前两个，实现比特嵌入，如指纹比特为“0”时，只需将第一个时间间隔内的数据包延迟到第二个时间间隔内发送；指纹比特为“1”时，需将第二个时间间隔内的数据包延迟到第 3 个时间间隔发送。值得注意的是，由于网络中自然的抖动等影响，引入冗余编码 r 来提高指纹的健壮性。

在提取端，遵循流指纹嵌入方案，按照与发送端相同的时间偏移以及时间间隔划分接收到的流，再比较第一个时间间隔和第二个时间间隔内的数据包数量，如果第一个间隔内的数据包数量大于第二个时间间隔内的数据包，则解码为“1”，反之应解码为“0”。

指纹的嵌入和提取步骤可概括为：嵌入端收到流 F_i ，根据该条流的信息用随机函数生成指纹序列 f_i ，然后通过操作 F_i 中数据包的发送时间将指纹 f_i 嵌入流中。指纹的提取端在收到数据流的第一个数据包开始，记录每个数据包到达的时间，从含有指纹的流 F_i' 中，解码出指纹信息 f_i' ，从嵌入指纹 f_i 所用的编码时隙中计算得到 $\bar{Y}_r^{f_i'}$ 的值，将其和 0 进行比较，得到 $Y_r^{f_i'}$ 为正值表明指纹位是“0”，若得到负值表明指纹位是“1”。

2.2 流指纹理论分析

2.2.1 流指纹模型分析

给定一个通用流模型，即令 F 代表一条网络流，其包含一系列的数据分组，表示为 $(P_1, P_2, \dots)$ ，其中，标号规则为这些数据包到达某一网络节点的顺序。从接收到来自流 F 的第一个数据包时间起，经过随机的一个时间偏移 o ($o > 0$) 之后，将流按照固定的时间长度 T ($T > 0$) 进行分片，这些时间片称为时间间隔，标记为 I_i ($i > 0$)。时间间隔 I_i 含有多个连续的数据包，用随机变量 C_i 表示。对于任意 3 个从 F 中随机选择的连续时隙，时间间隔分别为 $I_{1,k}, I_{2,k}, I_{3,k}$ ($k=1, 2, \dots, r$)。其中， r 表示冗余。相应地，这些时间间隔中所含有的数据包的计数为 $C_{1,k}, C_{2,k}, C_{3,k}$ ($k=1, 2, \dots, r$)。

对于一个持续时间足够长，包含足够多的时间间隔的流 F 而言， $C_{j,k}$ ($j=1, 2, 3, k=1, 2, \dots, r$) 满足从一个样本数量足够大的通用分布中进行随机抽样，因此， $C_{j,k}$ ($j=1, 2, 3, k=1, 2, \dots, r$) 之间是独立的，满足独立同分布，有

$$E(C_{j,k}) = \mu_c, D(C_{j,k}) = \sigma_c^2 \quad (1)$$

但是，给定 3 个选定的连续的时间间隔内的数据包不一定满足独立同分布，第二个时隙中数据包数目和 3 个时间按间隔总数据包均值之间的差值为

$$Y_k = C_{2,k} - \bar{C}_k \quad (2)$$

其中，

$$\bar{C}_k = \frac{1}{3}(C_{1,k} + C_{2,k} + C_{3,k}) \quad (3)$$

由于

$$E(\bar{C}_k) = \mu_c, D(\bar{C}_k) \leq \sigma_c^2 \quad (4)$$

因此

$$E(Y_k) = 0, D(Y_k) \leq 16\sigma_c^2/9 \quad (5)$$

由于 $C_{2,k}$ 服从独立同分布，可知 Y_k ($k=1, 2, \dots, r$) 同样服从独立同分布。由于流中数据包之间的时间间隔对齐情况，数据包的数目等可能受网络传输影响发生改变，可以考虑引入冗余编码 r 来提升指纹的健壮性，即选定了 r 份 3 个连续的时间间隔重复嵌入同样的一位指纹比特，可得到样本的均值定义为

$$\bar{Y}_r = \frac{1}{r} \sum_{k=1}^r Y_k \quad (6)$$

由于 $Y_k (k=1, 2, \dots, r)$ 服从独立同分布, 可得

$$E(\bar{Y}_r) = 0, D(\bar{Y}_r) = 16\sigma_c^2 / 9r \quad (7)$$

由上式可知, Y_k 关于均值 0 对称分布。根据计算的结果, $\bar{Y}_r$ 也关于均值 0 对称。

指纹的嵌入, 从结果上来看是改变样本均值 $\bar{Y}_r$ 的值, 即使 $\bar{Y}_r$ 的均值根据嵌入的比特位的值最大增大 μ_c 或减小 μ_c 。具体来说, 当指纹比特为 0

时, $\bar{Y}_r$ 增大 μ_c , 使得 $\frac{1}{r} \sum_{k=1}^r C_{2,k}$ 增大 μ_c , 而在操作时, 整体数据包的数量并不改变, 即 $\bar{C}_k$ 不变, 而是让第一个时间间隔的数据包延迟到第二个时间间隔内, 使得 $C_{2,k}$ 增大 μ_c , 重复这样的操作 r 次; 当指纹比特为 1 时, $\bar{Y}_r$ 减小 μ_c , 使得 $\frac{1}{r} \sum_{k=1}^r C_{2,k}$ 减小 μ_c , 即对 r 份连续的 3 个时间间隔, 将第二个时间间隔内的数据包延迟到第 3 个时间间隔内, 此时 $C_{2,k}$ 减小了 μ_c 。

2.2.2 检测率分析

在流指纹提取时, 可能会因为自然的网络抖动或攻击者的蓄意干扰使指纹比特无法正确提取, 即嵌入的指纹比特为 0, 计算结果的符号为负, 提取为 1; 或者嵌入的指纹比特为 1 时, 计算结果的符号为负, 提取为 0; 则得到的解码指纹错误, 而最终误判了两条流没有关联。因此, 在评估流指纹的性能时, 流指纹检测成功率是重要指标。

指纹检测成功率为一个指纹位被正确解码的概率, 即指纹比特为 0 时, 指纹检测成功率表示为 $\Pr[\bar{Y}_r^f > 0]$, 即符号为正的的概率; 而指纹比特为 1 时, 指纹成功率表示 $\Pr[\bar{Y}_r^f < 0]$, 即符号为负的概率。由于 $\bar{Y}_r$ 关于 0 对称, 因此可以只分析其中一种情况, 另一种情况同理。

通过随机抽样得 $Y_k (k=1, 2, \dots, r)$, 因此嵌入流指纹后得到的 $Y_k^f (k=1, 2, \dots, r)$ 同样是随机抽样的。由于 Y_k^f 的均值和方差有限, 由独立同分布的中心极限定理可知, 当从随机变量中选取的样本规模 r 足够大时, 随机变量 $\bar{Y}_r^f$ 近似地服从标准正态分布, 因此有

$$\Pr\left[\frac{\bar{Y}_r^f - E(\bar{Y}_r^f)}{\sqrt{V(\bar{Y}_r^f)}} > \xi\right] \approx 1 - \Phi(\xi) \quad (8)$$

其中

$$\Phi(\xi) = \int_{-\infty}^{\xi} \frac{1}{\sqrt{2\pi}} e^{-\mu^2/2} d\mu \quad (9)$$

用 $C_{1,k}^f$ 和 $C_{2,k}^f$ 分别表示当嵌入指纹比特为 0 后, 对应原来的时间间隔 $C_{1,k}$ 和 $C_{2,k}$ 的变化结果即

$$C_{1,k}^f = 0, C_{2,k}^f = C_{1,k} + C_{2,k} \quad (10)$$

由式(1)可知

$$E(C_{1,k}^f) = 0, E(C_{2,k}^f) = 2\mu_c (k=1, 2, \dots, r) \quad (11)$$

$$D(C_{1,k}^f) = 0, D(C_{2,k}^f) =$$

$$\sigma_c^2 + \sigma_c^2 + 2\text{Corr}(C_{1,k}, C_{2,k})\sigma_c^2 \quad (12)$$

其中, $\text{Corr}(C_{i,k}, C_{j,k}) = \pm 1 (i \neq j, i=1, 2, 3, j=1, 2, 3)$ 时得到的方差最大, 但具体情况取决于实际网络中流量的分布情况。

因此, $\bar{Y}_r^f$ 服从式(13)分布

$$E(\bar{Y}_r^f) = \mu_c, D(\bar{Y}_r^f) \leq 5\sigma_c^2 / r \quad (13)$$

根据式(8), 指纹检测成功率近似为

$$\Pr[\bar{Y}_r^f > 0] = \Pr\left[\frac{\bar{Y}_r^f - E(\bar{Y}_r^f)}{\sqrt{V(\bar{Y}_r^f)}} > \frac{-E(\bar{Y}_r^f)}{\sqrt{V(\bar{Y}_r^f)}}\right] \approx 1 - \Phi\left[\frac{-E(\bar{Y}_r^f)}{\sqrt{V(\bar{Y}_r^f)}}\right] \geq 1 - \Phi\left[\frac{-\sqrt{r}\mu_c}{\sqrt{5}\sigma_c}\right] \quad (14)$$

式(14)表明, 给定 r , 指纹的检测成功率随 μ_c 减小而降低, 随 σ_c^3 增大而降低, 反之一样。因此, 只要保证 $\mu_c > 0$, 无论 C_i 服从哪种分布类型, 都可以通过增大冗余编码值 r 的方式使得指纹检测成功率趋近于 1。

2.3 恶意干扰分析

2.3.1 时间延迟干扰

时间延迟干扰是数据流在正常传输时所遭受的网络延迟, 如排队延迟、传播延迟以及恶意攻击者形成的延迟等, 时间延迟示意图 2 所示。

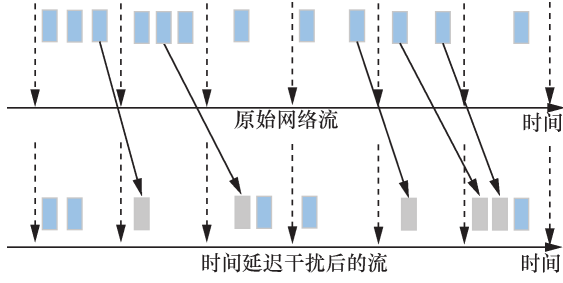


图2 时间延迟示意
Figure 2 Time delay diagram

时间延迟可造成嵌入提取双方的时钟和时间间隔出现偏差，并会扰乱嵌入指纹后的数据流在每个时隙中数据包的数量，从而导致指纹的错误提取。部分因网络抖动引起的延迟是独立分布的，假设攻击者施加的时间延迟相对于嵌入指纹的流是随机的，并且不知道指纹嵌入的具体位置。 q 表示一个随机变量或值， q^f 表示嵌入指纹后的随机变量， q^p 表示在出现了时间延迟干扰后的随机变量。令 $D(D>0)$ 表示攻击者可对数据包引入的最大延迟，令 $P_i (i>0)$ 表示受到时间干扰后某个时间间隔内数据包增大或减小的值，并且假设相对嵌入指纹的流是独立同分布的，即 $\text{Corr}(C_{j,k}, P_{j,k})=0$ 。该随机变量的期望假设为 μ_p ，方差设为 σ_p^2 ， P_i 之间是独立同分布的，则发生延迟干扰后每个时隙的数据包数量可表示为

$$C_{1,k}^p = C_{1,k}^f + P_{0,k} - P_{1,k} \quad (15)$$

$$C_{2,k}^p = C_{2,k}^f + P_{1,k} - P_{2,k} \quad (16)$$

$$C_{3,k}^p = C_{3,k}^f + P_{2,k} - P_{3,k} \quad (17)$$

可得

$$\overline{C_k^p} = \frac{1}{3}(C_{1,k}^p + C_{2,k}^p + C_{3,k}^p) = \overline{C_k} + \frac{1}{3}P_{0,k} - \frac{1}{3}P_{3,k} \quad (18)$$

因此，在嵌入比特为0时，有

$$\begin{aligned} E(C_{2,k}^p) &= 2\mu_c \\ D(C_{2,k}^p) &\leq 4\sigma_c^2 + 2\sigma_p^2 \end{aligned} \quad (19)$$

$$E(\overline{C_k^p}) = \mu_c$$

$$D(\overline{C_k^p}) \leq \sigma_c^2 + \frac{2}{9}\sigma_p^2 \quad (20)$$

根据式(2)得

$$Y_k^p = C_{2,k}^p - \overline{C_k^p} \quad (21)$$

因此有

$$\begin{aligned} E(\overline{Y_r^p}) &= \mu_c \\ D(\overline{Y_r^p}) &< \left(5\sigma_c^2 + \frac{20}{9}\sigma_p^2\right)/r \end{aligned} \quad (22)$$

在此情况下，流指纹的成功检测率可表达为

$$\begin{aligned} \Pr[\overline{Y_r^p} > 0] &= \Pr\left[\frac{\overline{Y_r^p} - E(\overline{Y_r^p})}{\sqrt{V(\overline{Y_r^p})}} > \frac{-E(\overline{Y_r^p})}{\sqrt{V(\overline{Y_r^p})}}\right] \approx \\ 1 - \Phi\left[\frac{-E(\overline{Y_r^p})}{\sqrt{V(\overline{Y_r^p})}}\right] &\geq 1 - \Phi\left[\frac{-\sqrt{r}\mu_c}{\sqrt{5\sigma_c^2 + \frac{20}{9}\sigma_p^2}}\right] \end{aligned} \quad (23)$$

假设每个时隙延迟的数据包数量服从泊松分布，图3结果表明，在每个时隙内的延迟率（延迟的数据包与原时隙内数据包数量的比值，此处取0.4）一定时，当原始数据流中每个时隙内数据包数量的期望值增加或方差减少时，指纹的检测率会有所增高，这是因为分布相对均匀的数据流抵抗延迟干扰的能力较强，而指纹的检测成功率随着冗余位的增加有所增高，所以在数据包分布的期望和方差一定时，可以适当提高冗余位来提高指纹的检测成功率。

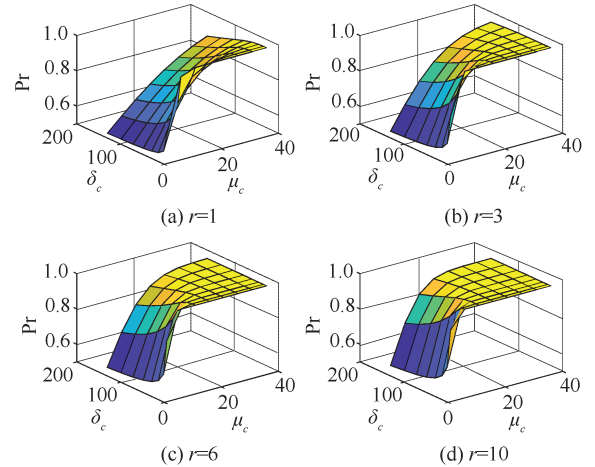


图3 延迟干扰下的检测成功率
Figure 3 Detection success rate under delayed interference

2.3.2 垃圾数据包干扰

一些攻击者为了破坏流指纹的嵌入，可能会实施在网络流中注入虚假垃圾数据包的方法，图4为垃圾数据包添加示意。

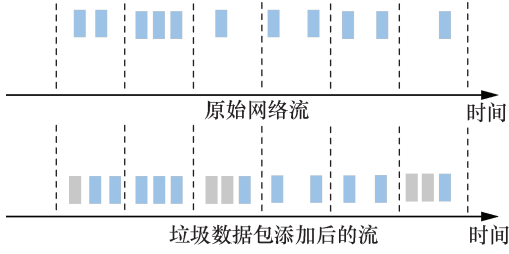


图4 垃圾数据包添加示意
Figure 4 Spam packet addition diagram

假设这些注入的数据包相对于嵌入的指纹流随机且独立，并且注入操作是在流进行了指纹嵌入之后以及指纹提取之前， q 表示一个随机的变量或值， q^f 表示嵌入了指纹后的随机变量， q^A 表示注入垃圾包后的随机变量， $A_i (i>0)$ 为一随机变量，表示在时间间隔 $A_i (i>0)$ 内添加的垃圾数据包的数量。同样地，不给定 A_i 遵循某种分布情况，但假设其期望为 μ_a ，方差为 σ_a^2 ，由于其相对于指纹流随机且独立，有 $\text{Corr}(C_{j,k}, A_{j,k})=0$ ；与 C_i 相同，假设 A_i 之间是独立同分布的，则有

$$\begin{aligned} C_{1,k}^A &= C_{1,k}^f + A_{1,k} \\ C_{2,k}^A &= C_{2,k}^f + A_{2,k} \\ C_{3,k}^A &= C_{3,k}^f + A_{3,k} \end{aligned} \quad (24)$$

可得

$$\begin{aligned} \overline{C_k^A} &= \frac{1}{3}(C_{1,k}^A + C_{2,k}^A + C_{3,k}^A) = \\ &= \overline{C_k} + \frac{1}{3}(A_{1,k} + A_{2,k} + A_{3,k}) \end{aligned} \quad (25)$$

因此，嵌入比特为 0 时有

$$\begin{aligned} E(C_{2,k}^A) &= 2\mu_c + \mu_a \\ E(\overline{C_k^A}) &= \mu_c + \mu_a \end{aligned} \quad (26)$$

$$\begin{aligned} D(C_{2,k}^A) &\leq 4\sigma_c^2 + \sigma_a^2 \\ D(\overline{C_k^A}) &\leq \sigma_c^2 + \frac{1}{3}\sigma_a^2 \end{aligned} \quad (27)$$

由式(2)得

$$Y_k^A = C_{2,k}^A - \overline{C_k^A} \quad (28)$$

因此有

$$\begin{aligned} E(\overline{Y_r^A}) &= \mu_c \\ D(\overline{Y_r^A}) &< \left(5\sigma_c^2 + \frac{2}{3}\sigma_a^2\right)/r \end{aligned} \quad (29)$$

则在此情况下的指纹检测成功率可以为

$$\begin{aligned} \Pr[\overline{Y_r^A} > 0] &= \Pr\left[\frac{\overline{Y_r^A} - E(\overline{Y_r^A})}{\sqrt{V(\overline{Y_r^A})}} > \frac{-E(\overline{Y_r^A})}{\sqrt{V(\overline{Y_r^A})}}\right] \approx \\ 1 - \Phi\left[\frac{-E(\overline{Y_r^A})}{\sqrt{V(\overline{Y_r^A})}}\right] &\geq 1 - \Phi\left[\frac{-\sqrt{r}\mu_c}{\sqrt{5\sigma_c^2 + \frac{2}{3}\sigma_a^2}}\right] \end{aligned} \quad (30)$$

假设每个时隙添加的数据包服从泊松分布，且添包率为 0.4。垃圾数据包在注入理论上对每个时隙内数据包数量之差没有太大的影响。原始数据流与图 3 设定相同的情况下，结果与图 3 类似，如图 5 所示，也可以通过增加冗余 r 的值来提升指纹的健壮性。

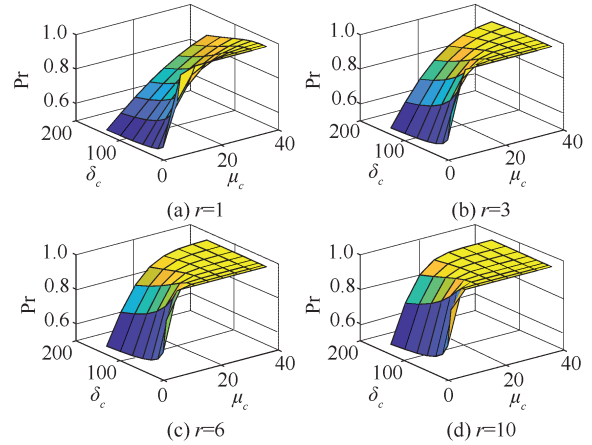


图5 添包干扰下的检测成功率
Figure 5 Detection success rate under packet addition interference

2.3.3 丢包干扰

数据包丢失影响流指纹性能，这种干扰一是来自正常网络通信中的拥塞等，二是来自攻击者的蓄意破坏。丢包示意如图 6 所示。

若 q 表示一个随机的变量或值， q^f 表示嵌入了指纹后的随机变量，则 q^L 表示在出现了数据包丢失干扰之后的随机变量。 $L_i (i>0)$ 为一随机变量，表示在时间间隔 $L_i (i>0)$ 内丢弃数据包的数量。同样地，不给定 L_i 遵循某种分布情况，但假设其期望为 μ_l ，方差为 σ_l^2 ，并且由于其相对于指纹流随机且独立，则有 $\text{Corr}(C_{j,k}, L_{j,k})=0$ 。与 C_i 相同，假设 L_i 相互之间是独立同分布的。假设丢包发生在流嵌入指纹之后，提取指纹之前，当嵌入指纹比特为 0 时，第一个时隙无数据包，则无法进行丢包，所以有

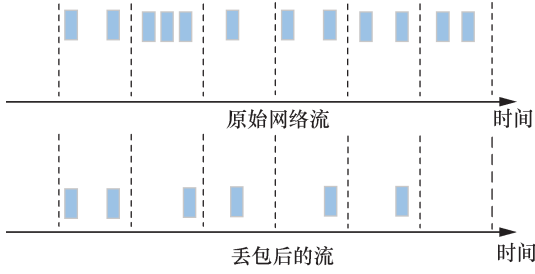


图 6 丢包示意
Figure 6 Packet loss schematic

$$\begin{aligned} C_{2,k}^L &= C_{2,k}^f - L_{2,k} \\ C_{3,k}^L &= C_{3,k}^f - L_{3,k} \end{aligned} \quad (31)$$

可得

$$\begin{aligned} \overline{C_k^L} &= \frac{1}{3}(C_{1,k}^L + C_{2,k}^L + C_{3,k}^L) = \\ &= \overline{C_k} - \frac{1}{3}(L_{2,k} + L_{3,k}) \end{aligned} \quad (32)$$

进一步可得

$$\begin{aligned} E(C_{2,k}^L) &= 2\mu_c - \mu_l \\ D(C_{2,k}^L) &\leq 4\sigma_c^2 + \sigma_l^2 \end{aligned} \quad (33)$$

$$\begin{aligned} E(\overline{C_k^L}) &= \mu_c - \frac{2}{3}\mu_l \\ D(\overline{C_k^L}) &\leq \sigma_c^2 + \frac{2}{9}\sigma_l^2 \end{aligned} \quad (34)$$

根据式(2)可得

$$Y_k^L = C_{2,k}^L - \overline{C_k^L} \quad (35)$$

因此有

$$\begin{aligned} E(\overline{Y_r^L}) &= \mu_c - \frac{1}{3}\mu_l \\ D(\overline{Y_r^L}) &< \left(5\sigma_c^2 + \frac{5}{9}\sigma_l^2\right)/r \end{aligned} \quad (36)$$

则在此情况下的指纹检测成功率可以为

$$\begin{aligned} \Pr[\overline{Y_r^L} > 0] &= \Pr\left[\frac{\overline{Y_r^L} - E(\overline{Y_r^L})}{\sqrt{V(\overline{Y_r^L})}} > \frac{-E(\overline{Y_r^L})}{\sqrt{V(\overline{Y_r^L})}}\right] \approx \\ &= 1 - \Phi\left[\frac{-E(\overline{Y_r^L})}{\sqrt{V(\overline{Y_r^L})}}\right] \geq 1 - \Phi\left[\frac{-\sqrt{r}\left(\mu_c - \frac{1}{3}\mu_l\right)}{\sqrt{5\sigma_c^2 + \frac{5}{9}\sigma_l^2}}\right] \end{aligned} \quad (37)$$

同样，原始数据流与图 3 设定相同，并假设每个时隙内丢包数量服从泊松分布且丢包率为 0.4 的情况下，整体情况与图 3 结果类似，如图 7 所示。在丢包期望值的 1/3 小于原始数据流的期望值时，指纹的提取成功率是有保障的，在上述情况下，也可通过增加冗余位 r 来提高流指纹的检测成功率。

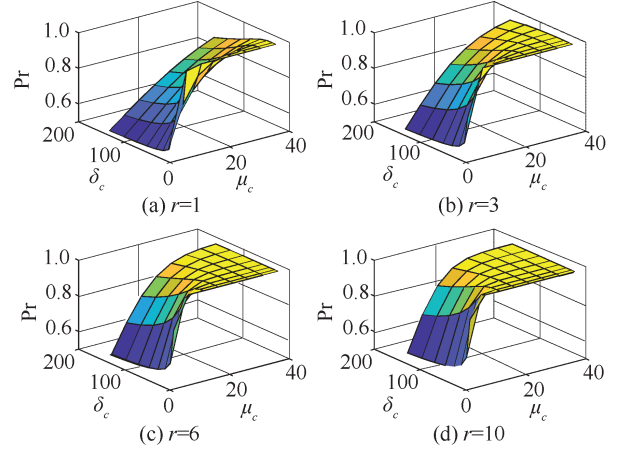


图 7 丢包干扰下的检测成功率
Figure 7 Detection success rate under packet loss interference

2.4 系统实现

当前典型的流水印方法（如 IBW、ICBW）在原型系统实现时都是使用 iptable 配合 libnetfilter_queue 用户空间接收内核中的数据包^[17,23]，在用户空间对数据包进行延迟等操作后，重新将数据包注入操作系统内核进行后续的转发等操作，由于这种方式会频繁进行操作系统用户态和内核态之间的上下文切换和数据包复制，势必会增加处理的时间和空间消耗。此外，在时间操作的精度方面，用户空间使用如 alarm、settimer 等定时器最高可达到毫秒级别的控制，但内核中的 hrtimer 可达到纳秒级别的时间控制，而且在等待时不占用 CPU，适用于多队列的并行操作，显著提高对数据包操作的效率和精度。因此，本文在操作系统的内核层设计并实现了流指纹原型系统。

2.4.1 系统总体设计

流指纹原型系统模块层次如图 8 所示，具体如下。① 系统初始化：包括设定模块运行模式、初始化参数。② 参数设置模块：流指纹的参数应该在嵌入端和提取端协商共享参数，本文设计时简化了双方的协商，在两端分别设定同样的参

数。③ 指纹编解码模块：设定参数后，设定不同的运行模式，对流进行指纹的编码和解码。④ 数据包管理模块：按照指定的规则对数据包进行捕获和发送给指纹的嵌入/提取模块。⑤ 指纹的嵌入/提取模块：根据生成的指纹序列，更改数据包发送的时间。

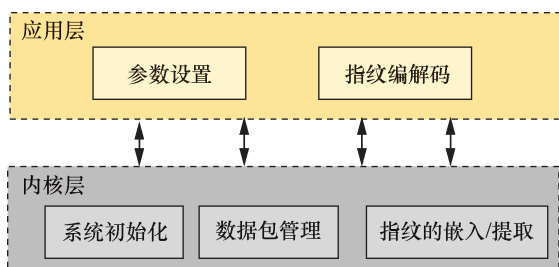


图8 流指纹原型系统模块层次

Figure 8 Modular hierarchy of the stream fingerprint prototyping system

2.4.2 系统实现

指纹系统实现包括应用层和内核层，应用层主要作为交互的界面，进行参数的增加、查询等以及相应流的指纹生成和指纹的冗余编码，内核与应用层之间利用 `ioctl` 函数进行交互。

系统初始化、数据包管理、指纹嵌入提取等操作都是在内核中完成的。指纹嵌入和提取主要对数据包进行捕获，再按照重新设定的时间进行发送。内核数据包捕获基于 `Netfilter` 框架开发，实现数据包的记录、网络地址或端口的转换、数据包过滤和其他包的控制等。本文利用 `Netfilter` 的钩子来进行数据包处理，通过内核模块在 `Linux` 网络栈的不同位置注册钩子回调函数，并在同一个位置注册多个函数。每个经过钩子处的数据包都会根据规则调用注册的回调函数进行响应处理。

流指纹嵌入/提取模块主要任务是将队列中的数据包按照指纹编码设定的时隙规则进行延迟操作。指纹嵌入时，首先记录时间间隔开始的时间，即偏移量 o ，再判断运行模式：若为嵌入模式，则初始化包发送标志为 0，判断是否结束嵌入，若结束指纹的嵌入，则释放队列中的后续数据包；否则让队列继续进行包的缓存，发送完一个时隙的数据包后重新记录时间间隔的开始。流指纹提取时，同样判断运行模式，若是提取模式，则判断队列时间间隔长度内的

数据包数量是否大于门限值；当大于门限值时可判定信号为 1，将信号解码到指纹比特；再判断是否得到了完整的指纹，若得到了完整比特则结束指纹的提取，释放队列中后续的数据包；若没有得到完整指纹则重复上述步骤，直到得到完整指纹比特。

3 指纹模型评估与应用

3.1 模型评估

3.1.1 评估环境

为对模型进行评估，本文搭建如图 9 所示的网络拓扑进行实验。用户 1、用户 2 为两个实体笔记本终端，分别运行 `iperf3` 软件，从而在两端产生交互的流量以用于流指纹的嵌入和提取；其余路由器节点使用 `VMware` 搭建的虚拟节点；各个路由器节点之间运行 `OSPF` 路由协议。在接入网关 1 上实施指纹的嵌入，接入网关 2、3 上进行指纹的提取。中间路由器 R322 在实验中利用 `TC` (traffic control) 模拟网络中的延迟丢包等干扰行为。

在两个终端上使用 `iperf3` 进行 `TCP` 流量的传输，用户 1 是发送端，用户 2 和用户 2' 是接收端。`iperf3` 通过不同端口发送 10 条流，带宽为 60 Mbit/s。

在实验评估中，测试了两种场景。场景 1 中流指纹传递跨越一个路由器（不包含嵌入端和提取端，标记为 $R=1$ ），即只在图 9 中的区域 1 内传递。场景 2 中流指纹传递跨越 4 个路由器（标记为 $R=4$ ），即从区域 1 传递到区域 2。在两种场景下，启用与嵌入提取无关的背景流（用 `bf` 表示）验证，其中有 6 条 `VLC` 播放器的流，4 条 `SSH` 流，图 9 中红色箭头表示背景流和 `iperf3` 测试流的混合流。

3.1.2 结果分析

基于时隙间隔的流指纹方案主要依赖于双方共享参数中的时间起始偏移 o 和时间间隔长度 T ，由于真实网络环境的各种扰动会导致 o 和 T 的变化，所以要找到最优的时间偏移 o 和时间间隔长度 T 的值。为了找到较优的时间偏移 o 和间隔长度 T ，首先在 $R=4$ ，`bf=0` 的情况下进行测试，即用户 1 和用户 2' 之间进行交互流量收发，如表 2 所示。

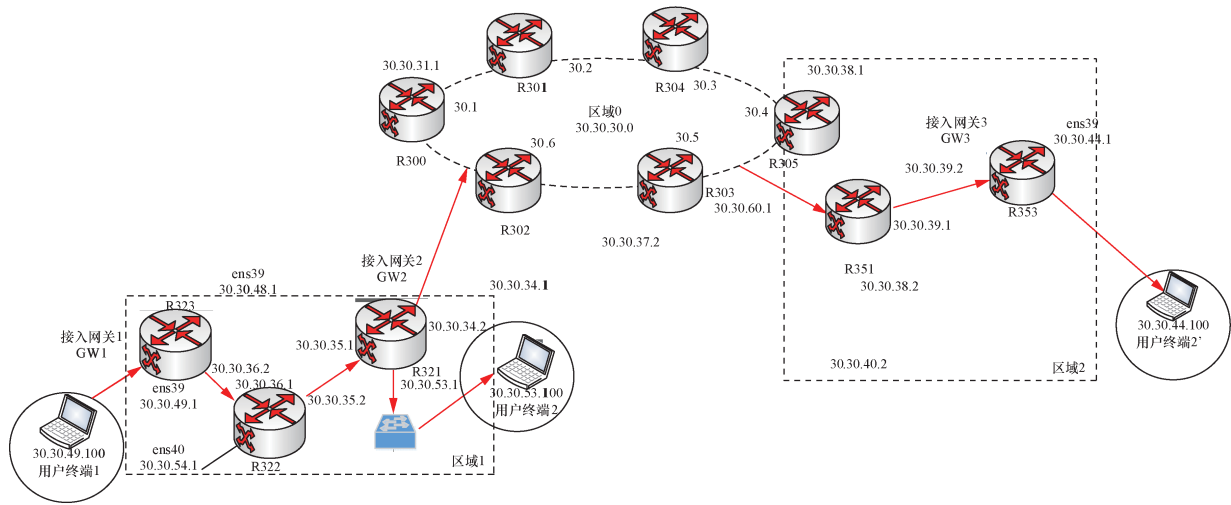


图 9 实验用网络拓扑

Figure 9 Experimental network topology

表 2 指纹参数的设置
Table 2 Setting of fingerprint parameters

参数	值
指纹长度(l)	16 bit
时间间隔长度(T)	200 ms, 250 ms, 300 ms, 350 ms, 400 ms, 450 ms, 500 ms
时间偏移(o)	600 ms, 800 ms, 900 ms

设置不同时间间隔 T 和时间偏移 o ，进行多组实验，结果如图 10 所示。当时间偏移 o 固定时，随着时间间隔 T 逐渐增加，指纹的检测成功率也逐渐增高。当时间间隔 T 不变时，检测率也随着时间偏移 o 的不断增大而逐步提高。显然，时间偏移为 900 ms 时，整体性能高于其他两种情况。因此，本文在后续的实验中选择时间偏移 o 为 900 ms，时间间隔为 400 ms。

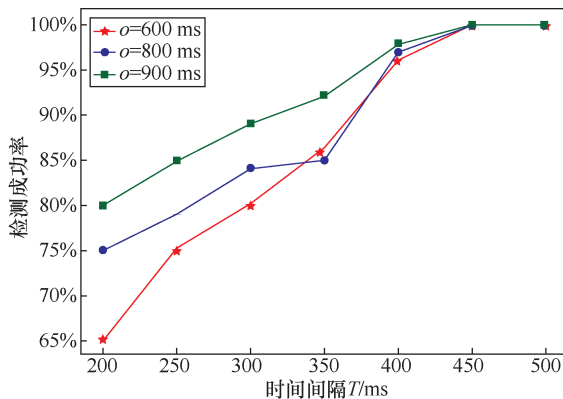


图 10 不同参数下指纹检测成功率

Figure 10 Fingerprint detection success rate under different parameters

(1) 本文方法与 IBW 方法实验结果对比
为了评估指纹冗余位 r 的值对指纹编码的影响，在不同的 r 条件下对本文方法的指纹识别率与 IBW 方法进行了对比，结果如图 11 所示。从实验结果中可以看出，指纹的检测成功率随着冗余位的增加而提高，并且在相同冗余位的情况下，本文方法运行效果优于 IBW 方法。

响，在不同的 r 条件下对本文方法的指纹识别率与 IBW 方法进行了对比，结果如图 11 所示。从实验结果中可以看出，指纹的检测成功率随着冗余位的增加而提高，并且在相同冗余位的情况下，本文方法运行效果优于 IBW 方法。

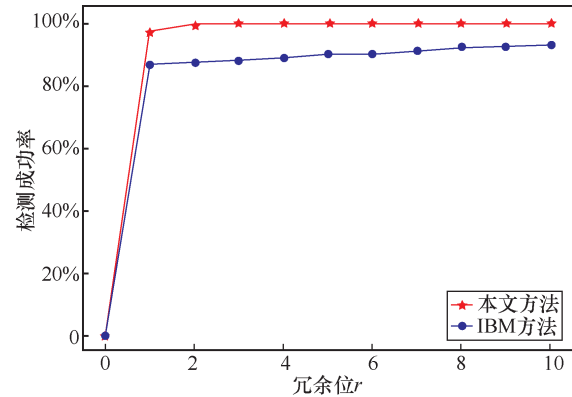


图 11 不同冗余位下的指纹检测成功率

Figure 11 Fingerprint detection success rate with different redundancy bits

由上述实验可以看出在冗余位为 3 时，本文方法检测成功率达到 100%，出于对效率和检测成功率的考虑，后续实验都选择冗余位为 3。

编码一个 m 位长度的指纹或水印，IBW 方法需要消耗 $4m$ 个时隙，而本文方法只需要消耗 $3m$ 个时隙，因此，编码同样长度的比特信息，本文提出的编码方法需要消耗的时隙和数据包更少，在冗余位 $r=1$ 时，数据速率为 4 packet/s 的情况下，嵌入相同长度的指纹，本文方法相比 IBW 方法可以缩减 1/4 的数据包使用量，表现则更高效（如图 12 所示）。

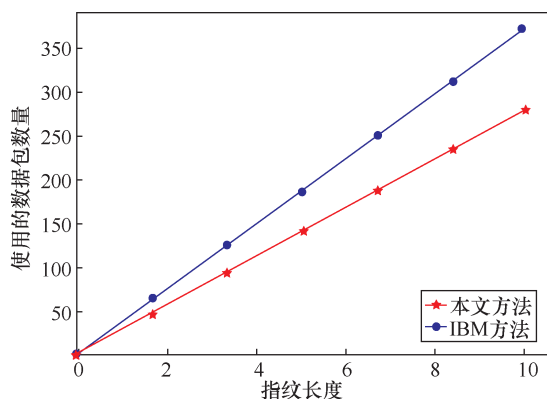


图 12 不同指纹长度所使用数据包数量

Figure 12 Number of packets used for different fingerprint lengths

(2) 本文方法抗干扰实验

本文对数据流经过不同的路由器数量 R 、不同背景流数量 bf 在不同的网络干扰情况下进行实验并根据结果进行对比分析。如前所述, 实验选择时间偏移 o 为 900 ms, 时间间隔为 400 ms, 冗余位 r 为 3。经过不同路由器时, 不同背景流下指纹检测率如图 13 所示。

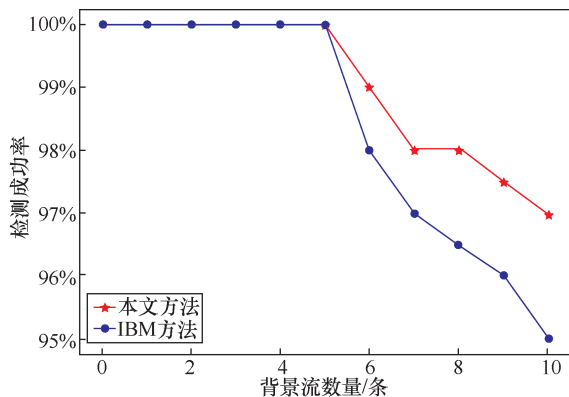


图 13 不同背景流下指纹检测率

Figure 13 Fingerprint detection success rate for different background streams

通过图 13 的结果可以看出, 经过不同数量的路由器时, 对指纹提取的成功率会产生一定的影响, 这是因为数据流经过不同路由器时会对数据流的时隙产生扰动。检测成功率也会随着背景流的数量增加有所下降, 这是因为嵌入指纹的流和背景流并行注入内核, 干扰了指纹嵌入和提取的同步。

不同时间延迟下指纹检测成功率如图 14 所示。可以看出, 随着时间延迟的增大, 检测成功率会有所下降, 这是因为指纹的嵌入和提取都是基于时隙来进行的, 时间的延迟会扰乱各个时隙

内数据包的分布。数据流经过路由器的个数和背景流与上文实验一样, 对指纹的检测成功率会有影响, 但指纹检测成功率都在 90% 之上。

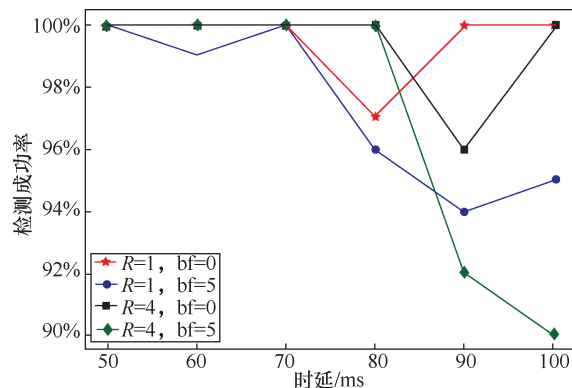


图 14 不同时间延迟下指纹检测成功率

Figure 14 Fingerprint detection success rate with different time delays

不同丢包率下指纹检测成功率如图 15 所示。可以看出, 随着丢包率的增加, 检测成功率会有所下降, 这是因为流指纹提取时会根据时隙内的数据包数量来判定提取的比特是 1 还是 0, 如果丢包, 时隙内数据包数量会发生变化, 会影响提取端的判断。此外, 有背景流时指纹提取检测成功率优于无背景流, 这是因为在丢包率固定时, 在有背景流的实验丢包时, 会随机丢弃背景流和嵌入流的包, 而在无背景流实验时, 丢弃的全是嵌入指纹的流, 因此对检测成功率影响稍大。在有背景流且丢包率很大的情况下, 指纹的检测成功率仍然有一定的保障。

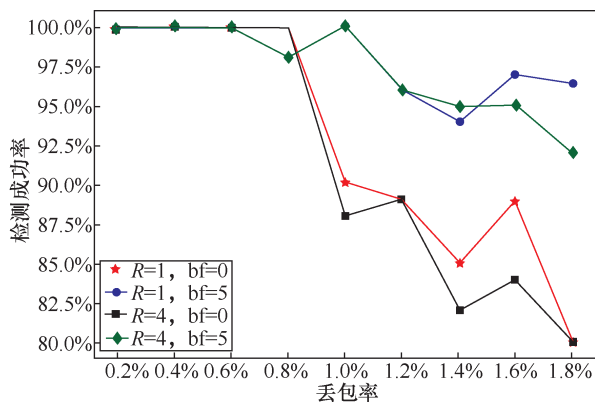


图 15 不同丢包率下的指纹检测成功率

Figure 15 Fingerprint detection success rate under different packet loss rates

在不同抖动时间下指纹检测成功率, 如图 16 所示。可以看出, 随着抖动时间增大, 检测率会有所下降, 这是因为流指纹的嵌入依据数据包的

到达时间, 如果每个数据包到达时间都随机提前或延迟一定时间, 则会扰乱各个时隙中的数据包数量, 对指纹的提取产生影响。

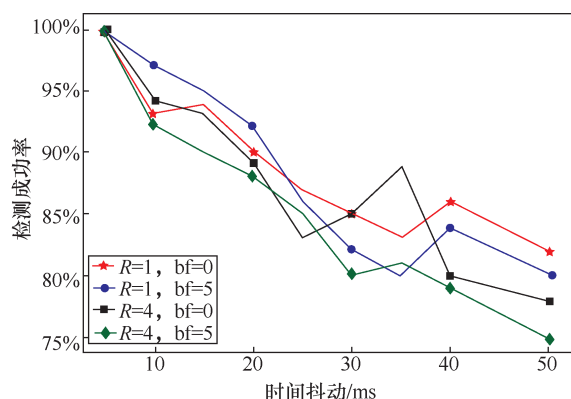


图 16 不同时间抖动下的指纹检测成功率

Figure 16 Fingerprint detection success rate under different time jitter

3.2 模型应用

流指纹可应用于检测中间人攻击 (MITM, man-in-the-middle attack) 的场景, 其主要目的是窃取或篡改信息而不被正常通信发现, 如图 17 所示, 给出中间人攻击典型模型, Alice 与 Bob 进行正常通信, 而攻击者 Jack 会对此正常通信发起攻击来窃取或篡改二者的通信内容, 于是, 实际情况变成了 Alice 与 Jack 通信、Bob 与 Jack 通信。

除此之外, 还有一些常见的中间人攻击, 如 DNS 欺骗、ARP 欺骗等。

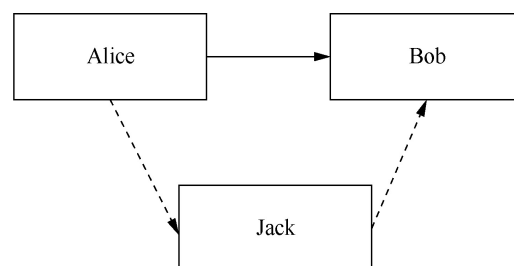


图 17 中间人攻击典型模型

Figure 17 Typical model of man-in-the-middle attack

发生这类欺骗的原因是用户缺少对服务提供者的身份验证, 因此本文设计基于流指纹的网络身份检测方法。如图 18 所示, Web 服务提供者设为可信域 1, 可信域 1 进行指纹的嵌入, 可信域 2 包括用户, 其安装了指纹提取器。当用户正常请求服务器时, 服务器在正常响应用户请求时的返回流中嵌入了指纹, 待用户收到返回流时, 将流中的指纹提取出来, 与原始指纹做对比。若指纹一致, 则认为服务提供者可信, 若用户在返回流提取不到指纹或提取的指纹与原指纹不一致, 则认为发生了中间人攻击, 并采取相应措施, 如图 19 所示。

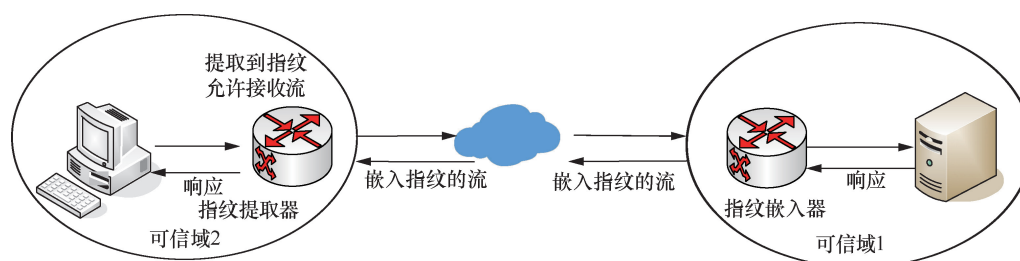


图 18 可信服务流交互示意

Figure 18 Illustration of trusted service flow interaction

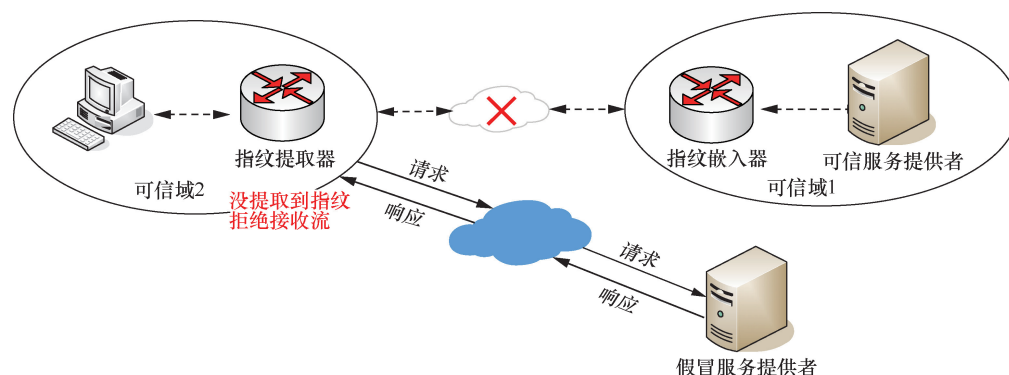


图 19 假冒服务流交互示意

Figure 19 Counterfeit service flow interaction schematic

由上述应用场景可以看出,借助流指纹模型,可有效检测中间人攻击,防御网络身份欺骗。除此之外,流指纹还可应用于攻击溯源、流量监控、流量关联和跳板检测等,相比被动的流量检测,流指纹在时空消耗、软硬件的使用上更具优势。

4 结束语

为了支持内生安全的数据可信,本文在基于流水印/指纹机制研究的基础上,设计了一种基于时隙的多重冗余流指纹模型,并对该模型进行了在网络典型干扰(如丢包、延迟等)环境下的评估,设计实现了一个流指纹原型系统,在系统的内核层进行了指纹的编解码、嵌入和提取,并对本文的流指纹模型系统的应用场景进行了阐述。后续还有一些工作可以继续展开,具体如下。

1) 指纹模型的理论分析只是建模分析,没有结合实际网络流量进行分析,实际网络中流量种类繁多,不同应用的网络流会呈现不同特点,因而有待进一步的细化建模分析。

2) 在流指纹原型系统设计与实现上有提升空间,如当前原型系统没有考虑指纹相关参数的传递机制。

3) 为了提高指纹嵌入和提取的精度和效率,本文系统是基于内核开发的,虽然相比传统方法有了很大提升,但仍有改进空间,下一步工作可考虑利用 DPDK、XDP、SmartNIC、DPU、P4 等技术,实现性能提升。

参考文献:

- [1] 郭江兴. 网络空间拟态防御研究[J]. 信息安全学报, 2016, 1(4): 1-10.
- [2] WU J X. Research on cyber mimic defense[J]. Journal of Cyber Security, 2016, 1(4): 1-10.
- [3] WANG X, REEVES D S, WU S F, et al. Sleepy watermark tracing: An active network-based intrusion response framework[C]//IFIP International Information Security Conference. 2001: 369-384.
- [4] YU W, FU X, GRAHAM S, et al. DSSS-based flow marking technique for invisible traceback[C]//2007 IEEE Symposium on Security and Privacy (SP'07). 2007: 18-32.
- [5] JIA W, TSO F P, LING Z, et al. Blind detection of spread spectrum flow watermarks[J]. Security and Communication Networks, 2013, 6(3): 257-274.
- [6] HUANG J, PAN X, FU X, et al. Long PN code based DSSS watermarking[C]//Proceedings IEEE INFOCOM. 2011: 2426-2434.
- [7] ZHANG L, WANG Z, WANG Q, et al. MSAC and multi-flow attacks resistant spread spectrum watermarks for network flows[C]//2nd IEEE International Conference on Information and Financial Engineering. 2010: 438-441.
- [8] XU X, ZHANG L, YAN J. PN code diversification based spread spectrum flow watermarking technology[C]//14th International Conference on Computational Intelligence and Security (CIS). 2018: 254-258.
- [9] WANG X, REEVES D S, WU S F. Inter-packet delay based correlation for tracing encrypted connections through stepping stones[C]//European Symposium on Research in Computer Security. 2002: 244-263.
- [10] WANG X, REEVES D S. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays[C]//Proceedings of the 10th ACM Conference on Computer and Communications Security. 2003: 20-29.
- [11] HOUMANSADR A, KIYAVASH N, BORISOV N. Rainbow: A robust and invisible non-blind watermark for network flows[C]//NDSS. 2009: 406-422.
- [12] HOUMANSADR A, BORISOV N. The need for flow fingerprints to link correlated network flows[C]//International Symposium on Privacy Enhancing Technologies Symposium. 2013: 205-224.
- [13] SOLTANI R, GOECKEL D, TOWSLEY D, et al. Fundamental limits of invisible flow fingerprinting[J]. IEEE Transactions on Information Forensics and Security, 2019, 15: 345-360.
- [14] REZAEI F, HOUMANSADR A. FINN: fingerprinting network flows using neural networks[C]//Annual Computer Security Applications Conference. 2021: 1011-1024.
- [15] IACOVAZZI A, SARDA S, ELOVICI Y. Inflow: Inverse network flow watermarking for detecting hidden servers[C]//IEEE INFOCOM 2018-IEEE Conference on Computer Communications. 2018: 747-755.
- [16] IACOVAZZI A, SARDA S, FRASSINELLI D, et al. DropWat: An invisible network flow watermark for data exfiltration traceback[J]. IEEE Transactions on Information Forensics and Security, 2017, 13(5): 1139-1154.
- [17] YAO Z, ZHANG L, GE J, et al. An invisible flow watermarking for traffic tracking: A hidden markov model approach[C]//IEEE International Conference on Communications. 2019: 1-6.
- [18] PYUN Y J, PARK Y H, WANG X, et al. Tracing traffic through intermediate hosts that repacketize flows[C]//IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications. 2007: 634-642.
- [19] HOUMANSADR A, BORISOV N. SWIRL: A scalable watermark to detect correlated network flows[C]//NDSS. 2011.
- [20] MO L, LYU G, WANG B, et al. The design and implementation of an efficient quaternary network flow watermark technology[C]//17th International Conference on Mobility, Sensing and Networking (MSN). 2021: 746-751.
- [21] REZAEI F, HOUMANSADR A. TagIt: Tagging network flows using blind fingerprints[J]. Proc Pri Enhancing Technol, 2017, 2017(4): 290-307.
- [22] 罗跃斌. 网络主动防御关键技术研究[D].长沙: 国防科学技术大学, 2017.
- [23] LUO Y B. Research on key technologies of network active de-

fense[D]. Changsha: National University of Defense Technology, 2017.

- [22] LIU W, LIU G, XIA Y, et al. Using insider swapping of time intervals to perform highly invisible network flow watermarking[J]. Security and Communication Networks, 2018, 2018.
- [23] WANG X, CHEN S, JAJODIA S. Network flow watermarking attack on low-latency anonymous communication systems[C]//2007 IEEE Symposium on Security and Privacy (SP'07). 2007: 116-130.
- [24] ZHANG L, LUO J, YANG M. An improved DSSS-based flow marking technique for anonymous communication traceback[C]//Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing. 2009: 563-567.
- [25] HOU J, LI Q, TAN R, et al. An intrusion tracking watermarking scheme[J]. IEEE Access, 2019, 7: 141438-141455.
- [26] WANG X, LUO J, YANG M. An interval centroid based spread spectrum watermark for tracing multiple network flows[C]//IEEE International Conference on Systems, Man and Cybernetics. 2009: 4000-4006.
- [27] WANG X, LUO J, YANG M. A double interval centroid-based watermark for network flow traceback[C]//The 14th International Conference on Computer Supported Cooperative Work in Design. 2010: 146-151.
- [28] LIU Y, GHOSAL D, ARMKNECHT F, et al. Robust and undetectable steganographic timing channels for IID traffic[C]//International Workshop on Information Hiding. 2010: 193-207.

[作者简介]



刘科显（1996-），男，河南周口人，北京邮电大学博士生，主要研究方向为网络安全。



关建峰（1982-），男，河南巩义人，北京邮电大学副教授、博士生导师，主要研究方向为网络架构、区块链与网络安全、移动互联网、大数据与人工智能。



张婉澈（1995-），女，安徽芜湖人，北京邮电大学硕士生，主要研究方向为内生网络安全。



何志凯（1986-），男，安徽蚌埠人，中国航空综合技术研究所高级工程师，主要研究方向为无人系统标准化与可用性。



闫迪嘉（2000-），男，北京人，主要研究方向为网络安全。