第 49 卷 第 2 期
2017 年 3 月

工 程 科 学 与 技 术
ADVANCED ENGINEERING SCIENCES

Vol. 49 No. 2
Mar. 2017

·CTCIS 2016 推荐论文·

DOI: 10.15961/j.jsuese.201601020

基于云流量混淆的 Tor 匿名通信识别方法

何永忠[1,2],李　响[1,2],陈美玲[1,2],王　伟[1,2]

(1. 北京交通大学 智能交通数据安全与隐私保护技术北京市重点实验室,北京 100044; 2. 北京交通大学 计算机与信息技术学院,北京 100044)

摘　要: 针对采用云流量混淆 Meek 机制的 Tor 匿名通信流量与其他普通云流量难于区分识别的问题,提出了基于流静态特征的 Tor 匿名通信识别方法和基于支持向量机 SVM 分类算法的 Tor 匿名通信识别方法。本文首先从连接特征分析、数据包静态特征分析以及数据流动态特征分析出发,通过对大量 Tor-Meek 通信流量以及非 Tor-Meek 通信流量的对比实验研究,确定了 7 个具有特异性和较强区分度的 Tor-Meek 通信流量的静态和动态流量征,然后在此基础之上提出了基于特征匹配算法的 Tor-Meek 匿名通信识别方法,该方法能够快速识别 Tor-Meek 通信流量,对于包含大于 200 个包的流识别准确率大于 90%。为了进一步适应 Tor 的版本变化带来的特征改变,基于 Meek 流分片机制的数据流统计特征分析,分别从长度及个数、长度方差、长度熵、接收发送序列等 4 个方面,提出了识别 Tor-Meek 流的 16 种 Tor-Meek 流量统计特征,采用 SVM 分类算法对 Tor-Meek 流量进行识别,通过系统的实验研究不同特征组合、不同算法参数选择的算法识别准确率和召回率,筛选出最优的特征组合和参数。在实验室环境中搭建实验数据采集平台并采集 Tor-Meek 通信和其他通信数据进行实验,该算法对长度大于 40 个包 Tor-Meek 流的识别准确率大于 97%,召回率大于 99%,且具有较高的识别效率。实验结果表明,采用特征匹配可以实现对云流量混淆 Tor 匿名通信的快速识别,而基于流分片统计特征的分类算法对不同 Tor 通信软件版本的变化具有更高的稳定性和识别准确率。

关键词: 匿名通信; Tor; 流量混淆; 流量识别

中图分类号: TP393.4　　**文献标志码:** A　　**文章编号:** 2096-3246(2017)02-0121-12

Identification of Tor Anonymous Communication with Cloud Traffic Obfuscation

HE Yongzhong[1,2], *LI Xiang*[1,2], *CHEN Meiling*[1,2], *WANG Wei*[1,2]

(1. Beijing Key Lab. of Security and Privacy in Intelligent Transportation, Beijing Jiaotong Univ., Beijing 100044, China;
2. School of Computer and Info. Technol., Beijing Jiaotong Univ., Beijing 100044, China)

Abstract: In order to solve the problem of identifying the meek-based Tor anonymous traffic from the TLS-based cloud computing service traffic, an identification method for Tor's anonymous communication based on traffic feature matching and a classification method of Tor's anonymous traffic based on SVM were proposed. Firstly, based on the analysis of connection, static packet and dynamic traffic of the captured Tor-Meek and non Tor-meek traffic in the lab environment, seven specific static and dynamic features of Tor-Meek traffic were identified. Lately, a traffic feature matching identification method for Tor's anonymous communication technique was proposed, which could be used to quickly detect Tor-Meek traffic and the accuracy rate is over 90% for longer traffic with packets number exceeding 200. In order to be robust to the upgrading and transformation of Tor versions, statistic features of the slicing of Tor-Meek traffic were analyzed including the length and count, length variation, length entropy, sequence of sending and receiving of the sliced traffic. Then 16 statistic features were identified, based on which an identification and classification method for Tor's anonymous traffic by using SVM machine learning algorithm was proposed. Different feature combinations and algorithm parameters were studied experimentally to decide which ones can yield the best accuracy and recall rate of the classification algorithm. It was shown that when the number of packets in one session wa above 40, and the length of each slice of one session was 40 packets, the identification accuracy was above 97 and the recall rate was over 99% for the SVM based method. The experiments results show that while the feature matching methods is

收稿日期: 2016-09-15

基金项目: 国家自然科学基金资助项目(61402035)

作者简介: 何永忠(1969—),男,副教授,博士. 研究方向: 系统安全; 网络安全. E-mail: yzhhe@bjtu.edu.cn

effective for quick identification, the machine learning method is more accurate and robust to the changing and upgrading of different versions of Tor browser in identifying anonymous traffic of specific versions of Tor-Meek.

Key words: anonymous communication; Tor; traffic obfuscation; traffic identification

匿名通信技术通过隐藏通信过程中通信双方的身份信息,或者隐藏通信双方的联系,从而有效地实现了对通信实体身份信息的保护[1]。匿名通信技术最早出现在 1981 年,Chaum 提出了基于节点混淆(Mix)的匿名通信技术[2]。Mix 节点接收多个发送者发来的消息,并对这些消息进行混淆处理,之后将消息发送给接收者,从而使得消息无法被跟踪,实现了匿名通信。目前,匿名通信技术根据实现技术主要可以分为两类:基于重路由的匿名通信技术和基于广播的匿名通信技术。基于重路由的匿名通信技术的典型实现有:Anonymizer[3]、Onion Routing[4]、Mixminion[5]、Crowds[6]等。基于广播的匿名通信技术的典型实现有 DC-Net[7]、P5[8]等。

第二代洋葱路由[9](The second generation Onion Router,Tor)匿名通信系统诞生于 2003 年,由美国海军研究实验室(US Naval Research Laboratory)赞助建立。Tor 匿名通信系统是目前应用最为广泛的匿名通信系统。截止到 2016 年 11 月,全球一共有大约 7 150 多个 Tor 中继节点和 2 150 个 Tor 网桥节点,平均每秒产生的流量约为 10.1 GB(Tor 网桥节点最高曾经达到 4 000 多个)[10]。全球每天有大约 20 万用户通过直接连接的方式使用 Tor,另外有大约 4 万用户通过网桥连接的方式使用 Tor。Tor 采用重路由技术和层层加密技术,有效地抵御了流量分析等各种攻击,为用户提供了良好的隐私保护。然而匿名通信技术在为合法用户提供身份信息保护的同时,也可能被滥用于网上的非法或者犯罪行为,给网络安全带来巨大的威胁。Tor 为进一步伪装和隐匿其通信流量,引入了多种集成了传输插件的网桥技术,包括 obfs3、obfs4、Meek、Flash proxy、FTE 以及 scramblesuit 等。通过这些技术可以对 Tor 流量进行流量混淆,以规避对 Tor 的流量审查和监管。2015 年 11 月,ISIS 恐怖组织公布了一份网络安全手册,在这份手册中 Tor 作为被推荐的浏览器,ISIS 称 Tor 可以“隐藏用户的身份和防止跟踪”。因此,如何识别 Tor 匿名通信流量并对其实施有效的监管,对保护网络安全打击违法犯罪具有重要意义。

目前,主要的流量识别技术可以分为:基于端口号、基于深层包检测(deep packet inspection,DPI)、基于行为特征和基于流统计特征的流量识别技术[8]。

基于端口号的流量识别方法,所依据的是互联网地址指派机构 IANA(Internet Assigned Numbers Authority)制定的端口号映射表。基于深层包检测的流量识别方法,又称为基于载荷特征的流量识别技术方法。该方法的核心是利用模式匹配算法搜索流量有效载荷(payload)中的特征值,从而来对各种不同的应用层协议进行识别。基于行为特征的流量识别方法,是依据传输层的主机行为模式对网络中不同应用的流量进行分类的方法。通过观察流量在传输层中如何连接、如何进行数据交互来判断其属于哪种应用。基于统计特征的流量识别方法,通过建立数据流特征的模型,并分析数据流的统计特征,比如流中数据包的平均大小,数据包到达时间以及数据包的时间间隔等,之后依据这些特征对不同的协议进行区分。对于数据流统计特征的建模、分析,和对数据流的分类,可以采用机器学习的方法,利用数据流的统计特征建立机器学习分类模型,然后选择恰当的机器学习算法来对流量进行分类。

然而与明文流量的识别技术不同,对于加密流量来说,检测者无法提取加密流量内部有效荷载的特征,因此给流量识别带来了更大的难度。为了识别加密流量,必须对使用了同一种加密协议的不同应用的流量进行区分。对使用了同一种加密协议的不同应用的流量进行区分。例如,Tor 使用了 TLS 加密技术对其流量进行加密,因此要对 Tor 流量进行识别,就要对 Tor 和非 Tor 的 TLS 流量进行分类。

为了提高识别率,针对不同的加密应用应设计不同的识别算法。由于针对 Tor 匿名通信流量识别采用了 TLS 加密以及规避审查技术,因此采用基于端口号或者深层包检测等技术不能对其流量进行有效的检测。目前,针对 Tor 匿名通信流量识别的研究报道较少,例如,何高峰等[11]提出了基于 TLS 指纹特征和基于报文长度分布特征的 Tor 匿名通信流量识别方法。通过利用 Tor 信元的结构和长度等特征,此两种识别方法均能对 Tor 匿名通信的流量进行有效识别。然而,在 Tor Browser 4.0 版本引入了 Meek 传输插件以后,Meek 作为 Tor 的一种特殊的网桥模式,成功地把 Tor 流量伪装成了基于 HTTPS 加密的云服务流量(云流量混淆技术)。一方面,Meek 使用了前置域名技术,通过第三方服务器进行流量转发,从而隐藏了正在和 Tor 网桥通信的事实,

使得传输内容看起来是在访问另一个站点。另一方面，Meek 使用了基于浏览器代理的加密技术，通过浏览器建立 HTTPS 隧道传输 Tor 流量，也就是将 Tor 的数据流封装成 HTTPS 的请求和响应序列，从而隐藏了 Tor 流量原有的特征。因此，上述方法无法对 Tor-Meek 流量进行有效的识别。

目前，Tor-Meek 匿名通信流量识别面临如下待解决的问题: 1) 由于 Meek 通过 Amazon、Azure 和 Google 云服务平台对 Tor 流量进行流量转发，所以其目的 IP 地址和端口号与其他应用访问这些云服务平台的 IP 地址和端口号相同，因此无法通过 IP 地址或端口号来区分 Tor-Meek 流量和普通的访问云服务平台的流量; 2) Meek 使用基于浏览器代理的加密技术，通过浏览器建立 HTTPS 隧道传输 Tor 流量，从而隐藏了 Tor 的 TLS 指纹特征和报文长度等特征，因此无法利用 Tor 流量原有的特征对 Tor-Meek 流量进行有效识别; 3) Tor-Meek 流量是通过 TLS 协议进行加密的，因此无法通过明文协议特征匹配来识别流量。

本文在搭建的实验环境中捕获 Tor-Meek 匿名通信流量并对其进行特征分析，包括连接特征分析、数据包静态特征分析、数据流动态特征分析和数据流统计特征分析等，确定了 Tor-Meek 的流量 7 种静态和动态特征，以及 4 类 16 种统计特征。提出了基于特征匹配的 Tor 匿名通信识别方法和基于 SVM 的 Tor 匿名通信识别方法。实验结果表明，本文提出的方法均可有效识别基于 Meek 的 Tor 匿名通信流量，且具有较高的准确率、召回率和性能。

1　Tor 与 Meek 的基本原理

1.1　Tor 基本原理

Tor 是一个分布式的匿名通信系统，由用户、洋葱代理、洋葱路由器、目录服务器及目标应用服务器等部分组成[12-13]。Tor 匿名通信系统架构如图 1 所示。

1) 用户(Alice) : 通过 Tor 网络与外界进行匿名通信的计算机。用户 Alice 通过洋葱代理来连接到 Tor 网络。

2) 洋葱代理(onion proxy，简称 OP) : 洋葱代理是用户 Alice 在本地系统中运行的代理服务程序。OP 的主要功能有两个: 一个是负责与本地的上层应用程序进行数据交换; 另一个是负责执行匿名通信协议，建立匿名通信链路，接收上层应用的数据，将其封装成 Tor 信元后发送至匿名通信链路。

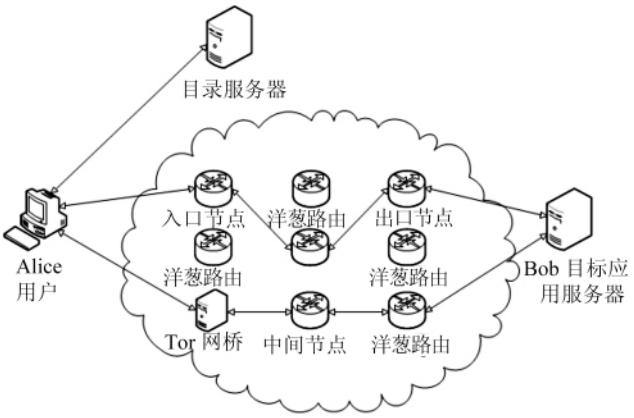

图 1　Tor 的网络结构

Fig. 1　Network structure of Tor

3) 洋葱路由器(onion router，简称 OR) : 洋葱路由器负责建立匿名通信链路和转发数据。所有的 OR 节点的信息都保存在目录服务器中。

4) 目录服务器(directory server) : 目录服务器负责汇总 Tor 网络运行状态，收集、存储和管理所有的 OR 节点信息。

5) 网桥节点(Bridge relays) : Bridge 节点是一种特殊的 OR 节点，它不被包含在 Tor 目录服务器所公开发布的 OR 节点列表中。当 Tor 目录服务器或普通的 OR 节点不能被直接访问时，用户可以直接与 Bridge 节点进行通信。

6) 目标应用服务器(Bob) : 目标应用服务器 Bob 是通信的目的端，他提供具体的应用服务。

当用户 Alice 通过 Tor 网络对目标应用服务器 Bob 进行访问时，首先 Alice 的主机上需要安装并运行洋葱代理 OP，之后 OP 连接 Tor 目录服务器，并从目录服务器上获得 Tor 节点列表。OP 根据自身的访问策略和 Tor 节点状态，选择 3 个节点分别作为入口节点，中间节点和出口节点，并在这 3 个节点之间建立层层加密的匿名通信链路，最终达到目标应用服务器 Bob。当 Alice 需要访问其他站点时，OP 会重新为 Alice 建立另一条匿名链路。

为了提高 Tor 匿名通信系统的可用性及安全性，Tor 引入了一种网桥(Bridge) 机制[14]，通过网桥机制 Tor 可以有效地抵御网络审查和流分析攻击。网桥机制的核心是网桥节点，它是一种特殊的 OR 节点，不在 Tor 目录服务器公开发布的 Tor 节点目录中，每次用户可以申请获取少量网桥节点，因此比 Tor 节点更难完全阻断。当 Tor 目录服务器或者公开的 Tor 节点不能被直接访问时，Tor 客户端可以通过网桥节点获得其它 OR 节点的信息，并以网桥节

点作为连接 Tor 网络的入口节点进行通信，从而规避了审查。此外 Tor 网桥还引入了多种流量混淆插件，通过这些流量混淆插件，Tor 流量得到了进一步的伪装。

1.2 Meek 基本原理

Tor Browser4.0 版本中增加了 Meek[15-16] 传输插件(pluggable transport)，它把 Tor 流量伪装成了基于 HTTPS 加密的云服务流量，从而有效地抵御网络审查。目前，Tor 支持 Meek-Amazon、Meek-Azure 和 Meek-Google 这 3 个选项，可以分别通过亚马逊云服务、微软云服务和谷歌云服务进行流量转发。

Meek 由两部分组成：前置域名技术(domain fronting)和基于 HTTP 的隧道代理(HTTP-based tunneling proxy)。前置域名技术用于连接至代理服务器；基于 HTTP 的隧道代理用于将 HTTP 请求序列转化为 Tor 数据流。Meek 客户端利用一个可以访问的前置域名(allowed. example)来保护 Tor 网桥的域名(forbidden. example)不被网络审查发现。前置域服务器在接收到请求后，解密该请求的 TLS 层，之后根据内部的 HTTP 请求的 Host header 将请求发送给 Tor 网桥。在完成 Tor 网络的访问之后，Tor 网桥以 HTTP 响应的方式将数据回传给客户端。Meek 客户端和 Meek 服务器是 Tor 和传输插件之间接口，对于 Tor 来说 Meek 客户端和 Meek 服务器之间是一种不透明的数据传输。Meek 工作原理如图 2 所示。

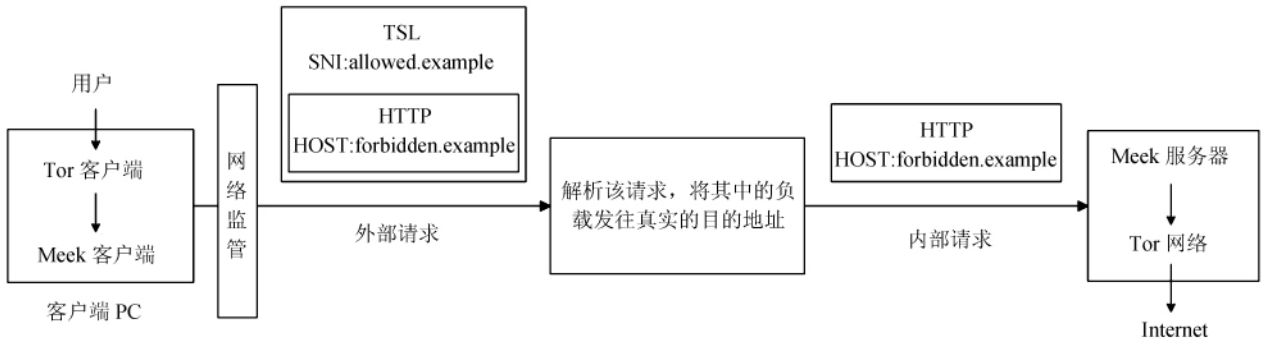

图 2 Meek 工作原理

Fig. 2 Meek's schematic diagram

Meek 客户端担任 Tor 客户端的上层代理。当 Meek 客户端接收到来自 Tor 客户端的一组数据时，它把数据封装到 POST 请求中，并通过前置域名服务将请求转发给 Tor 网桥。在 Tor 网桥上运行着一个服务器程序，即 Meek 服务器，它负责解析该 HTTP 请求，并把其中的负载内容发送到 Tor 网络。

在完成 Tor 网络的访问之后，数据经由 Tor 网桥返回到客户端。Meek 服务器在接收到 Tor 网桥准备回传给客户端的数据时，先把数据封装到 HTTP 响应的结构体内，再发送给 Meek 客户端。Meek 客户端在接收到该 HTTP 响应之后，把其中的内容传送给 Tor 客户端。这些请求和响应是严格序列化的。Meek 客户端直到收到前一个请求的响应之后，才会发送第 2 个请求。

Meek 服务器需要同时处理多个客户端访问，他为每个活动的客户端保持一条连接，并通过 session ID 标识不同客户的请求。Meek 客户端需要在 HTTP 请求的头部中加入一个 session ID。当 Meek 服务器收到用户请求之后，如果发现该请求的 session ID 是新的，那么他将为该请求打开一条新的连接，并启动 Tor 进程连接至 Tor 网络。

由于 HTTP 是一个基于请求的协议，它不会主动推送数据到客户端。为了使服务器能够回发数据到客户端，即便在没有数据传输的时候，Meek 客户端仍需要发送空的轮询请求(polling request)到服务器。这个轮询请求给了服务器回传数据的机会。轮询间隔从 100 ms 开始，并以指数递增，最大值为 5 s。

1.2.1 基于浏览器代理的加密

如果不进行进一步的伪装，Meek 并不会破坏 Tor 原有的 TLS 指纹特征。Tor 在其 TLS 握手阶段的表现出了其特有的指纹特征，这些特征在文献[7]中已经提到，并且成功地利用这些特征对 Tor 匿名通信流量进行了识别。

为了隐藏 Tor 的 TLS 指纹特征，Meek 客户端将其所有 HTTPS 请求代理给一个真实的浏览器，这样做使得 Meek 的请求看起来与一般的浏览器请求一样。Tor Browser 使用 Firefox 对它的 HTTPS 请求进行代理。Meek 代理仅运行在后台，而且不提供给用户接口，也不给用户浏览器传递认证状态信息。Tor

客户端程序同时启动 Meek 客户端和一个浏览器程序，之后配置 Meek 客户端把它的请求代理给浏览器程序。只有这个浏览器，才是真正与互联网进行连接部分。

1.2.2　前置名域技术

前置域名技术主要是为了将被禁止访问的域名转化成可被访问的域名。它将可被访问的域名放置在请求的外部，主要用于 DNS 查询；并将被禁止访问的域名放置在请求的内部，也就是 HTTP 请求的 HOST Header 中。前置域名技术利用了 TLS 的 SNI 扩展，客户端可以在 TLS 握手过程开始的时候指示尝试连接的主机名，这使得服务器可以针对同样的 IP 地址和 TCP 端口号呈现多个证书，即同一个 IP 地址能够访问不同的 HTTPS 协议的网站或者是其他建立在 TLS 上的服务，而且不要求所有的站点都使用相同的证书。例如，想要访问 http://appspot.com 下某个被禁止的子域名，那么采用这用策略后可以伪装成访问 http://www.google.com。

2　基于特征匹配的识别方法

在文献[15]中提到了 Tor-Meek 流量具有区别于其他流量的工作特性，其中包括 Tor-Meek 的 TLS 密码套件和扩展字段的特殊性、由于轮询请求而产生的心跳现象、以及报文的最大负载长度等。结合实验观察，确定用于 Tor-Meek 流量识别的特征集合。

在实验网络环境中，通过 Wireshark 工具捕获 Meek 流量。在数据采集时，先排除其他干扰流量，关闭除系统运行必须的进程之外的所有进程，保证不会产生干扰的 TLS 流量。排除干扰后启动 Wireshark 开始捕获流量。之后启动 Tor Browser，选择 Meek 连接方式进行连接，此时 Wireshark 便可以捕获到 Meek 的 TLS 流量。

基于 Tor 和 Meek 的工作原理，对所捕获的 Tor 流量进行特征分析。数据来源为 Tor Browser 4.0.2 ~6.0.5 版本。主要特征分析可以分为以下四各方面：1）连接特征分析：包括 Meek 的 TLS 连接个数、持续时间等；2）数据包静态特征分析：包括 TLS 握手报文的指纹特征（如 Cipher Suits、Extensions）等；3）数据流统计特征分析：包括 TLS 流的包长分布、延迟规律等；4）数据流动态特征分析：包括不同网络状态下 TLS 流接收、发送数据包序列的特征等。

根据以上分析确定 Tor-Meek 流量的 7 个特征，然后提出基于特征匹配的识别算法。

2.1　Tor-Meek 流量的静态、动态特征

2.1.1　单一链接特征

在通过 Meek 网桥连接 Tor 网络时，Meek 客户端会建立唯一的一条 TLS 连接与相应的云服务平台通信。另外，一台 PC 在同一时间内只允许启动一个 Tor Browser 进程，而 Meek 客户端是由 Tor Browser 在后台调用的，所以一台 PC 在同一时间内只能启动一个 Meek 客户端。综合来看，一台 PC 在同一时间内至多只有一条 Meek 的 TLS 连接。

当 Meek 的 TLS 连接断开而会话并没有结束时，Meek 客户端会为此会话重新建立 TLS 连接，或者说是恢复 TLS 连接。这是一种很常见的情况，比如由于网络环境不稳定，出现丢包或者拥塞等情况时，就会出现 TLS 连接断开并恢复情况。Meek 客户端在恢复 TLS 连接时会使用之前的 TLS 连接的 session ID，并通过 TLS 恢复会话的握手方式恢复该 TLS 连接。

2.1.2　有序连接特征

Meek 客户端接收到上层的 Tor 客户端发送来的一组数据时，它将数据封装到一个 HTTP 请求中，并通过前置域名技术将请求转发给 Meek 服务器。Meek 服务器解释该 HTTP 请求，之后把其中的负载内容发送到 Tor 网络。在完成 Tor 网络的访问之后，Meek 服务器同样把数据封装到 HTTP 响应的结构体内，返回给 Meek 客户端。Meek 客户端在接收到该 HTTP 响应之后，把其中的内容传送给 Tor 客户端。HTTP 的请求和响应是严格序列化的。也就是说，直到 Meek 客户端接收到上一个请求的响应之后，才会发送第 2 个请求。

2.1.3　TLS Cipher Suits 特征

Meek 的 TLS Client Hello 中的 Cipher Suits 与 Tor Browser 所集成的浏览器的 TLS Client Hello 中的 Cipher Suits 相一致，包括 Cipher Suits 的个数、内容和顺序。例如，Tor Browser 4.0.2 集成的浏览器为 Firefox ESR 31.3.0，则 Tor Browser 4.0.2 的 Meek 网桥的 TLS Cipher Suits 与 Firefox ESR 31.3.0 的 TLS Cipher Suits 相一致。

2.1.4　TLS Extensions 特征

Meek 的 TLS Client Hello 中的 Extensions 与 Tor Browser 所集成的浏览器的 TLS Client Hello 中的 Extensions 相一致，包括 Extensions 的个数、内容和顺序。

2.1.5　TLS Server Name 特征

依据所选择的 Meek 网桥，Meek 的 TLS 连接

Client Hello 报文中 extension server name 可以确定。Meek-Amazon 网桥为 https://a0.awsstatic.com/,Meek-Azure 网桥为 https://ajax.aspnetcdn.com/。

2.1.6 轮询请求特征

Meek 为了保持客户端和服务器之间的连接,即便在没有数据发送的时候,Meek 客户端仍需要发送空的轮询请求数据包到服务器,这个轮询请求给了服务器回传数据的机会。本文称这种现象为 Meek 的轮询请求机制。

Meek 的轮询请求机制的工作原理如下:使用 Meek 网桥连接 Tor 网络时,Meek 客户端在没有数据传输的第 100 ms 后,发起第 1 次轮询请求,发送负载内容为空的 HTTP 请求到 Meek 服务器,并由 Meek 服务器传回一个对这个请求的响应。之后等待一段时间间隔,如果仍然没有数据传输,则继续发起第 2 次轮询请求。每次发起轮询请求所等待的时间比之前一次增加 50%,直到最大值 5 s。一旦有数据传输,轮询请求过程就立刻结束。当再次出现无数据传输时,轮询请求机制将重新开始。

Meek 用于轮询请求的数据包的大小是固定的,其中,Meek-Amazon 用于轮询请求机制的数据包大小为:发送包 209 byte,接收包 395 byte;Meek-Azure 用于轮询请求机制的数据包大小为:发送包 202 byte,接收包 383 byte。

2.1.7 分组传输特征

在使用普通浏览器通过 SSL 方式下载大文件时,会由客户端发出一个下载请求,之后服务器端开始向本地传输数据,直到文件下载结束,该过程中不会有客户端和服务器端的交互。但是使用 Tor Browser 下载相同文件时,会有客户端和服务器之间的交互。也就是说,在下载过程中,客户端会向服务器发送数据包,之后再由服务器端返回一组数据包。此外,在每次服务器的响应中,回传的一组数据包中均会有长度较小的数据包的存在。所以 Meek 表现出了明显的大块数据分组传输的特征。

2.2 基于特征匹配的 Tor 匿名通信识别方法

2.2.1 主要思想

基于流特征的 Tor 匿名通信识别方法包含基于 Meek TLS 静态特征的流量过滤和基于 Meek 流动态特征的流量识别两部分。

首先,Meek 流是一条特殊的 TLS 流,因此只要识别一条 TLS 流是否具有 Meek 流所独有的特征,即可确定该条流是否为 Meek 流,对于非 TLS 流可以直接排除。通过筛选 TLS 流对一般网络流量进行过滤,可以排除大部分的干扰。

Meek 在 TLS 握手阶段,表现出了 TLS Client Hello 中 Cipher Suits、Extensions 以及 Server Name 的指纹特征。这些指纹特征与 Tor Browser 所集成的浏览器的 TLS 的指纹特征相一致。目前最新版本的 Tor Browser 所集成的浏览器为 Firefox 38 ESR。根据最新 netMarketShare[18] 浏览器市场份额统计结果显示:Firefox 38 市场份额为 0.29%。而使用 Firefox 特定的版本访问 Amazon 云服务 https://a0.awsstatic.com/或者 Azure 云服务 https://ajax.aspnetcdn.com/将占据更小的比例。所以通过 Meek TLS 指纹特征对 TLS 流量进行过滤,之后所需要执行进一步检测的数据流将大幅减少。

Meek 的轮询请求机制使得 Meek 的 TLS 流呈现出了其独有的特征,包括其用于轮询请求报文的长度和轮询请求的时间规律,这些特征可以作为最终识别 Meek 流的依据。通过判断一条经过初步过滤的 TLS 流是否出现了 Meek 的轮询请求机制,即可判断该 TLS 流是否为 Meek 流。

2.2.2 基于静态特征的流量过滤

基于 TLS 静态特征的流量过滤是通过 Meek 流在 TLS 握手阶段 Client Hello 报文中的指纹特征对待识别的 TLS 流进行过滤。判断一条待识别的 TLS 流的 Client Hello 报文中 Cipher Suits 字段、Extensions 字段以及 Server Name 信息是否与 Meek 特征相匹配,从而实现对 Meek 流的过滤。基于 TLS 指纹特征的流量过滤方法步骤如下:

步骤 1:判断 ClientHello 报文中的版本信息是否为 SSLv3 或者 TLS。若成立则执行步骤 2,否则,判断为其他类型流量。

步骤 2:判断 ClientHello 报文中 Cipher Suits 个数是否满足 Meek 特征。若成立则执行步骤 3,否则判断为其他类型流量。

步骤 3:提取 ClientHello 报文中 Cipher Suits 内容,判断其内容和顺序是否符合 Meek 特征。若成立则执行步骤 4,否则,判断为其他类型流量。

步骤 4:判断 ClientHello 报文中 Extensions 个数是否满足 Meek 特征。若成立则执行步骤 5,否则,判断为其他类型流量。

步骤 5:提取 ClientHello 报文中 Extensions 内容,判断其内容和顺序是否符合 Meek 特征。若成立则执行步骤 6,否则,判断为其他类型流量。

步骤 6:提取 ClientHello 报文中 Server Name 内

容,判断其内容是否符合 Meek 特征。若成立则执行成功,否则,判断为其他类型流量。

基于 TLS 静态特征的流量过滤方法仅需分析 TLS 流中 Client Hello 报文,识别速度快,能适用于在线识别。但需要人工对 Meek 特征进行维护,以适应 Meek 指纹特征的变化。

2.2.3　基于 Meek 动态特征的流量识别

基于 Meek 流动态特征的流量识别是在通过基于 TLS 指纹特征的流量过滤方法对流量进行初步过滤的基础之上,根据 Meek 的轮询请求机制所表现出的特征,对待识别的 TLS 流做最后一步的识别判断,以达到识别 Meek 流的目的。基于 Meek 流动态特征的流量识别方法步骤如下:

步骤 1: 对待识别的 TLS 流提取特殊长度的数据包,并计算数据包的时间间隔。其中,Meek-Amazon 和 Meek-Azure 特征包长度见第 2.1.6 节所述。

步骤 2: 将时间间隔序列带入轮询请求机制判断器进行判断。若判断成功,则该 TLS 流为 Meek 流,否则不是。

2.2.4　轮询请求机制判断器

Meek 的轮询请求机制是当在没有数据传输的第 100 ms 时,发起第一次轮询请求,Meek 客户端发送一个负载为空的请求给 Meek 服务器,服务器响应一个负载为空的数据包。之后若仍然没有数据传输,则在第 $100\times(1+50\%)$ ms 时发起第 2 次轮询请求。以此类推,直到最大值 5 s,以后每次轮询请求的时间间隔为 5 s。按照这个规律计算,一共需要经过 11 次轮询请求后,轮询请求的时间间隔从 100 ms 增长到 5 s。

轮询请求机制判断器是判断一组待识别的时间间隔序列中,是否含有满足上述时间间隔规律的子序列。若有,则记录轮询请求机制的发起次数,并根据一个阈值来判断是否为 Meek 流; 否则返回失败。

实现方法是通过一个数组 pollingScore[8] 记录从第 1 到 11 次轮询请求所发生的次数。此外还需要约定一个允许范围,用于计算一个时间间隔是否满足轮询请求的时间间隔。由一组时间间隔序列可以计算得到一个记录了轮询请求发起次数的数组 pollingScore[],之后根据约定的阈值对该数组进行判断。该阈值由一个二元组确定,即(第 n 次轮询请求,发起次数 a) 其中 $n\in(1,11)$。例如第 2 次轮询请求,发起了 3 次。对于满足阈值的结果,即可认为是 Meek 流; 否则不是。

3　基于 SVM 的识别方法

Meek 流具有单一链接的特征,也就是说 Meek 客户端与服务器的通信是建立在单独的一条 TLS 连接之上的,且同一 PC 在同一时间内只有一条 Meek 的 TLS 连接。Meek 的 TLS 连接可能保持很长的时间,并且在一条 TLS 连接上可能出现多种网络状态。首先,Meek 建立 TLS 连接。之后 Meek 利用这条 TLS 连接与 Tor 网络建立连接,经过 3 跳握手后,与 Tor 网络连接成功。接下来,用户将利用这条 TLS 连接开始对 Tor 网络进行访问。用户可能在一段时间内不进行任何操作,这时没有任何网络访问,网络处于空闲状态; 用户也可能浏览网页,这时将产生间断的网络访问; 用户也可能下载一个较大的文件,这时将在一段时间内产生持续的网络访问,网络处于繁忙状态。由此,提出了 Meek 流分片模型,核心思想是将一条 Meek 的 TLS 流,按数据包接收和发送的时间先后顺序,分割成大小相同的片段,使每个分片内只包含一种网络状态。

基于统计特征和 SVM[19] 的 Tor 匿名通信识别方法,其核心思想是在对 Meek 流进行分片的基础之上,寻找到 Meek 流所特有的分片,即 Meek 特征分片。然后为 Meek 特征分片构建分类器,该分类器可以实现对 Meek 特征分片和非 Meek 特征分片的分类。在对一条未知的 TLS 流进行识别时,首先依据约定的分片规则对该流进行分片,然后利用 Meek 特征分片分类器对待识别的 TLS 流的分片进行分类。之后判断该待识别的 TLS 流中是否包含 Meek 特征分片,如果包含则该条 TLS 流为 Meek 流,否则不是 Meek 流。

在使用 Meek 网桥连接 Tor 网络时,Meek 客户端首先与 Meek 服务器建立连接,之后通过 Meek 服务器与 Tor 网络建立连接。在 Meek 客户端与服务器成功建立 TLS 连接后,Meek 客户端首先会向 Meek 服务器请求 Tor 网络状态和 Tor 节点目录信息,之后通过 Meek 服务器开始建立 Tor 链路的 3 跳握手过程,并进行后续的访问。由于 Meek 的请求和响应是严格序列化的。所以,在此过程中的每一步,都是有着严格先后顺序的。因此可以选用每个 Meek 流中的第 1 个分片,作为 Meek 特殊分片,用于区分 Meek 流和非 Meek 流。

通过 Meek 流量特征分析可以发现一条 Meek 流中数据包接收和发送的序列表现出了许多与一般 TLS 流不同的特征。一方面,在网络空闲时 Meek 会发起轮询请求机制,有固定字节的数据包按一定的

时间规律在客户端和服务器之间传送; 另一方面, Meek 在传输大量数据时,会呈现出明显的分组传输特征。此外,Meek 流是一条有序连接,Meek 的请求和响应是严格序列化的。

为表现出 Meek 流中数据包接收和发送序列的特征,用一组有序的 0、1 串来表示数据包的接收和发送,0 表示接收包,1 表示发送包(根据计算需要,有时用 -1 和 1 表示) 。之后对该组序列进行随机性检验计算,包括频度检验、游程检验、前向累加和、后向累加和以及位信息熵。通过这些检验值的计算,可以表现出一组序列的随机性,从而表现出 Meek 流中数据包接收和发送序列的特征。除了接收发送序列特征,本文还选择了长度及个数、长度方差和长度熵 3 类特征。

此外,为了进行 SVM 分类,还需要对这些特征值进行了归一化处理,归一化的目的是将每个特征值都映射成为 0 到 1 之间的实数,之后组成特征向量。具体的特征值和归一化方法如表 1 所示。其中,数据包长度信息熵的归一化处理采用了直接除 3 的形式,是由于该统计量大多数分布在 0 到 3 之间,对于大于 3 的情况,则取 1 为其归一化结果。位信息熵也采取同样的做法。

表 1 特征值和归一化方法

Tab. 1 Flag value and normalization method

特征类型	统计量	归一化方法
长度及个数特征	所有数据包的总长度	所有数据包的总长度/分片的最大总长度
	接收数据包的总长度	接收数据包的总长度/分片的最大总长度
	发送数据包的总长度	发送数据包的总长度/分片的最大总长度
	接收数据包的个数	接收数据包的个数/分片大小
	发送数据包的个数	发送数据包的个数/分片大小
长度方差特征	所有数据包的长度方差	所有数据包的长度方差/所有数据包长度方差最大值
	发送数据包的长度方差	发送数据包的长度方差/所有数据包长度方差最大值
	接收数据包的长度方差	接收数据包的长度方差/所有数据包长度方差最大值
长度熵特征	所有数据包的长度信息熵	所有数据包的长度信息熵/3
	发送数据包的长度信息熵	发送数据包的长度信息熵/3
	接收数据包的长度信息熵	接收数据包的长度信息熵/3
接收发送序列特征	频度	—
	游程个数	游程/分片大小
	前向累加和	前向累加和/分片大小
	后向累加和	后向累加和/分片大小
	位信息熵	位信息熵/2

4 实验评估

4.1 实验环境

内网通过网关连接至 Internet,流量采集程序部署在网关上,这样做可以采集到该局域网内所有的网络流量。对于网关所捕获的网络流量,由一台处理机对其进行识别和分类处理。系统网络拓扑如图 3 所示。

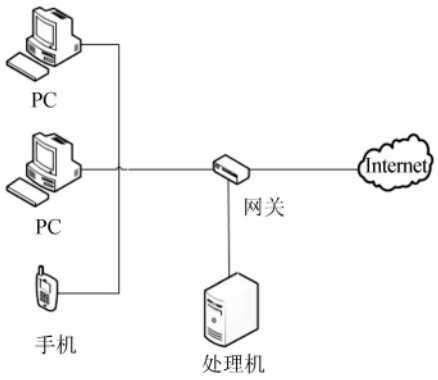

图 3 系统网络拓扑

Fig. 3 Topology of system network

4.2 流量识别的评价标准

本文将使用召回率(*recall*) 和准确率(*accuracy*) 两项指标来评价识别结果。计算方法如下:

$$recall = TP/(TP + FN),$$

$$accuracy = (TP + TN)/(TP + FN + FP + TN)。$$

其中,*TP* 为被分类器正确地划分为正例的实例数,*TN* 为被分类器正确地划分为负例的实例数,*FP* 为被分类器错误地划分为正例的实例数,*FN* 为被分类器错误地划分为负例的实例数。

4.3　基于特征匹配识别的实验评估

基于特征匹配的 Tor 匿名通信识别方法分为: 基于 Meek TLS 静态特征的流量过滤和基于 Meek 流动态特征的流量识别两个步骤。

算法首先通过基于 Meek 静态特征的流量过滤后,目的是排除非 Meek 流的干扰,这样就只需对少数剩下的流量再做进一步的识别判断。因为该过滤方法无法对使用相应版本的 Firefox 浏览器访问 Amazon 等云服务时所产生的 TLS 流和 Tor-Meek 流进行有效区分,因此需要进一步采用基于 Meek 流动态特征的流量识别步骤。该步骤的核心是检测一条待识别流中是否含有 Meek 的轮询请求机制特征。为了验证流动态特征的特异性,本文对捕获的所有 TLS 流都进行了测试,包括通过 Meek TLS 静态特征过滤的 TLS 流和未能通过过滤的 TLS 流。实验发现,除 Meek 流之外,任何其他的 TLS 流都未能满足轮询请求机制的识别条件。

4.3.1　不同长度流的识别准确率

本组实验的目的是考察不同长度的 Meek 流的识别效果。分别将 Meek 流按长度分为小于 40 个包、40 ~ 99 个包、100 ~ 199 个包、200 ~ 299 个包、300 ~ 399 个包和大于 400 个包的 Meek 流。本组实验样本中一共包含 522 条 IPv4 的 Meek 流和 5 217 条 IPv6 的 Meek 流。

分别计算以 Meek 流发起第 1 到第 11 次轮询请求机制作为识别准则的准确率。例如,以发起第 3 次轮询请求机制作为判断标准,那么,如果一条待识别的 Meek 流发起轮询请求机制并增长到第 3 次,则认定该条流为 Meek 流。计算结果如图 4 和 5 所示。

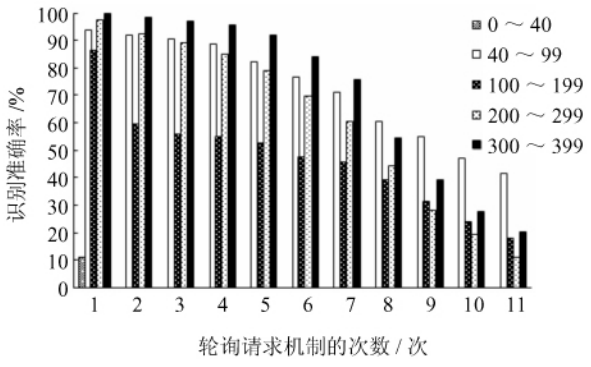

图 4　不同长度 IPv6 流识别的准确率

Fig. 4　Accuracy rate of IPv6's flow recognition

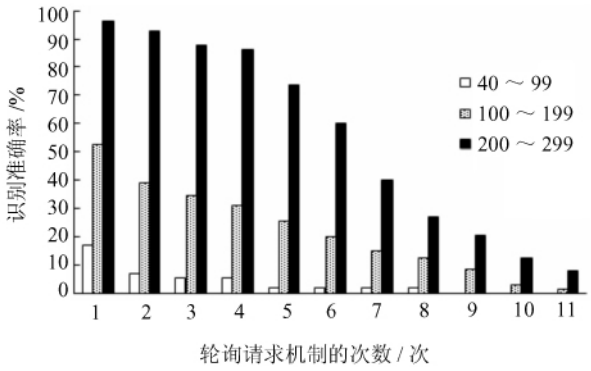

图 5　不同长度 IPv4 流的识别结果的准确率

Fig. 5　Accuracy rate of IPv4's flow recognition

当判定准则设定的轮询次数越多时,就会有更多的 Meek 流被漏报,因此准确率就越低。对于相同的轮询次数,对于长度小于 40 个包的 Meek 流,产生的轮询请求机制次数很少,其中,IPv4 环境下为 0; 。对于长度大于 200 的 Meek 流,有 90% 以上都发起了第 2 次轮询请求机制。而且长度越长的 Meek 流,所发起轮询请求机制的次数越多,识别的准确率越高。

4.3.2　不同 Tor Browser 版本的识别准确率

本组实验的目的是考察不同 Tor Browser 版本以及不同 Meek 网桥的 Meek 流的识别效果。实验数据包含 Tor Browser 4.2 Meek-Amazon、Tor Browser 4.2 Meek-Azure、Tor Browser 5.2 Meek-Azure、Tor Browser 5.5 Meek-Azure、Tor Browser 6.0 Meek-Azure 的 Meek 流,共 5 739 条。计算方式与 4.3.1 节相同,计算结果见如图 6。

通过实验结果可以发现: 对于 Tor Browser 5.X 和 6.X 版本的 Meek 流,有 90% 以上都发起了第 2 次轮询请求机制。对于 Tor Browser 4.X 版本的 Meek 流相对较低,有 70% 以上发起了第 2 次轮询请求机制,分析其原因,主要是在实验时 Tor Browser 4.X 版本过相对较旧,因而连接状态不稳定,产生了大量的重传包对识别效果造成了影响。

4.3.3　基于特征匹配的识别准确率

通过以上 2 组实验的结果可知: 对于长度较短(小于 40 个包) 的 Meek 流,一般检测不到 Meek 的轮询请求机制; 对于长度大于 200 个包的 Meek 流,有 95% 以上可以检测到第 2 次 Meek 的轮询请求机制。结合 Meek 的连接特性来看,可以认为对于长度过短的 Meek 流是无效的 Meek 流,因为即便该 TLS 流是 Meek 流,但它不可能通过 40 个以内的数据包完成 Tor 连接的建立过程,更不能进行后续访问 Tor 的操作,所以对长度过短的 TLS 可以直接排除。在排除长度小于 40 个包的流之后,对不同的 Tor Browser 版本识别的准确率如图 7 所示。

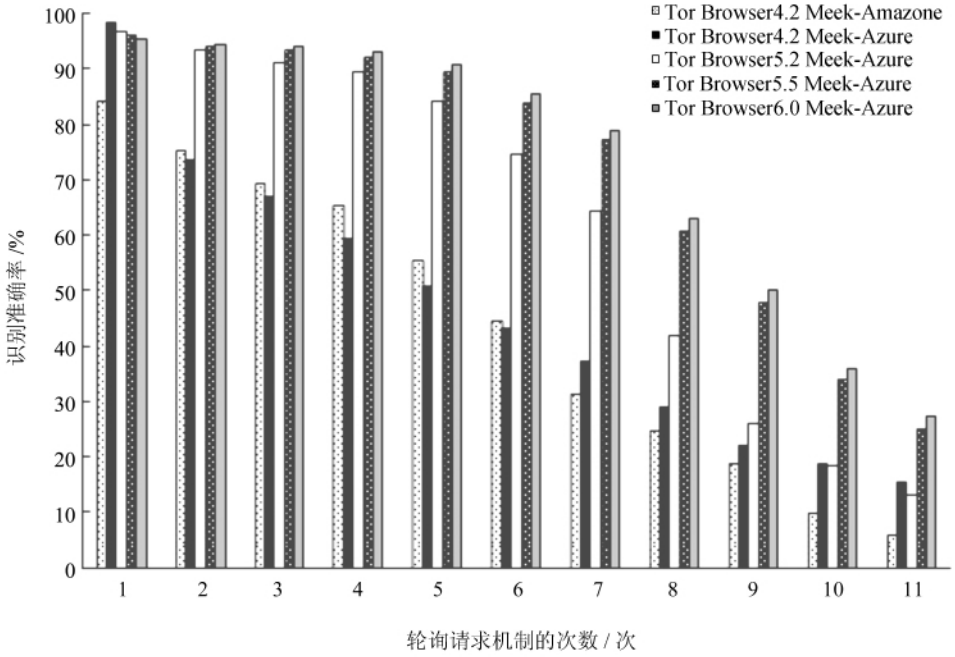

图 6 不同 Tor Browser 版本识别的准确率

Fig. 6 Accuracy rate of tor's different versions

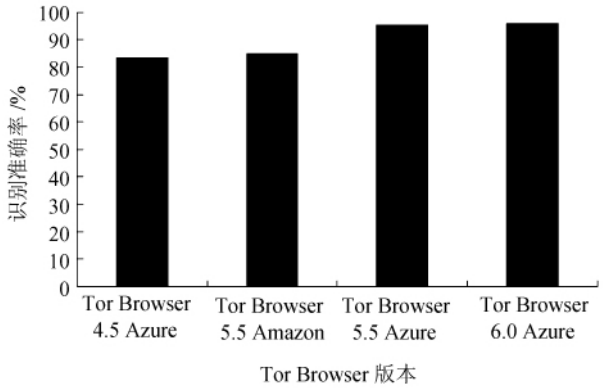

图 7 基于特征匹配识别方法的准确率

Fig. 7 Accuracy rate of flow features matching

4.3.4 实时性分析

基于流特征的识别的计算主要集中在对轮询请求机制的判断上。为评价该方法的识别时间性能，本文对平均长度为 314 个包的 SSL 流执行 1×10^5 次轮询请求机制判断，在除去 IO 操作的耗时后，共耗时 706 ms。基于流特征的识别方法具有较好的实时性。

4.4 基于 SVM 识别的实验评估

基于特征匹配的识别算法容易受到版本变化与更新的影响，本文第 3 节提出基于统计特征的分类算法 可以有效解决该问题。基于 SVM 的 Tor 匿名通信识别的关键部分是 SVM 分类器的构造和 特征值的选取。对于 SVM 分类器，使用 LibSVM 工具包，选择 C-SVC，线性核函数。之后对于惩罚系数 c 的选择，使用不同的值进行多次训练和检测，根据检测结果选取最优解，c 按 2 的指数次幂进行取值，变化范围从 2^0 到 2^{10}。

4.4.1 分片大小对识别准确率的影响

在基于 Meek 流分片的模型中，分片的大小将对识别的结果将产生直接的影响，分片过大或者过小，都会使识别的准确率下降。为了选择最佳的分片大小，需要对不同的分片大小分别进行了测试。本次实验使用 1 000 条 Meek 流和 200 条非 Meek 流作为训练样本，分片大小分别取 20、30、40 和 50 个数据包，之后对不同的分片大小和惩罚系数 c 的组合分别进行多次训练和检测。检测结果见表 2。

表 2 分片大小实验的准确率

Tab. 2 Accuracy rate of fragmentation sizes

分片大小	准确率/%				
数据包数	$c=1$	$c=32$	$c=128$	$c=512$	$c=1\ 024$
20	88.0	92.5	93.0	94.0	93.0
30	89.5	91.5	93.0	93.0	92.5
40	92.0	97.5	97.5	95.5	95.0
50	99.4	99.4	99.4	99.4	99.4

由检测结果可以看出，识别的准确率随着分片大小从 20 增加到 50 是依次递增的，当分片大小为 40 时，识别的准确率可以达到 97.5%（$c=32$ 或 128 时），当分片大小为 50 时，识别的准确率可以达到 99.4%。因此，基于 Meek 流分片模型的识别方法，

对于 Meek 流的识别有较高的准确率。

4.4.2　特征组合的选取对识别准确率的影响

另外一个需要关注的问题是特征组合的选取。下面将通过实验对 4 类特征不同的组合进行评估。测试方法同第 4.4.1 节,选择不同的特征值和惩罚系数 c 的组合进行多次训练和检测,根据检测结果选择最佳的特征值和惩罚系数 c 的组合。

本次实验使用与分片大小实验中相同的训练和检测样本,分片大小取 40。本次实验中选用不同的特征类型的组合分别进行多次训练和检测,其中,A 代表长度及个数特征,B 代表接收发送序列特征,C 代表长度方差特征,D 代表长度熵特征。例如,$A+B$ 即为长度及个数特征和接收发送序列特征的组合。检测结果见表 3。

表 3　特征组合实验的准确率

Tab. 3　Accuracy rate of feature combinations

特征组合	准确率/%				
	$c=1$	$c=32$	$c=128$	$c=512$	$c=1\ 024$
A	95.0	98.0	98.0	98.0	98.0
B	70.5	71.0	71.5	71.5	71.5
C	56.0	65.0	66.5	67.0	67.0
D	60.0	67.5	68.5	68.5	68.5
$A+B$	95.5	99.5	99.5	99.5	99.5
$A+C$	84.5	95.5	93.5	93.0	93.0
$A+D$	97.0	98.0	98.0	99.0	99.0
$B+C+D$	96.5	88.5	90.0	91.0	91.0
$A+C+D$	92.5	93.5	94.0	96.0	97.0
$A+B+D$	97.0	99.5	99.5	99.5	99.5
$A+B+C$	83.0	99.5	99.5	99.5	99.5

由检测结果可以看出,当选用长度及个数特征 + 接收发送序列特征 + 长度熵特征的组合时,分片大小为 40 的准确率可以由之前的 97.5% 上升到最高 99.5%($c=32,128,512,1024$ 时)。因此在本文所以出的基于 SVM 的识别模型中可以选用长度及个数特征、接收发送序列特征和长度熵特征作为 Meek 分片的特征值,以便获得更好识别准确率和识别效率。

4.4.3　参数优化后的识别准确率

根据以上两组实验的检测结果可以得知: 选用分片大小 40,长度及个数特征、接收发送序列特征和长度熵特征 3 类特征作来构造分类器。对不同的 Tor Browser 版本的 Meek 流分别进行了大量的检测,识别的准确率均可以达到 97% 以上,见表 4,识别的召回率 99% 以上,结果见表 5。

表 4　参数优化的识别准确率

Tab. 4　Accuracy rate of optimized identification

版本	准确率/%				
	$c=1$	$c=32$	$c=128$	$c=512$	$c=1\ 024$
5.0.1	92.0	96.6	97.0	97.4	97.4
5.2.1	91.9	96.5	97.1	97.6	97.6
5.5.2	92.2	96.6	97.3	97.7	97.7
5.5.4	92.7	96.8	97.4	97.8	97.8
6.0.5	92.9	97.0	97.4	97.5	97.7

表 5　参数优化的识别的召回率

Tab. 5　Recall rate of optimized identification

版本	准确率/%				
	$c=1$	$c=32$	$c=128$	$c=512$	$c=1\ 024$
5.0.1	100	100	100	100	99.5
5.2.1	100	100	100	100	100
5.5.2	100	100	100	100	100
5.5.4	100	100	100	100	100
6.0.5	100	100	99.4	100	100

4.4.4　实时性分析

基于流特征的识别的计算主要集中在对轮询请求机制的判断上。为评价基于 SVM 的识别方法的识别时间性能,本文对平均长度为 316 个包的 SSL 流执行 1×10^5 次特征提取,在除去 IO 操作的耗时后,共耗时 6 236 ms。分类需要 159 ms,可见基于 SVM 的识别方法具有良好的时间性能。

5　结　论

自从 Tor Browser 4.0 版引入了 Meek 流量传输插件以来,Tor 得到了进一步的混淆和伪装,有效地抵御网络审查和流量分析攻击。本文主要针对 Tor-Meek 匿名通信开展分析和研究。阐述了 Tor 和 Meek 的工作原理,包括 Meek 使用的前置域名技术和基于浏览器代理的加密技术等。并通过对 Meek 进行流量特征分析,总结提出了 7 个 Tor-Meek 的流量特征,包括 Meek 的单一有序连接特征、TLS Client Hello 静态特征、网络空闲时的轮询请求机制动态特征等以及 4 类 16 种统计特征。在此基础上,本文提出了基于特征匹配的 Tor 匿名通信识别方法和基于 SVM 的 Tor 匿名通信识别方法。实验结果表明,以上方法均可有效识别基于云流量混淆 Meek 的 Tor 匿名通信流量,且具有较高的准确率、召回率和实时性能。

未来可在识别 Tor-Meek 匿名通信流量的基础

之上,进一步对其通信的内容进行分类和识别,以便对 Tor 匿名通信流量实施更加有效的监管。此外,Tor 除了使用了 Meek 网桥技术之外,还有多种其他的网桥技术。如何对其他网桥技术实现有效的识别,本文尚未做出研究和讨论,故可以对 Tor 的其他网桥技术开研究和分析。

参考文献:

[1] Pfitzmann A, Köhntopp M. Anonymity, unobservability, and pseudonymity—A proposal for terminology [M]//Designing Privacy Enhancing Technologies. Heidelberg: Springer, 2001: 1 – 9.

[2] Serjantov A. Anonymizing censorship resistant systems [M]//Peer-to-Peer Systems. Heidelberg: Springer-Velag, 2002: 111 – 120.

[3] Reed M G, Syverson P F, Goldschlag D M, et al. Anonymous connections and onion routing [J]. IEEE Journal on Selected Areas in Communication, 1998, 16(4): 482 – 494.

[4] Danezis G, Dingledine R, Mathewson N. Mixminion: Design of a type Ⅲ anonymous remailer protocol [C]//Proceedings of the 2003 IEEE Symposium on Security and Privacy. Oakland, California: IEEE, 2003: 2 – 15.

[5] Reiter M K, Rubin A D. Crowds: Anonymity for web transactions [J]. ACM Transactions on Information and System Security, 1998, 1(1): 62 – 92.

[6] Chaum D. The dining cryptographer's problem: Unconditional sender and recipient untraceability [J]. Journal of Cryptology, 1988, 1(1): 65 – 75.

[7] Sherwood R, Bhattacharjee B, Srinivasan A. P5: A protocol for scalable anonymous communication [J]. Journal of Computer Security, 2005, 13(6): 839 – 876.

[8] Zhao Guofeng, Ji Chaoming, Xu Chuan. Survey of techniques for Internet traffic identification [J]. Journal of Chinese Computer Systems, 2010, 31(8): 1514 – 1520. [赵国锋,吉朝明,徐川. Internet 流量识别技术研究 [J]. 小型微型计算机系统, 2010, 31(8): 1514 – 1520.]

[9] Dingledine R, Mathewson N, Syverson P. Tor: The second-generation onion router [C]. Conference on Usenix Security Symposium, USENIX Association, 2004: 21 – 21.

[10] Tor Project [EB/OL]. [2016 – 04 – 05] https://metrics.torproject.org.

[11] He Gaofeng, Yang Ming, Luo Junzhou, et al. Online identification of Tor anonymous communication traffic [J]. Journal of Software, 2013, 24(3): 540 – 556. [何高峰,杨明,罗军舟,等. Tor 匿名通信流量在线识别方法 [J]. 软件学报, 2013, 24(3): 540 – 556.]

[12] Tor's documentation [EB/OL]. [2016 – 04 – 05]. https://www.torproject.org/docs/documentation.html.en.

[13] Dingledine R, Mathewson N. Tor's protocol specifications [EB/OL]. [2016 – 04 – 05]. http://tor.eff.org/cvs/doc/tor-spec.txt, 2008.

[14] Tor's protocol specifications [EB/OL]. [2016 – 04 – 05]. https://gitweb.torproject.org/torspec.git/tree/attic/bridges-spec.txt.

[15] doc_meek [EB/OL]. [2016 – 04 – 05]. Tor bug tracker & Wiki. https://trac.torproject.org/projects/tor/wiki/doc/meek.

[16] Fifield D, Lan C, Hynes R, et al. Blocking-resistant communication through domain fronting [J]. Proceedings on Privacy Enhancing Technologies, 2015, 2015(2): 1 – 19.

[17] Chaum D. Untraceable electronic mail, return addresses, and digital pseudonyms [J]. Communications of the ACM, 1981, 24(2): 84 – 88.

[18] netMarketShare [EB/OL]. [2016 – 04 – 05]. http://www.netmarketshare.com/.

[19] Vapnik V N. The nature of statistical learning theory [M]. New York: Springer-Verlag, 1995: 112 – 268.

(编辑 赵 婧)

引用格式: He Yongzhong, Li Xiang, Chen Meiling, et al. Identification of Tor anonymous communication with cloud traffic obfuscation [J]. Advanced Engineering Sciences, 2017, 49(2): 121 – 132. [何永忠,李响,陈美玲,等. 基于云流量混淆的 Tor 匿名通信识别方法 [J]. 工程科学与技术, 2017, 49(2): 121 – 132.]