分 类 号_____ 学号 M200971630

学校代码 10487 密级_____

华中科技大学

硕士学位论文

基于条件随机场的入侵检测系统的研究与实现

学位申请人： 熊鋆洋

学 科 专 业： 通信与信息系统

指 导 教 师： 谭运猛 副教授

答 辩 日 期： 2012 年 1 月 4 日

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering

Research and Implementation of Intrusion Detection System based on Conditional Random Fields

Candidate: Xiong Junyang

Major: Communication & Information System

Supervisor: Associate Professor Tan Yunmeng

Huazhong University of Science & Technology

Wuhan, 430074, P.R.China

January, 2012

独创性声明

本人声明所呈交的学位论文是我个人在导师的指导下进行的研究工作及取得的研究成果。尽我所知，除文中已标明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　月　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在_____年解密后适用本授权数。

本论文属于

不保密□　。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　月　日　　日期：　　年　月　日

摘要

分布式拒绝服务（Distributed Denial of Service，DDOS）攻击对网络安全产生了极大的威胁，并且随着针对网络层和传输层 DDOS 攻击的检测技术日趋完善以及应用层的服务越来越复杂，产生了以应用层服务为攻击目标的应用层 DDOS 攻击，这类攻击通过发送合法的请求消耗大量的服务器计算资源，使得攻击难以被检测。

本文在分析正常用户的访问行为和攻击者的访问行为的差异的基础上提出了一种基于条件随机场的应用层 DDOS 攻击检测方法。与以往的对用户访问行为建模的方法相比，我们采用能够表示观察序列上长距离的依赖关系并且有效解决了标记偏置问题的条件随机场来描述用户访问行为。我们从正常用户的访问日志中训练得到一个以条件随机场表示的正常用户访问行为模型，并根据请求序列所表现出的访问行为和正常用户访问行为模型的偏差来判断该请求序列是否是攻击序列。本文采用请求序列的平均熵来衡量请求序列所表现出的访问行为和正常用户访问行为模型的偏差。

本文设计了一个实验来检验本文所提出的攻击检测方法的有效性。我们模拟随机 URL 攻击和重放 session 攻击这两类应用层 DDOS 攻击来构建测试集，利用本文所提出的检测方法对测试集中的所有请求序列进行检测并计算性能评估参数来评价检测方法的有效性。实验结果表明本文提出的攻击检测方法能够极为有效地检测出应用层 DDOS 攻击。

关键字：应用层，分布式拒绝服务，条件随机场，平均熵，访问行为

ABSTRACT

Distributed denial of service (DDOS) attack is a serious threat to the Internet. As for the technologies of detecting the DDOS attacks on the network and transport layer are maturing and the application layer services are becoming more and more complex, resulting in a new type of DDOS attacks, whose targets are application layer services, named application-layer DDOS attacks. Application-layer DDOS attacks utilizing legitimate requests to overwhelm the computing resources of the server are hard to be detected.

In this thesis, we analyze the differences between the browsing behaviors of the normal users and attackers, and then we present a method based on conditional random fields to detect application-layer DDOS attacks. In contrast to prior methods modeling the browsing behaviors of the users, we utilize conditional random fields which are able to represent long-range dependencies of observations and solve the label bias problem to describe the browsing behaviors. We train a normal user browsing behavior model expressed with conditional random fields from the web server log and determine whether a request sequence is an attack sequence according to the deviation between the browsing behavior reflected by the request sequence and the browsing behavior model of the normal users. We use the average entropy of the request sequence to measure the deviation between the browsing behavior of the request sequence and the browsing behavior model of the normal users.

An experiment is designed to test the effectiveness of the proposed detection method. We simulate random-URL attacks and repeated-session attacks to build two test sets, and then we use the proposed method to detect all the request sequences in the test sets and calculate the performance evaluation parameters respectively to evaluate the effectiveness of the presented method. The experiment result shows that the proposed method is very effective.

Keywords: application layer, distributed denial of service, conditional random fields, average entropy, browsing behavior

目录

1 绪论

1.1 研究的背景和意义

随着互联网飞速发展，它已成为当今信息时代的基石，成为人们获取信息的主要来源，渗透到人们工作和生活的方方面面。然而因特网的最初设计目的是保证开放性和可扩展性[1]，而没有过多地考虑安全性，因此因特网容易遭受攻击。同时，各种应用系统的复杂程度也在不断增加，在增加新的功能的同时，也带来了新的漏洞，易于被攻击者发现并实施攻击。在过去的二十年间，随着互联网的飞速发展，网络攻击事件的数量也在逐步升高，而且攻击的形式越来越多样化，攻击的复杂程度也越来越高，网络安全形势越来越严峻。

卡耐基梅隆大学的社区应急响应小组的调查报告显示网络攻击事件的数目从1988 年的 8 起上升到 2002 年的 82094 起，到 2003 年时为 137529 起[2]。根据美国中央情报局和联邦调查局的计算机犯罪和安全调查报告，在 2002 年，被调查组织中有 60%遭受过攻击，而在 2006 年这一比例增长到 72%。我国的网络安全形势同样不乐观，调查显示，在 2006 年有 54%的被调查机构遭受过攻击，2007 年这一比例上升到 65.7%。

分布式拒绝服务（Distributed Denial of Service，DDOS）攻击是最具威胁的攻击方式之一[3]，是在拒绝服务（DOS）攻击的基础上发展起来的。DOS 攻击通过消耗远程主机的资源或网络带宽来使得合法用户得不到服务，其可分为逻辑攻击和泛洪攻击两类[4, 5]。逻辑攻击利用远程服务器上的软件漏洞来使攻击目标崩溃或降低其性能，泛洪攻击是发送大量虚假请求来耗尽目标主机的 CPU、内存或网络带宽。DDOS 攻击通过联合许多被攻击者控制的主机一起对目标主机发起 DOS 攻击[6, 7]，攻击者侵入到网络上的一些防卫能力较弱甚至毫无防卫的主机中安装傀儡程序，从而控制这些主机，使它们成为“僵尸”[8, 9]，大量傀儡主机构成的网络称为“僵尸网络”[10, 11]。由于 DDOS 攻击的分布式特征，同时随着网络性能的提升，攻击者可以控制更多傀儡主机，形成更大规模的僵尸网络，使得其难以防御[12]，并且随着 DDOS 攻击工具的广泛传播，降低了发起攻击的门槛，使得 DDOS 攻击事件时有发生，造成了巨大的经济损失。

2000 年 2 月，Yahoo 和 E-bay 等商业网站相继遭到 DDOS 攻击，致使 Yahoo 停止服务 2 个小时，造成至少 12 亿美元的损失[13]。红色代号蠕虫在 2001 年夏天肆虐，大量主机被感染，造成的损失高达 26 亿美元[14]。

互联网在过去的二十年间得到了跳跃式的发展，随之而来的是网络安全问题，而在各种网络安全威胁之中，DDOS 攻击由于其特有的易操作性、难以防御性和巨大的破坏性，已成为最具威胁的攻击方式，因此，如何有效防御 DDOS 攻击是一个重要的研究课题。

1.2 相关研究

DDOS 攻击自出现之时就展现了它的绝强攻击力，吸引了研究人员的注意，在这些年间取得了一些成果，但 DDOS 攻击技术也在不断改进，对 DDOS 攻击的研究仍在继续。

对 DDOS 的研究大致可分为预防、检测和响应三个方面[15]。

预防是在攻击发起之前采取措施来保护网络免受攻击，文献[16, 17]介绍了带有路由选择的重叠技术来阻止大规模的泛洪攻击。

响应是在攻击发生后采取措施降低攻击带来的损害，一般采取追踪攻击源[18, 19, 20, 21]和通过阻塞攻击包来降低攻击强度[22, 23]两种方式。John Ioannidis 等[18]提出了一种回推架构，在路由器上探测并预先丢弃可能是攻击的包。Hal Burch 等[19]和 Stefan Savage 等[20]介绍了一种不依赖与中间网络服务提供商之间的合作来追踪欺骗包真实来源的方法。Alex C. Snoeren 等[21]采用基于哈希的技术对网络流进行审计跟踪从而追踪到网络中最近发送的单 IP 包的来源。R. Mahajan 等[22]采用在单个路由器上探测和控制流量聚集的本地机制和一个路由器与其上级路由器之间的合作回推机制来探测和控制高带宽流量。Jelena Mirković[23]设计了部署于源端网络的 D-WARD 系统，通过不断监测传输流并周期性地和正常传输流模型进行比较，限制不匹配的传输流的速度，从而保证即使在攻击发生的时候正常传输流依然能够得到很好的服务。

检测是识别正在发生的攻击，一直是研究的重点方向。DDOS 攻击检测按检测模式可分为误用检测，异常检测和混合式检测，按照部署位置分类可分为源端检测、中间网络检测和目的端检测，按照攻击针对的网络层次可分为链路层检测、网络层检测、传输层检测和应用层检测。下面按攻击针对的网络层次来介绍对应的攻击检测方法。

针对链路层的攻击极少，主要是针对 ARP 的攻击。David Whyte 等[24]通过各个网络连接器上 ARP 活动的相关度计算一个总的异常分，从而检测扫描蠕虫在独立网络单元上传播，这一方法能够用很少的扫描次数快速地检测出零日蠕虫。M. Farahmand 等[25]提出了一种在广播域中利用速率、突发度、暗区和顺序扫描这四个

ARP 流标准来检测那些由蠕虫、恶意扫描和非安全配置的服务所引起的速率异常的方法。Y. Yasami 等[27]用改进的隐马尔可夫模型[26]来表示一个基于主机的 ARP 异常检测算法，建立一个统计异常检测系统来区别网络 ARP 流的正常和异常行为。

网络层的攻击方式主要是 ICMP 泛洪攻击。Yann Labit 等[29]提出一种名为 HIDDEN 的检测方法，通过计算 ICMP 请求响应延时时间序列的概率的熵来进行实时检测，并且在时间序列的概率上使用 Hausdorff 距离[28]来进一步降低误检率。J. Udhayan 等[30]提出了无害率的概念，它表示 ICMP 流的生成和传输速率，并通过 ICMP 窗口限制方案来得到无害率，进而从 ICMP 流中移除攻击生产区并促进真实流。

传输层的攻击主要是基于 TCP 和 UDP 协议，主要的攻击方式有 TCP flood 和 UDP flood，是比较常见的两种 DDOS 攻击。Haining Wang 等[31]提出一种基于 SYN-FIN（RST）对的协议行为的检测机制，并应用非参数积累和的方法使检测机制对站点和访问模式不敏感。Vasilios Siris 等[32]采用自适应阈值算法和积累和算法对 SYN flood 攻击进行检测，自适应算法是根据在一个给定的时间间隔内收到的 SYN 包的数目是否超过阈值来进行检测，并且根据对 SYN 包的平均数目的估计来自适应地设定阈值，积累和算法是根据积累量是否大于某个阈值来判断是否为攻击。Xu Rui 等[33]将人工免疫系统应用到入侵检测中，根据 IP 地址的不连续性提出了带有权重的四段检测技术，并引入新的特征值匹配规则来改进否定选择算法，应用检测器和特征值过滤器窗口来检测 UDP flood 攻击。

针对应用层的攻击是 DDOS 攻击发展的新趋势，由于这种攻击方式在传输层及以下各层的表征都正常，因此能够避开针对应用层以下各层的方法的检测，因而对应用层 DDOS 攻击检测技术的研究成为现在的研究热点。T. Yatagai 等[34]对用户访问页面的行为进行了分析，提出了分别基于用户浏览页面的顺序和浏览时间与页面信息量大小相关性的两种检测算法来检测 HTTP-GET flood 攻击。肖军等[35]根据攻击请求的生成方式将应用层 DDOS 攻击分为 5 类，根据正常用户的访问行为和应用层 DDOS 攻击的不同，提出了访问异常属性和 session 异常度模型，可以有效地区分正常 session 和应用层 DDOS 攻击 session。

1.3 本文的主要贡献及组织结构

由于传统的传输层和网络层的 DDOS 攻击的防御技术已经很成熟，而且随着 web 应用的快速发展，应用层的服务越来越复杂，需要越来越多的计算资源，同时网络带宽在不断增加，使得服务器资源成为主要瓶颈[36]，因此 DDOS 攻击已经向

应用层转移，而应用层服务的特点使得应用层的 DDOS 攻击不需要很高的攻击请求速率就能获得很好的攻击效果，并且更加隐蔽，更加难以检测。本文在分析了正常用户访问行为与攻击者的访问行为之间的区别的基础上，针对应用层的 DDOS 攻击提出了一种基于条件随机场的攻击检测方法，该方法采用条件随机场对正常用户访问行为进行建模，计算访问序列所表现出的访问行为与正常用户访问行为模型的偏离度，根据偏离度是否在合理范围来判断此访问序列是否是攻击序列。

本文首先介绍了 DDOS 攻击检测技术的研究背景及意义，回顾了已有的 DDOS 攻击检测技术并指出了 DDOS 攻击及检测技术的发展趋势；然后针对本文的研究侧重点介绍了应用层 DDOS 攻击的一些检测方法和本文所用的核心算法；接着介绍了 web 访问模型，从而提出系统的设计方案；之后详细介绍了系统各模块的实现方式；再通过实验验证了系统的有效性；最后进行总结，指出不足之处与下一步的研究方向。

后续各章的内容安排如下：

第 2 章介绍了应用层 DDOS 检测技术以及本文的核心算法，即条件随机场模型。

第 3 章介绍了用户访问模型并据此提出了系统的设计方案，然后详细阐述了此方案。

第 4 章介绍了系统的具体实现。

第 5 章给出了系统性能评估标准，介绍了实验的设定并对实验结果进行了分析。

第 6 章对本文所做工作进行总结，指出不足之处并对下一步工作进行展望。

2 相关算法和技术

上章中已提到 DDOS 攻击正由传统的网络层和传输层向应用层转移，应用层DDOS攻击具有更好的攻击效果和隐蔽性，因而本文研究的也是应用层DDOS攻击。为此，本章首先简要介绍了一些应用层 DDOS 攻击检测技术，然后对本文提出的检测方法中用于建立用户访问模型的条件随机场进行简单介绍。

2.1 应用层 DDOS 攻击检测技术

应用层 DDOS 攻击与传统的网络层和传输层 DDOS 攻击相比危害更大，更加隐蔽，更加难以检测。由于应用层 DDOS 攻击具有非对称性，客户端只需要少量的资源就能耗尽服务器资源，因此，只需要发送少量的，低速率的请求就能够达到攻击效果，更具隐蔽性；另一方面，由于只需要发送少量的请求，那么就不需要大量的僵尸主机，不需要攻击者建立大规模的僵尸网络，由于客户端只需要少量的资源，那么就降低了对僵尸主机性能的要求，甚至可以通过移动设备（例如智能手机）来发起攻击，使得发起攻击更加容易。被攻击者控制的僵尸主机发出的攻击请求和正常用户发出的请求在内容上是相同的，它们之间的区别仅仅在目的上。攻击者可以通过一个大规模的僵尸网络来发起攻击，僵尸网络中的僵尸主机的地理位置可以相隔很远，不能根据 IP 前缀来过滤攻击请求。应用层 DDOS 攻击通常具有正常的 TCP 连接，使用真实的 IP 地址，传统的基于网络层和传输层的检测技术不再起作用。攻击者不再使用纯粹的带宽泛洪攻击，而是将攻击伪造成突发流[37]，并且模拟正常用户的浏览行为，使得攻击更加隐蔽，从而避开检测。

应用层 DDOS 攻击由于其出色的攻击效果和隐蔽性已经成为黑客们的新宠儿，给网络防御带来了新的挑战，同时，也成为新的研究热点。已有一些研究人员在这个领域取得了一定成效，提出了一些针对应用层 DDOS 攻击的检测技术。根据检测的基础，应用层 DDOS 攻击检测技术可以分为基于攻击特点的检测技术和基于用户行为的检测技术两类。

2.1.1 基于攻击特点的检测技术

这类攻击检测技术的依据是攻击一般是由攻击者注入到僵尸主机中的恶意代码按照预先设定好的机制发起的，具有一个固定的模式，不具备像人一样的思考能力；攻击者扫描整个网站从中随机选取链接作为攻击请求或者以一定的模式顺着页面的链接产生请求序列，这样就有可能是请求正常用户看不到的链接；为了最大化攻击

效果，僵尸主机一般以自己最大的能力发送攻击请求，耗尽自身带宽等攻击特点。

基于图林测试的检测方法[38, 39]采用 CAPTCHA（Completely Automated Public Turing test to tell Computers and Humans Apart）技术向客户端发送一些数字问题或图形测试，让客户端返回答案，如果客户端没有返回答案或者返回错误的答案，则认为是僵尸主机，不再允许其访问，否则认为是正常用户，允许其继续访问。由于人能够很容易地解答出数字问题或识别出扭曲的图形而机器却不能，从而能够区分人和机器。这类方法存在一些缺陷，首先由于生成数字问题或图形以及对答案进行验证的计算较为复杂，需要较多计算资源，使得检测机制本身可以成为攻击目标；其次对于正常用户来说，解决这些问题需要额外的时间，可能会使用户觉得厌烦；最后这种检测方法在拒绝僵尸主机访问的同时也阻止了搜索引擎的爬虫的访问，使得网页不会被搜索引擎收录。检测方法[39]虽然用一定的措施弥补了第一个缺陷，但还是存在后面两个问题。

基于诱饵链接的检测方法[40]在页面中嵌入一些没有语义的或者对正常用户不可见的诱饵链接（例如隐藏在图像映射中的链接），这些链接不会被正常用户访问，但是攻击主机是通过扫描网站从中随机选取链接作为攻击请求或者一定的模式顺着页面的链接产生请求序列，因此可能被攻击主机访问。当诱饵链接被某个主机访问时，就认为其是僵尸主机。

基于一种鼓励机制的检测方法 speak-up[41]鼓励所有客户端加快发送速率，若客户端做出响应，大幅提高了发送速率，则认为是正常用户，若发送速率没有提高则是僵尸主机。判断的依据是攻击主机发送请求的速率一般比正常用户快，攻击主机为了得到最大的攻击效果以尽可能快的速率发送请求，使发送请求的速率接近一个上限（这个上限可能是攻击者为了避开服务器端的检测而设定的一个阈值，也可能是僵尸主机的拥有者设定的一个上行速率最大值），而正常用户还有许多剩余带宽，当服务器鼓励加快发送速率时，正常用户还有较大的上升空间，而僵尸主机已经达到最大发送速率，故不能再提高发送速率。鼓励客户端加快发送速率的方法是以一定的概率随机丢弃请求，然后立即让客户端重发被丢弃的请求，从而使得客户端以更高的速率发送请求。

2.1.2 基于用户行为的检测技术

这类检测技术是建立在正常用户的浏览行为与僵尸主机请求序列所表现出的浏览行为的差异的基础上的。正常用户访问时会有鼠标点击，而僵尸主机没有；正常用户浏览时是顺着页面上的链接浏览自己感兴趣的内容，而僵尸主机可能是随机选

择链接访问或者按固定的模式顺着链接访问；正常用户在打开一个页面之后会查看一段时间，然后再点击链接进入下一个页面，其请求序列的时间间隔表现出一种不均匀分布，而僵尸主机可能是以固定的速率发送请求，其时间间隔是不变的，或者以随机的时间间隔发送请求，其分布与正常用户的分布不一致。基于用户行为的检测技术一般以上面的几点差异为基础，对正常用户行为进行建模，通过比较客户端的浏览行为与模型之间的差异来判断其是正常用户还是攻击主机。

检测方法[42]从系统日志中抽取 session 间隔时间、请求间隔时间和 session 资源消耗量（每个请求消耗的 CPU，磁盘和网络带宽）三个参数来建立合法用户行为的分布集，由此基于一个 session 是由合法用户产生的概率来计算 session 的可疑度。

检测方法[43]用请求速率、页面查看时间和请求序列（例如请求对象及它们的顺序）来描述浏览行为，用隐半马尔可夫模型[44]对正常用户的浏览行为进行建模，根据模型计算用户请求序列的平均熵，若其和正常用户请求序列的平均熵的偏离量大于某个阈值，则判断为异常。

2.2 条件随机场

条件随机场（Conditional Random Fields，CRFs）是由 John Lafferty 等在隐马尔可夫模型和最大熵模型的基础上提出的一种用于序列分段和标注问题的概率模型。

条件随机场可用一个无向图（准确的说是半有向图）表示，如图 2.1 所示。设图 $G=(V, E)$，X 和 Y 是随机变量的集合，随机变量 X_v 或 Y_v 对应图中的顶点 v，在条件 X 下，随机变量 Y_v 对于图服从马尔可夫属性，可表示为 $p(Y_v \mid X, Y_w, w \neq v) = p(Y_v \mid X, Y_w, w \sim v)$，其中 $w \sim v$ 表示顶点 w 和 v 相邻，那么满足上述条件的随机变量集合(X, Y)是一个条件随机场。

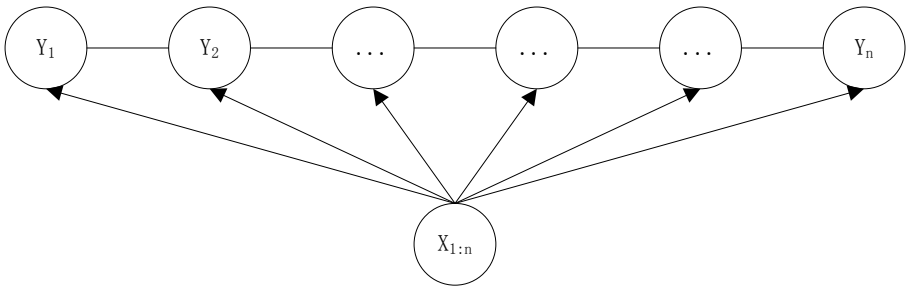

图 2.1 条件随机场

一个最简单情况是图中与 Y 对应的顶点是一个链状结构，此时称之为线性链条件随机场。根据随机场的基础理论，在给定观察序列 X 的情况下，标记序列 Y 的联

合概率分布与指数形式的势函数成正比，如式 2.1 所示，式中 $Z(X)$为归一化因子，t_k 和 s_k 分别为转移特征函数和状态特征函数，λ_k 和 μ_k 为对应的参数。

$$p(Y \mid X)=\frac{1}{Z(X)}\prod_{i=1}^{n}\exp\left(\sum_{k}\lambda_k t_k\left(y_{i-1},y_i,X\right)+\sum_{k}\mu_k s_k\left(y_i,X\right)\right) \quad （式 2.1）$$

在实际应用中，特征函数根据具体问题来定义，一般定义为一个取值是 0 或 1 的函数，即当满足某种条件时取 1，否则取 0，然后采用极大似然估计法从训练数据中估计出各个特征函数对应的参数，就可得到条件随机场模型，然后依此模型来解决实际问题。

对于本文所研究的应用层 DDOS 攻击检测问题而言，用户的访问行为可以通过其请求序列表现出来，而条件随机场是处理序列相关问题的一个非常优秀的模型，因此可用条件随机场来描述用户的访问行为，建立正常用户的访问模型，进而来检测应用层 DDOS 攻击。

2.3 本章小结

本章将现有的应用层DDOS攻击检测技术分为基于攻击特点的检测技术和基于用户行为的检测技术两类并分别进行简单介绍，最后简单介绍了条件随机场模型及其实际应用方式。

3 基于条件随机场的入侵检测系统的设计

本章首先介绍了用户访问 WEB 页面的行为模型，然后根据此模型提出了一种新的应用层 DDOS 攻击检测方法，这一方法用条件随机场对正常用户的浏览行为进行建模，通过计算待检测用户的访问行为与正常用户访问行为模型的偏离度来判断其是否攻击者，最后详细介绍系统各模块的设计方案。

3.1 用户访问模型

一个网页由许多元素组成，要完整地显示一个页面需要层叠样式表（Cascading Style Sheets，css）来控制样式，JavaScript 来实现一些动态效果，需要展示图片，因而需要在网页文件中引入 css 文件、JavaScript 文件和图片文件等其它文件（虽然 css 和 JavaScript 可以直接写入到网页文件中，但网页开发人员为了便于管理和提高重用度会将它们写入到文件中，再在需要的时候引入到页面文件中）。在本文中，网页文件称之谓主页面，被网页文件引入的图片文件等其它文件称之为内嵌对象。

用户进行网上冲浪时，首先通过从浏览器的收藏夹中选择链接，或者手动输入地址，或者从搜索引擎的搜索结果中点击的方式进入感兴趣的站点，这时浏览器向服务器发送一个指向主页面的 HTTP 请求，服务器对该请求做出响应，将对应的主页面发送给浏览器，然后浏览器对接收到的主页面进行解析，接着浏览器向服务器请求主页面引入的各个内嵌对象，当服务器对这些请求做出响应，将这些内嵌对象发送给浏览器之后，浏览器就能显示出完整的页面给用户查看了，之后就是用户查看和思考时间，最后用户从页面中选择一个感兴趣的链接进入下一个页面，重复以上两步，这就是用户访问的基本过程。

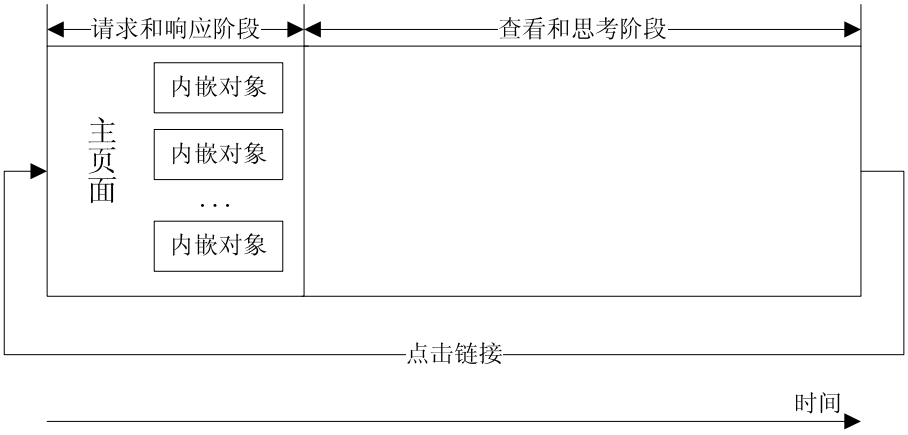

图 3.1 用户访问过程

上述用户访问过程可分为请求和响应阶段与查看和思考阶段两个阶段，如图 3.1 所示。请求和响应阶段是浏览器向服务器请求主页面及其包含的内嵌对象，服务器响应这些请求并将对应的主页面及内嵌对象发送给浏览器，浏览器显示完整页面的阶段，这一阶段的持续时间很短。查看和思考阶段是用户查看浏览器显示的页面并进行思考的阶段，这一阶段的持续时间较长，一般会比请求和响应阶段的持续时间长得多。

用户从开始访问一个站点到最后离开该站点，会按一定的顺序访问站点中的一部分网页，构成一个访问页面的序列。若将一个请求响应阶段与其对应的查看思考阶段看成一个访问单元，这个访问单元对应一个页面的访问时间，那么用户的访问页面序列在时间上表现为一个连续的访问单元序列。用户向服务器发送的请求到达服务器后，服务器在响应的同时会将该请求记录在服务器日志中，那么用户的访问页面序列在服务器日志中体现为一个请求序列。访问页面序列与服务器日志中记录的请求序列的对应关系如图 3.2 所示，每个主页面请求之后跟着一定数量的内嵌对象请求，两个主页面请求之间的时间间隔为一个访问单元，主页面请求及其后面的内嵌对象请求对应一个访问单元的请求和响应阶段，一个页面的最后一个内嵌对象请求与下一个主页面请求之间的时间间隔对应访问单元的查看和思考阶段。

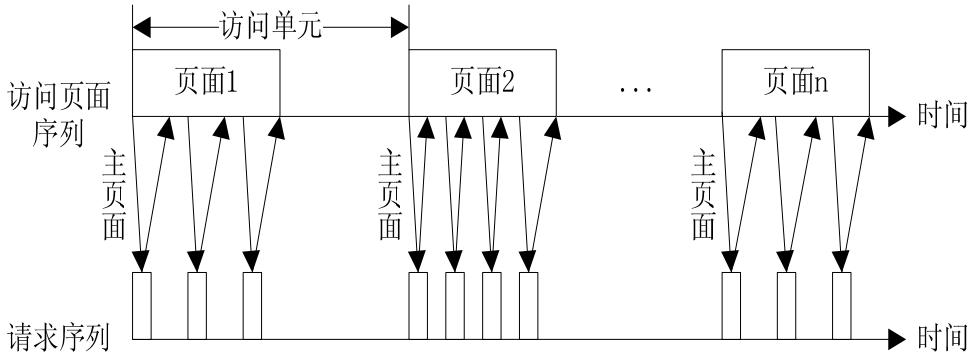

图 3.2 访问页面序列和请求序列关系

用户访问的页面序列体现了用户的浏览行为，而用户访问的页面序列反应在服务器日志记录的请求序列中，因此可以通过分析服务器日志来对用户的浏览行为进行建模。

3.2 系统总体设计

上节介绍了用户访问 WEB 站点的一般过程及其在服务器日志上的体现，本节以这一过程模型和用户访问的一些统计特征为基本依据，在分析正常用户访问行为

和攻击者访问行为的区别的基础上，提出一种利用服务器日志对正常用户的访问行为进行建模，然后以此模型检测应用层 DDOS 攻击的方法，并给出整个检测系统的总体设计方案。

以往的研究指出用户对网页的访问的分布是一个不均匀分布，一些网页经常被访问，而一些网页很少被访问，网站中 10%的网页吸引了 80~90%的用户请求，网页的流行度排名和网页被访问的次数服从 Zipf 分布，说明不同的用户的浏览行为具有集中性，虽然网络技术在不断改变（例如新的协议、脚本语言和缓存架构），但由于这一特性表现了人们在网络上组织、存储和访问信息的基本特点，因此这一特性将长期保持不变。同时一个网站的页面不会频繁改变，即一个网站不会频繁地增加新的页面，或者删除原有的页面，或者改变原有页面的内容，因而用户访问的网页也不会频繁改变，表明用户的访问行为在一定时间内具有稳定性。

用户的访问行为具有集中性和稳定性为本文对正常用户的访问行为建立模型，并以之来检测应用层 DDOS 攻击提供了基本的理论依据。由于用户访问行为具有集中性，用户访问的页面不会随机分布到整个站点的所有网页中，因此正常用户的访问行为可以和攻击者的访问行为区分开来，使得对正常用户访问行为建立模型变得有意义；另一方面，由于用户的访问行为在一定时间内具有稳定性，因此对正常用户访问行为建立的模型不会短时间内失效而不能再正确描述正常用户的访问行为，从而不能以此模型来区别正常用户和攻击者，使得对正常用户的访问行为建模不再具有可行性。

本文提出的基于正常用户行为模型的应用层DDOS攻击的检测方法的基础是正常用户的访问行为和攻击者发送到服务器的请求序列表现出的访问行为是不同的，并假设攻击者不能完全模拟正常用户的访问行为。

攻击者向受害服务器发送的请求序列的构成方式主要有随机选择和重放正常用户请求序列两种方式。随机选择的方式是每次从整个站点中随机选择主页面或内嵌对象作为请求对象构成请求序列，或者以类似搜索引擎爬虫的方式随机选择一个页面作为起始点，然后从这个页面中随机选择一个链接作为下一个请求对象，不断重复这一过程构成请求序列。重放正常用户请求序列的方式是截取正常用户向服务器发送的请求序列，然后不断地向受害服务器循环发送这个请求序列，这种方式更加隐蔽，更加难以检测。本文将以这两种方式构成攻击序列的应用层 DDOS 攻击分别称为随机 URL 攻击和重放 session 攻击。

本文主要考虑的是以上两种攻击请求序列构成方式，攻击者以这两种方式构成的攻击请求序列所表现出的访问行为与正常用户的访问行为有以下几点不同之处：

（1）正常用户一般是点击一个链接进入一个页面，然后查看一段时间，再点击链接进入下一个页面，表现在请求序列上就是一个主页面后跟一定数量的内嵌对象，然后是下一个主页面及其后面的内嵌对象，如此延续。序列中主页面和其上一个内嵌对象的时间间隔较长一些，主页面和其下一个内嵌对象的时间间隔及内嵌对象之间的时间间隔较短一些；主页面以一定的顺序出现，每个主页面之后出现哪些内嵌对象以及这些内嵌对象出现的顺序都是一定的，即每个主页面和其后面的出现的内嵌对象序列存在一个相互依赖关系。正常用户的请求序列如图 3.3 所示。而随机 URL 攻击的请求序列中的每个请求对象都是从整个网站的所有主页面和内嵌对象中随机选取的，因此可能是任意一种形式，可能主页面后出现的内嵌对象序列中有不属于这个主页面的内嵌对象，可能内嵌对象序列是以错误的顺序出现，也可能内嵌对象序列之前缺乏主页面。图 3.4 显示一个随机 URL 攻击序列。

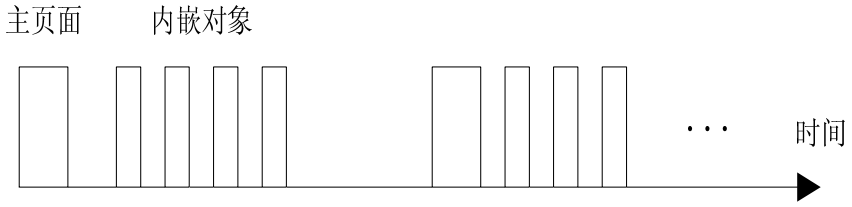

图 3.3 正常用户请求序列

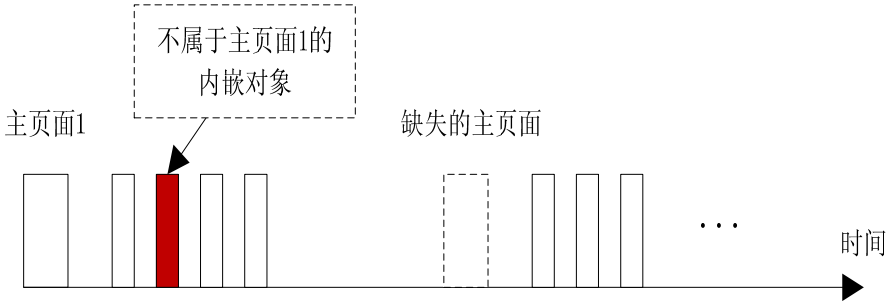

图 3.4 随机 URL 攻击序列

（2）正常用户由于打开一个网页后要查看一段时间再选择一个感兴趣的链接点击进入下一个页面，因而在两个页面之间的间隔时间较长，即访问过程的查看和思考阶段较长一些，表现在请求序列上就是一个主页面和其上一个主页面的最后一个内嵌对象的间隔时间较长，而攻击者为了达到更好的攻击效果，往往会以较快的速度发送请求，若是以重放正常用户访问序列的方式构造攻击序列，那么就会以较快的频率重放序列，从而缩短访问两个页面之间的间隔时间，使得查看和思考阶段较短。

（3）正常用户只访问自己感兴趣的页面，不会不断重复地访问同一个或几个页

面，而重放 session 攻击表现出的行为正是以不同的时间间隔非周期性地不断重复地访问同一个或几个页面。

前面介绍了正常用户的访问行为和攻击序列表现出的访问行为的几点不同，正是由于这几点不同之处使得对正常用户的访问行为建模，并利用所建模型来对正常用户和攻击者进行区分成为可能。本文采用条件随机场这一概率模型来对正常用户的访问行为进行建模，以前的一些研究者也提出了一些采用隐马尔可夫模型、隐半马尔可夫模型和最大熵马尔可夫模型[45, 46]等其他一些概率模型来对用户行为进行建模的方法，与这些概率模型相比条件随机场模型有着固有的优点。

隐马尔可夫模型是一种产生式模型，它表示观察序列和标记序列的一个联合概率分布，因此需要枚举所有可能的观察序列。为了便于处理，产生式模型在观察序列上做出了严格的独立假设，即某一时刻的状态只与这一时刻的状态（标记）有关，而与其它时刻的状态及观察无关，某个时刻的观察是由这个时刻的状态产生的，这就需要观察是适于实际任务的原子实体。而在实际中，观察序列一般存在长距离的依赖关系或者多个相互影响的特征。

最大熵马尔可夫模型是一种判别式模型，它表示一个在给定观察序列的情况下，可能的标记序列的概率，是一个条件概率分布。标记序列的条件概率可以依赖于观察序列的任意的，非独立的特征，这些特征可以在不同的粒度水平上表示观察序列的特性，标记之间的转移概率不仅可以依赖于当前的观察，也可以依赖于过去或未来的观察，因此最大熵马尔可夫模型克服了隐马尔可夫模型在观察序列上有严格的独立假设的缺陷。但是最大熵马尔可夫模型存在标记偏置问题（label bias problem），这是由于转移概率是给定当前状态和观察序列的条件下下一个状态的条件概率，是在每个状态进行局部归一化的，从而导致状态的外出转移极少，甚至只有一个外出转移，而与观察序列无关。

条件随机场模型也是一个条件概率模型，它具有最大熵马尔可夫模型的优点，不存在观察序列上的严格的独立假设，能够表示观察序列上长距离的依赖关系；同时条件随机场模型是单一的指数模型表示在整个观察序列的条件下整个标记序列的联合概率，对整个序列进行全局归一化，从而解决了最大熵马尔可夫模型以及其它基于下一状态分类器的非产生式有限状态模型的标记偏置问题。

根据前文介绍的正常用户的访问行为和攻击序列表现出的访问行为的区别以及条件随机场模型相对于其它可用于对用户访问行为进行建模的概率模型的优点，本文在此提出采用条件随机场对正常用户访问行为进行建模，并以此模型来检测应用层 DDOS 攻击的检测模型，模型框图如图 3.5 所示。检测系统首先从正常用户访问

日志中提取所需信息构成训练数据集，然后对此训练数据集用条件随机场进行训练得到正常用户的访问模型，完成系统初始化。系统完成初始化之后即可进行攻击检测工作，用户的请求序列进入检测系统后，检测系统会根据已有的正常用户访问模型来判别该请求序列是否来自正常用户，如果来自正常用户，则提取该请求序列信息放入训练数据集中，如果来自攻击者，则进行进一步处理（例如提取该请求序列的 IP，将此 IP 列入黑名单，在以后的一段时间内不再接受来自该 IP 的请求）。由于网站的内容每隔一段时间会更新，导致用户的访问行为具有时间局部性，因此需要每隔一段时间对正常用户的访问行为进行更新，用于模型更新的训练数据来自前一段时间内经检测系统判断为正常用户的请求序列。

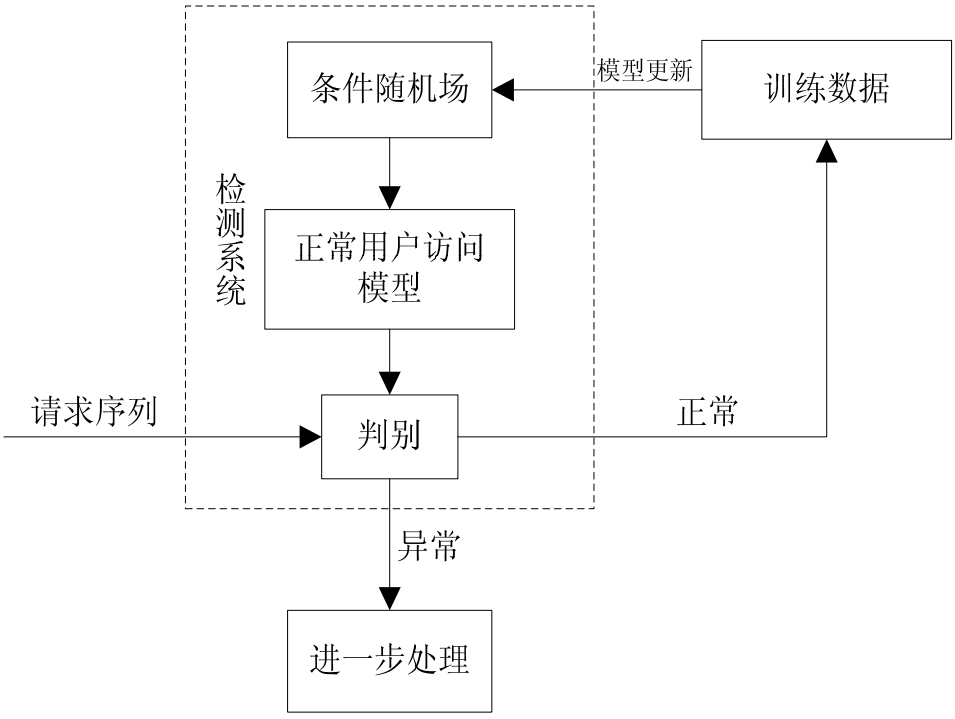

图 3.5 检测模型

从上述攻击检测模型来看检测系统具有从正常用户访问日志中训练得出正常用户访问模型以及根据训练所得正常用户访问模型判别用户请求序列是否是攻击请求序列这两大功能，这两大功能通过正常用户的访问模型联系起来。在训练部分，由于条件随机场处理的观察序列和标记序列，而正常用户的访问日志是按请求到达的时间顺序记录每个请求的信息，不具备这种格式，因此需要提取正常用户访问序列信息，并将其转换为观察序列和标记序列对，即对正常用户访问日志进行数据预处理，将其转化为训练数据集。根据式 2.1 可知，条件随机场概率模型由特征函数及其对应的参数决定，因此需要从训练数据集中抽取能够表现正常用户访问行为特点

的特征函数，并估计出对应参数，训练过程就是提取特征和参数估计的过程。

本文还设计了一个性能评测系统来验证上述应用层DDOS攻击检测系统的有效性。要评价系统的有效性，首先要有一个测试数据集，其中包含正常用户访问序列和攻击序列，正常用户访问序列可以在正常用户访问日志中选取，而攻击序列则需要根据攻击序列的生成方式来模拟；然后利用检测系统对测试数据集中的所有请求序列进行判别，依据判别结果计算相关指标来对检测系统有效性进行评价。

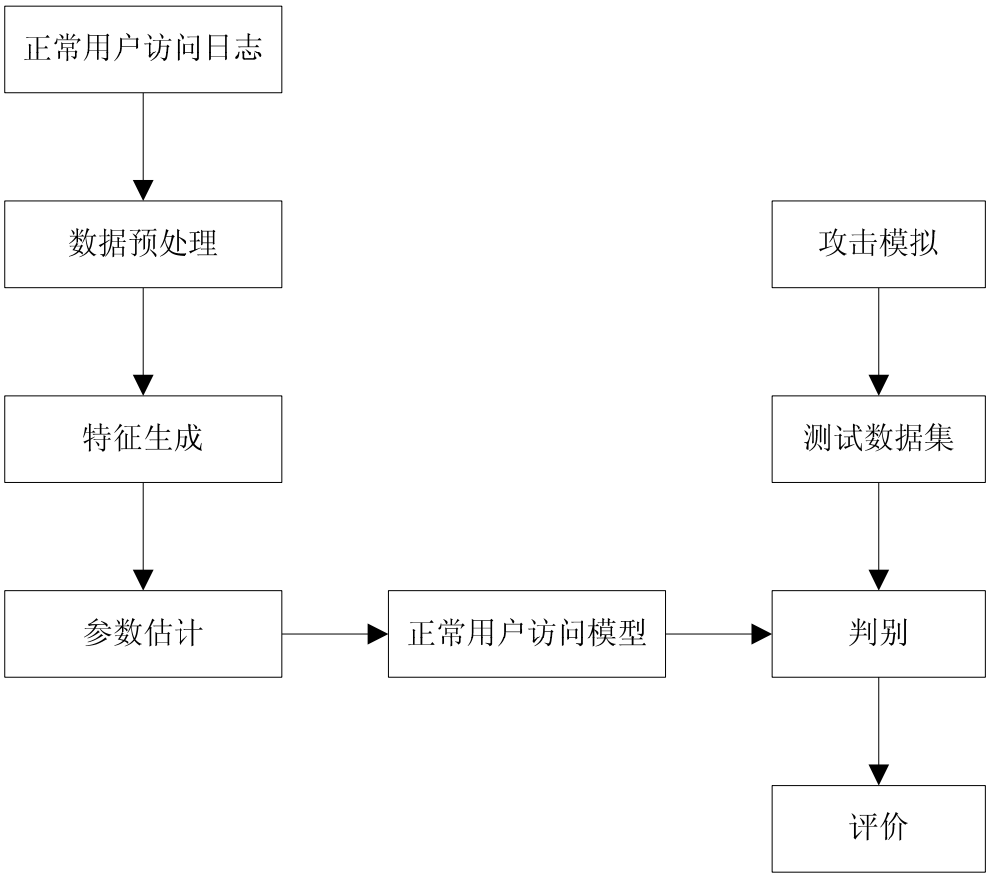

图 3.6 攻击检测和评测系统总体框架

根据上面对攻击检测系统和性能评测系统的描述，本文设计的整个攻击检测和性能评测系统的系统框架如图 3.6 所示，其大致可分为六个模块：

（1）数据预处理模块。该模块对从日志文件中提取正常用户的请求序列，然后将其转化为观察序列和标记序列对，构成训练数据集。

（2）特征生成模块。该模块从训练数据集中提取能够代表正常用户访问行为特点的特征函数。

（3）参数估计模块。该模块用于估计特征函数对应的参数。

（4）判别模块。该模块根据正常用户的访问模型判别输入的用户请求序列是否

是攻击序列。

（5）攻击模拟模块。该模块分别对随机 URL 攻击和重放 session 攻击进行了模拟，构造这两种攻击的请求序列，并加入到正常用户访问序列集中形成测试数据集。

（6）评价模块。该模块利用判别模块对测试数据集中的所有序列进行判别，然后计算相关系统评价指标。

3.3 各模块设计方案

3.3.1 数据预处理模块

本文采用条件随机场从 WEB 服务器日志中训练出正常用户的访问行为模型，服务器日志以请求到达服务器的时间顺序按行记录每个请求的信息，而条件随机场是一个解决序列标注问题的概率模型，其处理的对象是观察序列和标记序列，因此在建立模型之前要预先对服务器日志进行预处理，将其中的相关信息转换成观察序列和标记序列对。

服务器日志每一行的格式如图 3.7 所示，包括请求主机的 IP 地址或域名、请求者的 email 或其它唯一标识符、请求者进行身份验证时的用户名、请求达到服务器的时间，请求方法、请求的资源 URL、协议版本号、服务器的响应状态码和服务器返回给客户端的数据的字节数，其中用户标识符和授权用户名两项一般不记录，用“-”表示。

IP地址	标识符	授权用户	请求时间	方法/资源/协议版本号	状态码	传输字节数

图 3.7 服务器日志格式

由于是按请求到达时间将请求信息记录于日志中的，而每个用户的请求到达时间是相互交错的，一个用户的所有请求在日志中并不是连续出现的，因此需要将每个用户的请求序列从日志中提取出来。此外，用户访问行为是通过请求时间和请求的资源表现出来的，因此对于请求序列中的每一条只需要包含请求时间和请求的资源。为此可以按行读取日志文件，提取每行中的用户 IP 地址、请求时间和请求的资源这三项信息，采用哈希表存储用户请求序列，根据读取到的 IP 地址找到对应的请求序列，将请求时间和请求的资源作为一个请求信息单元加入序列中，最终将用户的 IP 地址或域名和该用户的请求序列一一对应起来，构成一个映射。

提取出用户的请求序列之后，还需要将请求序列转化为观察序列和标注序列对。在本文中观察是请求时间和请求的资源，标注是请求的对象所属的类别，即前文提

到的主页面和内嵌对象，因此请求序列即观察序列，而标注序列则需要对请求序列进行标注得到。之所以将标注设定为主页面和内嵌对象，是因为实际中请求的对象要么是主页面，要么是内嵌对象，如此设定具有实际的物理意义，此外标注的个数决定了计算时所用到的矩阵的阶数，阶数越高，计算的复杂度越高，耗时越长，而只用两个标注可以使计算速度较快。

根据 3.1 节对用户访问模型的描述，用户访问过程中的查看和思考阶段要比请求和响应阶段长得多，表现在请求序列上就是主页面和其上一个内嵌对象的时间间隔比主页面和其下一个内嵌对象的时间间隔以及两个相邻内嵌对象的时间间隔长得多，因此可以根据请求序列中相邻请求的时间间隔来对观察序列进行标注，若当前请求和上一请求的时间间隔大于某个阈值则当前请求就标记为主页面，否则标记为内嵌对象，请求序列的第一个请求标记为主页面。对于阈值的选取，可以对所有用户请求序列的相邻请求时间间隔进行统计，根据请求时间间隔的分布选取一个合适的阈值。

3.3.2 特征生成模块

条件随机场模型由特征函数和对应的参数来确定，在使用条件随机场来解决实际问题时，关键的一点就是要选取合适的特征函数，而本文采用条件随机场对正常用户的访问行为进行建模，因此需要选取能够反映正常用户访问行为的特征函数。

用户的访问行为通过用户的请求序列来体现，也就是通过由请求序列转化而来的观察序列和标记序列对来体现，因此特征函数要表现出用户访问行为和观察序列和标记序列对的内在联系。

用户在点击一个链接之后，会向服务器发送一个主页面请求和一系列内嵌对象请求，主页面决定了其后的内嵌对象序列，一个内嵌对象序列依赖于其前面的主页面，这种决定和依赖不仅体现在主页面后出现哪些内嵌对象上，而且体现在这些内嵌对象出现的顺序上。另一方面，用户点击查看完一个页面之后，下一次点击查看的页面和当前所查看的页面可能有一定的联系，可能是当前页面上的一个链接所指向的页面。

为了体现用户行为的上述特点，可将一个特征函数定义为如果当前请求的资源为 r_0，前一个请求的资源为 r_{-1} 并且当前的标记为 l_0，则函数值为 1，否则为 0，其表达式如式 3.1 所示，这一特征函数通过在使其值为 1 的条件中包含当前请求的资源、前一请求的资源以及当前请求的标记来体现请求序列的顺序，主页面和内嵌对象的依赖关系。

$$s(y_i, X) = \begin{cases} 1 & \mathrm{x}_{i-1} = r_{-1}, x_i = r_0, y_i = l_0 \\ 0 & \text{否则} \end{cases} \quad \text{（式 3.1）}$$

由于请求序列的长度一般都大于 5，因此可以对上述特征函数进行扩展，将特征函数的窗口范围增加到 5，可得到一个新的特征函数，其定义为如果当前请求的资源为 r_0，前面两个请求分别为 r_{-2} 和 r_{-1}，后面的两个请求分别为 r_1 和 r_2，并且当前的标记为 l_0，则函数值为 1，否则为 0，其表达式如式 3.2 所示，这个特征函数是对上一个特征函数的补充，体现了观察序列上更长的依赖关系。

$$\mathrm{s}(y_i, X) = \begin{cases} 1 & x_{i-2\sim i+2} = r_{-2\sim 2}, y_i = l_0 \\ 0 & \text{否则} \end{cases} \quad \text{（式 3.2）}$$

用户访问行为的另一个特点体现在请求时间间隔上。当用户点击一个链接或以其它方式访问一个页面时，首先会向服务器发送一个对主页面的请求，然后服务器将主页面发送给浏览器，浏览器对这个主页面进行解析之后会向服务器发送以此主页面所包含的内嵌对象为目标的请求，当服务器响应这些请求，将主页面包含的所有的内嵌对象发送给浏览器之后，浏览器就能完整显示出用户所访问的页面，用户在查看这个页面一段时间后再点击链接继续访问下一个页面。请求内嵌对象的过程是由浏览器自发完成的，因此主页面和其后面的内嵌对象以及相邻内嵌对象之间的时间间隔较短；而用户查看页面需要的时间要长一些，因此属于前一页面的请求序列的最后一个请求（一般为一个对内嵌对象的请求）与属于下一个页面的请求序列的第一个请求（一个对主页面的请求）的时间间隔相对来说较长。

对于这个由用户访问行为造成的请求序列上各个请求时间间隔的长短交替变化的特点，可以将特征函数定义在时间间隔上。但考虑到受网络环境、页面内容的长短和页面内容吸引人的程度等其它因素的影响，请求时间间隔是在一定范围内波动的，若以类似前面两个特征函数的定义方式将特征函数定义在具体的观察值上，即请求时间间隔上（例如定义为若当前请求与前一请求的时间间隔为 t 则取值为 1，否则取值为 0），那么会使特征函数与当前用于训练的数据，即前一段时间的用户访问情况过于紧密地结合在一起，从而产生过拟合的情况，使得最后训练得出的模型不具有推广性；而且由于具体的时间间隔可能会去取很多值，使得特征函数的个数过多，进而大幅降低计算速度。为此，可对时间间隔进行分段，将特征函数定义为如果当前请求和前一请求的时间间隔在范围 T 中并且当前请求的标记为 l_0，则取值为 1，否则取值为 0，如式 3.3 所示，在使特征函数取值为 1 的条件中加入当前请求的标记可以体现请求时间间隔与请求的文件类型之间的联系。用户行为主要表现在

上面描述的请求时间间隔的一种长短交替的起伏变化上，而且长短时间间隔之间有明显的差距，因此对时间间隔进行合理的分段是能够体现这种起伏变化的，并且不会产生过拟合的情况，同时能够减少特征函数的个数，从而降低计算复杂度，加快计算速度。对时间间隔分段的具体方式，即分为几段，每段的取值范围设为多少，可通过对训练数据集中所有请求序列的所有相邻请求时间间隔进行统计，根据相邻请求时间间隔分布来决定分段的数量及每段的取值范围。

$$\mathrm{s}(y_i, X)=\begin{cases}1 & (x_i - x_{i-1}) \in T, y_i = l_0 \\ 0 & \text{否则}\end{cases} \quad \text{（式 3.3）}$$

基于与定义式 3.2 这一特征函数相同的原因，还可在时间间隔上定义一个与之类似的特征函数，同样取窗口长度为 5，将特征函数定义为如果当前请求和其前面两个请求以及其后面两个请求这 5 个请求的平均时间间隔在某个取值范围 T 内并且当前请求的标记为 l_0，则函数值取为 1，否则函数值取为 0，其表达式如式 3.4 所示。这里对连续 5 个请求的平均时间间隔分段的具体策略同样需要根据训练数据集中所有的请求序列中所有连续 5 个请求的平均时间间隔的分布来制定。

$$\mathrm{s}(y_i, X)=\begin{cases}1 & (x_{i+2} - x_{i-2})/5 \in T, y_i = l_0 \\ 0 & \text{否则}\end{cases} \quad \text{（式 3.4）}$$

如果忽略观察序列，仅从标记序列来看，其体现了请求序列中请求对象类型的变化，请求的资源类型在主页面和内嵌对象之间的转移关系，进而可以从侧面体现用户的访问行为。因此可以定义一个仅涉及请求标记（状态）的状态转移特征函数，其定义为如果当前请求的标记为 l_0 并且前一请求的标记为 l_{-1}，则函数值为 1，否则为 0，其表达式如式 3.5 所示。

$$t(y_i, y_{i-1}, X)=\begin{cases}1 & y_i = l_0, y_{i-1} = l_{-1} \\ 0 & \text{否则}\end{cases} \quad \text{（式 3.5）}$$

对正常用户行为建模就是从正常用户的观察序列和标记序列对中训练得到一个条件随机场的概率模型，这个模型由正常用户请求序列中所有能够体现正常用户访问行为的特征函数来体现，需要从所有的用户观察序列和标记序列对中抽取合适的特征函数。为此，根据上面对特征函数的定义需要对观察序列和标记序列对进行从头到尾的扫描，在每个位置上以此按照各个特征函数的定义提取特征函数。

由于特征函数是一个取值仅为 0 或 1 的二值函数，其值为 0 的时候对计算不产生影响，特征函数本质上是观察序列和标记序列对上曾经出现过的以某种方式进行组合之后的信息，本文称之为特征，因此提取特征函数就是按特征函数取值为 1 的

条件从观察序列和标记序列对中提取相应的特征。

前文根据用户访问行为的特点定义了 5 类特征函数，即 5 种从观察序列和标记序列对中抽取特征的方法，对训练集中所有的观察序列和标记序列对进行扫描，在每个位置分别以这 5 种方法抽取特征，就能得到用于建立正常用户行为模型的所有特征。抽取特征的这一过程相当于依据 5 个模板从观察序列和标记序列对中生成特征的过程，因此可以采用定义模板的方式来提取特征，并可将特征模板写入到一个模板文件中，这样可以很方便地增加或去除特征生成方法，对构成模型的特征函数进行调整，进而对得到的模型进行调整，使之更好地解决实际问题。

本文根据前文描述的 5 类特征函数定义了 5 个相应的特征模板，如表 3.1 所示。每个特征模板都有一个名称以及一个抽取规则，模板所代表的抽取方法由名称和抽取规则唯一决定，不同模板的抽取规则可能相同，例如特征模板 1 和特征模板 3 以及特征模板 2 和特征模板 4 的抽取规则是一样的，但由于它们的名称不同，因此它们是不同的特征模板，以不同的方式提取特征；若抽取规则包含以“/”分隔的两项，那么前一项是对于观察序列的抽取规则，后一项是对于标记序列的抽取规则，若抽取规则只要一项，则根据其名称来决定是对观察序列还是标记序列进行抽取。

表 3.1 特征模板

编号	名称	抽取规则
1	VR1	[-1,0]/[0]
2	VR2	[-2,-1,0,1,2]/[0]
3	VT1	[-1,0]/[0]
4	VT2	[-2,-1,0,1,2]/[0]
5	E01	[-1,0]

对于从训练集中抽取的每个特征应该有一个唯一的标识符使它们能够相互区别开来，这个标识符由特征模板的名称和由这个特征模板抽取出的信息来组成，如果由同一特征模板两次抽取的信息相同，则认为这两次抽取的是同一个特征，如果在两次抽取中，所有的特征模板或抽取出的信息中的任一项不同，则这两次抽取出的特征不同。之所以要在标识符中加入特征模板的名称，是因为考虑到不同的特征模板抽取出的信息可能相同。

3.3.3 参数估计模块

采用条件随机场对用户行为进行建模除了要选择能够反映用户行为的特征函数，从训练数据中提取相应的特征之外，还需要从训练数据中估算出每个特征对应

的参数。

参数估计一般使用极大似然估计，假设给定的训练集包含 N 个观察序列和标记序列对，可表示为 $D=\left\{x^{(i)},y^{(i)}\right\}_{i=1}^{N}$，其中 $x^{(i)}$为观察序列，$y^{(i)}$为标记序列，对数似然函数如式 3.6 所示，式中 λ 为由所有参数构成的向量。在实际应用中特征函数的个数一般有很多，本文的应用情景也是如此，可能导致最后得到的模型过拟合，可以采取对范数过大的参数向量进行惩罚的方式来避免过拟合的情况发生。对式 3.6 加入惩罚因子并将条件随机场模型表达式代入之后，得到一个新的对数似然函数表达式，如式 3.7 所示，式中用 f_k 统一表示转移特征函数和状态特征函数，T 表示序列的长度。

$$l(\lambda)=\sum_{i=1}^{N}\log p\left(y^{(i)}\mid x^{(i)}\right) \qquad \text{（式 3.6）}$$

$$l(\lambda)=\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K}\lambda_k f_k\left(y_t^{(i)},y_{t-1}^{(i)},x_t^{i}\right)-\sum_{i=1}^{N}\log Z\left(x^{(i)}\right)-\sum_{k=1}^{K}\frac{\lambda_k^2}{2\sigma^2} \qquad \text{（式 3.7）}$$

参数估计就是寻找使式 3.7 最大的参数向量 λ，这是一个以 $l(\lambda)$为目标函数的最优化问题，并且由于 $l(\lambda)$是一个凸函数，其局部最优解即为全局最优解，保证了这个最优化问题是可以求解的。对于最优化问题的求解一般采用改进的迭代缩放算法（improved iterative scaling，IIS）[47]和有限内存的 BFGS（L-BFGS）算法 [48]，但 IIS 算法的收敛速度慢，需要的迭代次数太多，训练时耗时过长，而有限内存的 BFGS 算法是二阶收敛的，其计算速度比 IIS 算法快得多，因此本文采用后一种算法来进行参数估计。

采用有限内存的 BFGS 算法进行参数估计的一般过程是首先设置一个参数向量的初始值，然后根据此初始值计算目标函数（即对数似然函数）及其梯度的值，然后用有限内存的 BFGS 算法根据目标函数及其梯度的值计算出一个使目标函数更接近其最小值或最大值的新的参数向量，接着不断循环用新的参数向量计算目标函数及其梯度的值并用这两个值再次计算新的参数向量这一过程，直到满足收敛条件。由上述过程可看出计算目标函数的梯度是参数估计中的一个重要环节，目标函数的偏导数的表达式如式 3.8 所示。

$$\frac{\partial l}{\partial \lambda_k}=\sum_{i=1}^{N}\sum_{t=1}^{T}f_k\left(y_t^{(i)},y_{t-1}^{(i)},x_i^{(i)}\right)-\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{y,y'}f_k\left(y,y',x_i^{(i)}\right)p\left(y,y'\mid x^{(i)}\right)-\frac{\lambda_k}{\sigma^2} \qquad \text{（式 3.8）}$$

式 3.7 中的 $Z(x)$和式 3.8 中的概率 p 可用前向-后向算法的方法来计算，其大概

过程是分别从后往前及从前往后遍历整个序列，每到达一个新的位置时根据上一个位置的后向向量或前向向量计算当前位置的后向向量或前向向量，$Z(x)$等于序列末尾位置的前向向量的各元素之和，概率 p 可由这些前向向量和后向向量以及 $Z(x)$来计算。在计算前向向量和后向向量时，为了避免溢出，需要对它们进行缩放。

在估计出各个特征函数的参数之后，就得到了整个正常用户访问行为模型，需要将此模型存储下来以便以后在对用户请求序列进行判别时使用。需要存储的信息包括特征的个数以及每个特征及其参数，可以采用按行存储的方式，第一行存储特征个数，以后每行存储一条特征信息，包括参数和特征标识符，两者之间用一个分隔符隔开。

3.3.4 判别模块

检测用户请求序列是否是攻击序列是基于正常用户访问行为模型的，因此首先要读取训练得到的模型，进行初始化。由于模型的特征及其参数信息是按行存储在文件中的，因此只需要按行读取模型文件，并按照存储时信息组织的方式进行解析，按照分隔符将参数和特征标识符分开，从特征标识符中解析出特征的信息，然后将它们存储到相应的数据结构中，以便在以后对用户请求序列进行判别时使用。

对用户请求序列进行判别的依据是其表现出的行为和正常用户行为的偏离程度，可以采用平均信息熵来对偏离程度进行度量，式 3.9 给出了对平均信息熵的定义，式中 $p(O)$表示观察序列（即请求序列）的概率，L 为请求序列的长度。观察序列的概率可以用前向算法来计算，其值等于观察序列最后位置的前向向量中各个元素之和。

$$entropy = -\ln\left(p(O)/L\right) \qquad \text{（式 3.9）}$$

之所以用平均熵作为偏离程度的度量标准是因为信息熵会随着序列长度的增加而增大，对于熵的比较只在序列长度相同时才有意义，而实际中用户请求序列的长度是各不相同的。

在计算出用户请求序列的平均熵之后还需要有一个正常用户的请求序列的平均熵的取值范围，若用户请求序列的平均熵在这个范围之内则判定为正常用户，否则为攻击者。正常用户的请求序列的平均熵的取值范围的选取可根据训练集中所有用户请求序列的熵的分布来选择一个合适的置信区间。

3.3.5 攻击模拟模块

为了验证本文所提出的攻击检测方法的有效性，需要建立一个测试数据集，其

中包含正常用户的请求序列和攻击序列，并标记出每个序列是正常序列还是攻击序列。测试集中的正常用户的请求序列可以从正常用户访问日志中随机选取一部分，而攻击序列则需要按照攻击序列的构造方式来模拟产生攻击序列，本文主要考虑的是 3.2 节所描述的随机 URL 攻击和重放 session 攻击，因此只对这两种攻击方式进行模拟。

随机 URL 攻击的请求序列中的每个请求的请求资源都是从整个网站中随机选取的，为此在模拟攻击序列时可从正常用户访问日志所包含的所有请求资源中随机选取一部分来构成攻击序列。对于攻击序列中每个请求之间的时间间隔的取值，即发送请求的速率的设定采用类似于一组随机强度的脉冲的方式，在一个随机长的时间段内以一个随机选取的恒定速率发送请求，然后间隔一个随机长的时段再以一个随机选取的恒定速率发送请求并持续一个随机长的时间，如此延续直到攻击序列的时间长度到达预先设定攻击持续时间，图 3.8 显示了攻击序列的请求速率的设定。

重放 session 攻击的攻击方式是不断向受害服务器发送正常用户的请求序列，模拟这种攻击的请求序列可以从正常用户的访问日志中随机选取一个用户的请求序列，然后在预先设定的攻击持续时间内不断重复这个正常序列来构成攻击序列，两个相邻的正常序列之间的时间间隔是一个在一定范围内均匀分布的随机数，正常序列中每个请求之间的时间间隔保持不变。

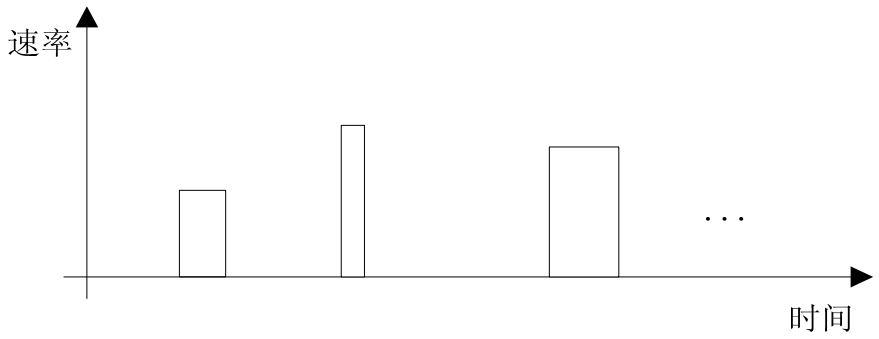

图 3.8 随机 URL 攻击请求速率

将这些依据构建序列构成方式模拟得到的并标记为攻击的请求序列和从正常用户访问日志中随机选取的并标记为正常的一些请求序列写入到文件中就完成了测试数据集的构建。

3.3.6 评价模块

评价检测系统的有效性首先要有个评估标准，本文选取检测率和误检率作为评价标准，检测率是系统检测出的攻击序列的个数与测试集中攻击序列的总数的比率，

误检率是系统将正常序列误判为攻击序列的个数与测试集中正常序列的总数的比率，一个检测系统的检测率越高并且误检率越低说明该系统的有效性越好。

评价检测系统的有效性就是计算检测率和误检率，为此首先要遍历测试集中的所有请求序列，用检测系统对每个请求序列进行判别并记录下判别结果，然后结合训练集中原有的序列是攻击序列还是正常序列的标记来计算检测率和误检率。

3.4 本章小结

本文首先介绍了用户的访问过程模型，接着介绍了用户访问行为具有集中性和稳定性，并结合用户的访问过程模型阐述了正常用户访问行为模型和攻击行为的不同之处，进而提出用条件随机场对正常用户访问行为建模来检测应用层 DDOS 攻击的方法，并给出了检测系统和评测系统的总体设计方案，最后详细介绍整个系统各个模块的设计思路。

4 基于条件随机场的入侵检测系统的实现

上一章介绍了利用条件随机场对正常用户访问行为建模的应用层DDOS攻击检测方法，并设计了一个性能评测系统来对攻击检测系统的有效性进行评价，给出了整个攻击检测系统和性能评测系统的总体设计方案和给个模块的设计思路。本章将在上章所介绍的整个系统设计思路的基础上详细介绍系统所有六个模块的具体实现。

4.1 数据预处理模块的实现

本文所提出的攻击检测方法是利用条件随机场从正常用户访问日志中训练得到正常用户的访问行为模型，并利用此模型计算用户请求序列的平均熵，根据此平均熵的值是否在正常范围内来判断此用户是否是攻击者。正常用户访问日志是以请求到达时间来记录各个用户的请求信息的，而用条件随机场训练用户行为模型时需要的训练数据是一系列观察序列和标记序列对，因而需要将原始的正常用户访问日志转化为与各个用户的请求序列相对应的观察序列和标记序列对，这就是数据预处理模块的主要功能。

数据预处理分两步完成，第一步从日志文件中提取每个用户的请求序列并将它们存放在哈希表中使每个用户的IP或域名与该用户的请求序列对应起来，第二步将用户的请求序列转换成观察序列和标记序列对并存放在哈希表中以供后面的训练部分使用。相应地，数据预处理模块可分为两个子模块：提取用户请求序列子模块和转换子模块。

```
198.206.37.10 - - [06/Aug/1995:00:12:34 -0400] "GET /history/apollo/images/footprint-logo.gif HTTP/1.0" 200 4209
freenet.edmonton.ab.ca - - [06/Aug/1995:00:12:37 -0400] "GET /shuttle/missions/sts-67/sounds/ HTTP/1.0" 200 936
slip14.elysian.net - - [06/Aug/1995:00:12:40 -0400] "GET /images/bluemarb.gif HTTP/1.0" 200 4441
198.206.37.10 - - [06/Aug/1995:00:12:42 -0400] "GET /history/apollo/apollo-10/apollo-10.html HTTP/1.0" 200 3592
```

图 4.1 日志文件片段

提取用户请求序列子模块提取请求序列的方式依赖于日志文件中记录的用户请求信息的具体形式，图 4.1 显示了日志文件的一个片段，这个片段中包含两条请求信息，每条占据一行，每条请求信息包含多个用空格隔开的信息项。建立正常用户行为模型只需要其中的三项：用户IP或域名、请求时间和请求的资源，要从每条请求信息中抽取这三项需要根据请求信息的具体组织形式才能准确地截取所需的信息。

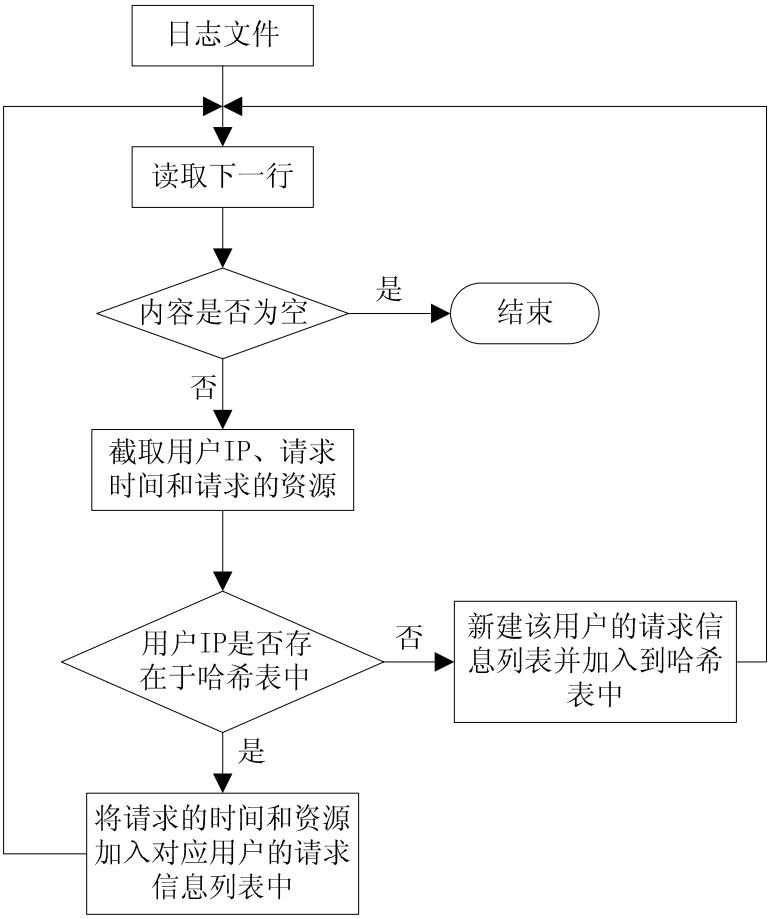

图 4.2 提取请求序列子模块流程

提取用户请求序列子模块的工作流程如图 4.2 所示，其具体步骤如下：

（1）打开日志文件，开始按行读取文件内容。

（2）读取日志文件的下一行，判断读取内容是否为空，若为空，则说明已经到达文件末尾，文件内容已经被全部读取，提取请求序列任务结束，否则进入步骤（3）。

（3）从读取到的一行请求信息中提取用户 IP 或域名、请求时间和请求的资源，并将后两者封装到一个对象中，称之为记录对象。具体的提取方式是截取读取到的字符串从开始到第一个空格之间的子串作为用户 IP，截取字符“[”与“]”之间的子串并将其解析为时间戳作为请求时间，截取两个引号之间的子串作为请求的资源。

（4）判断哈希表中是否已经存在以截取到的用户 IP 或域名作为键的记录，若存在，则进入步骤（5），否则进入步骤（6）。

（5）根据截取到的用户 IP 或域名找到哈希表中对应的请求信息列表，然后将记录对象加入到该列表中，然后进入步骤（2）继续读取下一条请求信息。

（6）新建一个用户请求列表，将记录对象加入到该列表中，然后将以截取到的

用户 IP 为键和以请求列表为值的键值对加如到哈希表中，然后进入步骤（2）继续读取下一条请求信息。

在完成提取之后，哈希表中存放有所有用户的请求序列，其中每个键值对代表一个用户的请求序列，键为用户 IP 或域名，值为该用户的请求信息列表，此列表以请求到达的先后顺序存放记录对象。

转换子模块是将用户的请求序列转换为观察序列和标记序列，观察序列为请求序列本身，即为上一子模块得到的哈希表中存放的用户请求信息列表，而标记序列则是对请求序列进行标记得到。标记的具体方式是若请求序列中相邻请求的时间间隔大于阈值，则标记为主页面，用 0 表示，否则标记为内嵌对象，用 1 表示。

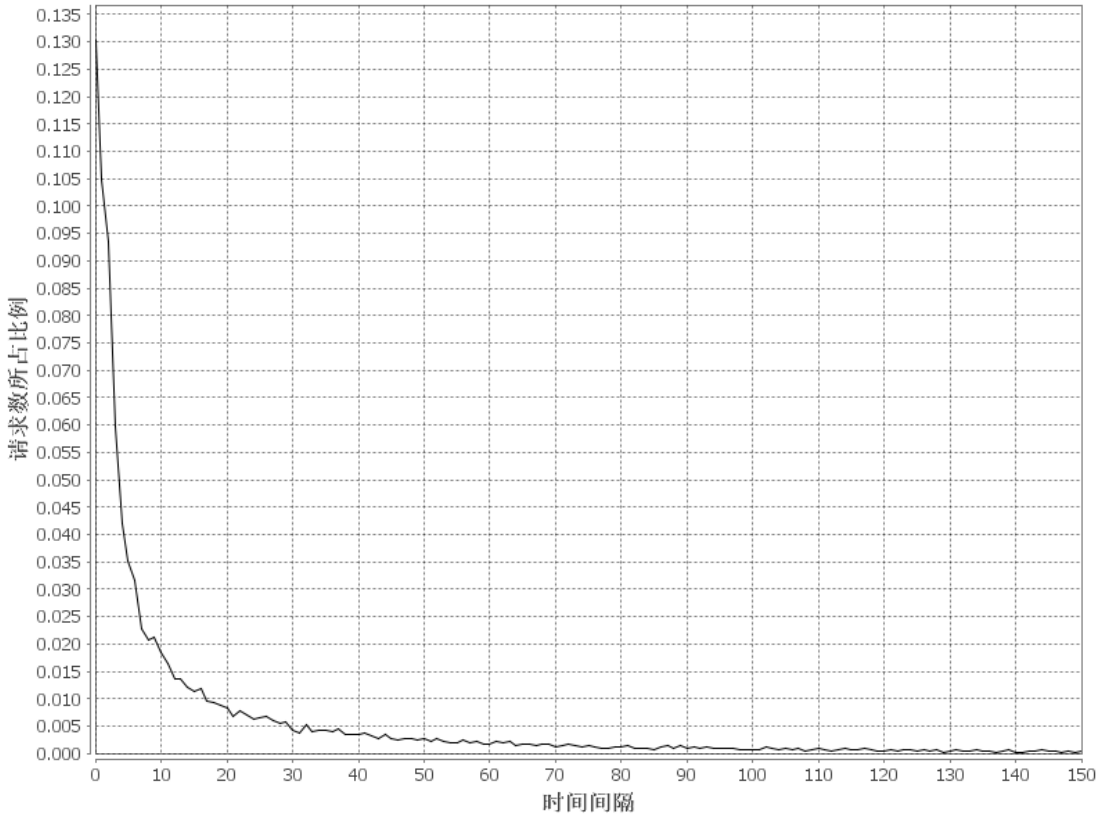

图 4.3 请求时间间隔分布

阈值需要根据所有请求时间间隔的分布来确定。本文采用美国国家航空航天局肯尼迪航天中心的 WWW 服务器在 1995 年 8 月 6 日一天的日志作为训练数据，其中包含的所有用户请求序列的所有的请求时间间隔分布如图 4.3 所示，从图中可以看到请求个数所占比例随时间间隔的增加急剧减少，并且有统计得出请求时间间隔小于 10 秒的请求数大约占总请求数的 60%，这是因为一个主页面后跟一定数量的内嵌对象，用户的请求序列中大多数请求都是内嵌对象，内嵌对象与其前一请求的

时间间隔较短，而主页面与其前一请求的时间间隔较长。从另一个方面考虑，大部分主页面都是以 html 为后缀名的文件，对以这类文件为目标的请求与其前一请求的时间间隔进行统计，得到如图 4.4 所示的分布图，从图中可以看出请求时间间隔在 10 秒处出现了一个峰值，并且统计结果表明仅有大约 15%的请求与其前一请求的时间间隔小于 10 秒。综合以上两个统计结果，本将阈值设为 10 秒。

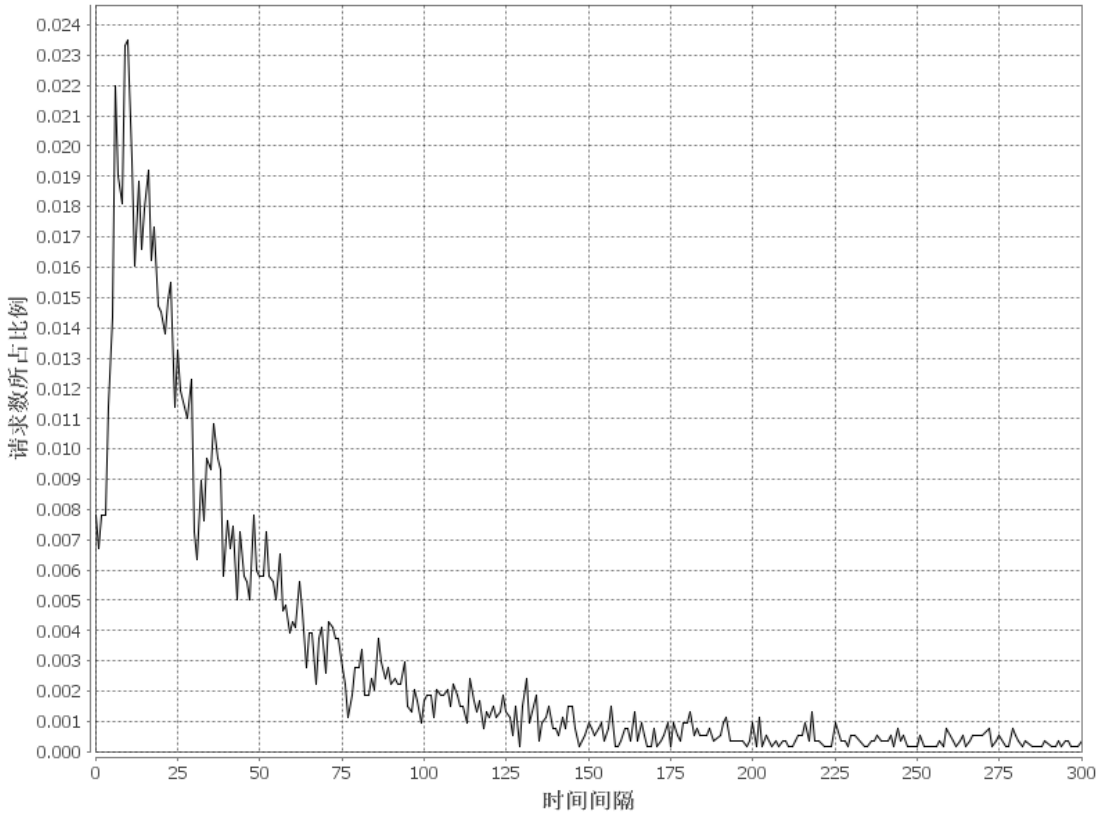

图 4.4 以 html 文件为目标的请求的时间间隔分布

转换子模块的工作流程如图 4.5 所示，转换过程如下：

遍历哈希表中的所有键值对，即所有用户的请求序列，对每个键值对取其值，即请求信息列表，遍历其中的每个记录对象，即遍历请求序列中的每个请求，若当前请求是请求序列中的第一个请求，则将其标记为主页面，否则根据记录对象中记录的请求时间来计算当前请求与上一请求的时间间隔，若时间间隔大于阈值 10 秒，则将当前请求标记为主页面，否则标记为内嵌对象。遍历完一个用户的请求序列后会得到一个对应的标记序列，用一个整型数组表示，将其和一个记录有请求序列中每个请求信息的记录对象数组一起封装到一个观察序列和标记序列对对象中，以这个对象为值，以对应用户的 IP 或域名为键，将此键值对存放到一个哈希表中。

数据预处理模块经过从日志中提取用户请求序列和对请求序列进行标记后得到

每个用户的观察序列和标记序列对，并将每个用户的 IP 或域名与封装该用户的观察序列和标记序列对的对象一一对应存放于一个哈希表中，这样就得到了条件随机场直接可用的数据，以便后面的训练过程使用。

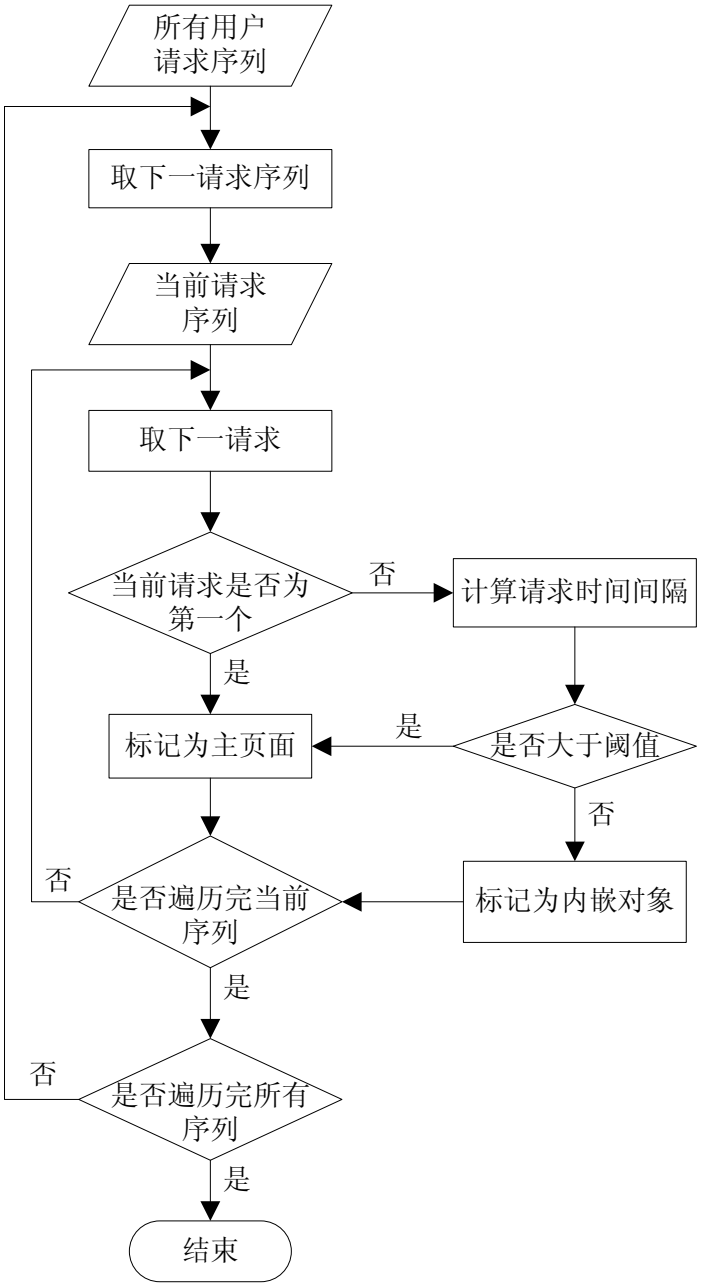

图 4.5 转换子模块流程

4.2 特征生成模块的实现

特征生成模块的主要功能是根据模板文件中的定义的所有特征模板从训练数据

中提取能够反映用户访问行为的特征。

本文的特征模板文件如图 4.6 所示，文件按行存放特征模板，以“#”开始的行为注释，在初始化特征模板文件时不会对改行进行处理，在每一个存放特征模板的行中以“：”将名称和抽取规则分隔开来。模板文件中各个特征模板的抽取规则如下：

（1）特征模板 VR1 抽取观察序列中当前和前一扫描位置的请求资源以及标记序列中当前扫描位置的标记，若当前扫描位置为观察序列和标记序列对的开始位置，则不进行抽取。

（2）特征模板 VR2 抽取观察序列中当前及其前两个和后两个扫描位置的请求资源以及标记序列中当前扫描位置的标记，若当前扫描位置为观察序列和标记序列对开始的两个或最后的两个位置，其前两个或后两个位置不存在，则不进行抽取。

（3）特征模板 VT1 抽取观察序列中当前和前一扫描位置的请求时间以及标记序列中当前扫描位置的标记，然后计算请求时间间隔并按照该时间间隔所在的时间间隔段将其映射到相应的数字上，将这个数字和当前位置的标记组合在一起构成特征。

（4）特征模板 VT2 抽取观察序列中当前扫描位置的前数第二个和后数第二个扫描位置的请求时间以及标记序列中当前扫描位置的标记，然后计算当前及其前两个和后两个这 5 个连续请求的平均时间间隔并按照该时间间隔所在的时间间隔段将其映射到相应的数字上，将此数字和当前扫描位置的标记组合在一起构成特征。

（5）特征模板 E01 抽取标记序列中当前以及前一扫描位置的标记，若当前扫描位置为观察序列和标记序列对的开始位置，则不进行抽取。

```
#状态特征
VR1:[-1,0]/[0]
VR2:[-2,-1,0,1,2]/[0]
VT1:[-1,0]/[0]
VT2:[-2,-1,0,1,2]/[0]
#转移特征
E01:[-1,0]
```

图 4.6 特征模板文件

特征模板 VT1 和 VT2 在抽取特征时需要有一个时间间隔段与数值的映射关系，对应这两个特征模板的时间间隔分段方式都是在对日志中所包含的所有用户的请求序列的时间间隔分布进行分析的基础上制定的，但具体的分段方式又各不相同。

与特征模板 VT1 对应的是对相邻请求的时间间隔进行分段，在 4.1 节中已对日志中所有用户的请求序列的所有相邻请求的时间间隔做过统计分析，其分布如图 4.3

所示。请求的个数占总请求数的比例是随时间间隔的增大而急剧减小的，并且时间间隔小于 20 秒的请求数占据了大约 70%，为此将大于 20 秒划为一段，若时间间隔大于 20 秒，则将其映射为数值 21，另一方面，为了使特征具有一定的敏感度，在小于等于 20 秒的范围内不再进行分段，即以时间间隔本身作为要映射的数值，并且由于日志记录的请求时间是精确到秒的，那么在小于等于 20 秒的范围内的时间间隔就映射为 0 到 20 这 21 个数，最终所有的相邻请求时间间隔映射到 0 到 21 这 22 个数中。

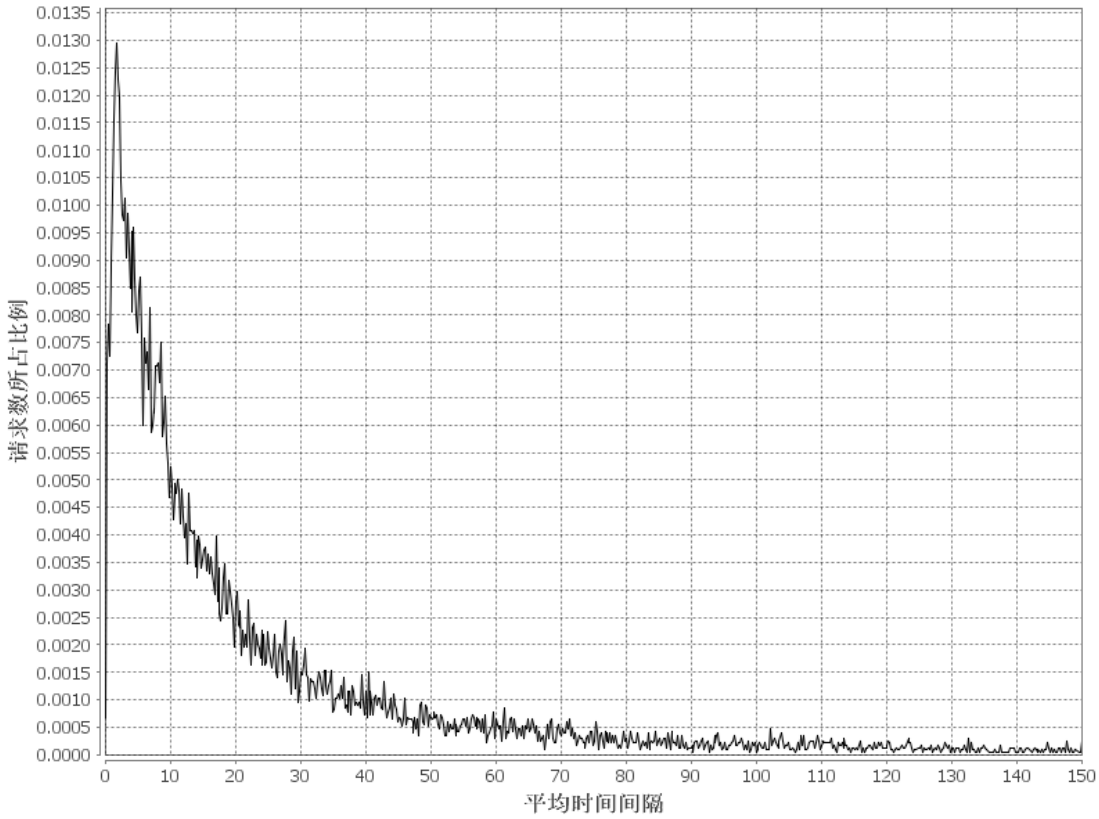

图 4.7 连续 5 个请求的平均时间间隔分布

与特征模板 VT2 对应的是对请求序列中连续 5 个请求的平均时间间隔进行分段，对日志中所包含的所有用户请求序列的所有连续 5 个请求的平均时间间隔进行统计可得如图 4.7 所示的分布图。从图中可看出大部分请求的平均时间间隔集中在前面一段较小的范围内，因此进行分段的一个基本原则段的长度沿着平均时间间隔增加的方向依次递增。由于平均时间间隔分布在 2 秒处有一个峰值，并且平均时间间隔小于等于 2 秒的请求数大约占总请求数的 10%，因此将小于等于 2 秒划为一段。由于在 2 秒之后请求数占总请求数的比例随着平均时间间隔的增大而急剧减小，并综合考虑将生成的特征的个数所带来的计算复杂度以及生成的特征是否能够合理的

反应用户访问行为，因此将大于 2 秒的范围划为 3 段。统计结果表明平均请求时间间隔小于等于 20 秒的请求数大约占总请求数的 60%，所以将大于 20 秒划为一段。在 2 秒到 20 秒之间需要分为两段，统计结果显示平均请求时间间隔在大于 2 秒到小于等于 5 秒这个范围内的请求数大约占总请求数的 15%，平均请求时间间隔在大于 5 秒到小于等于 20 秒这个范围内的请求数大约占总请求数的 35%，这两段的长度分别为 3 和 15，前者中的请求数所占比例比小于 2 秒这段中请求数所占比例大约增加了 5%，后者中的请求数所占比例比前者中的请求数所占比例大约增加了 20%，这两段长度的比值和它们的请求数所占比例相对于各自的前一段的请求数所占比例增加的值的比值是相当的，因此将大于 2 秒到小于等于 5 秒划为一段，将大于 5 秒到小于等于 20 秒划为一段。最终平均时间间隔分为小于等于 2 秒、大于 2 秒到小于等于 5 秒、将大于 5 秒到小于等于 20 秒和大于 20 秒四段，并且处于这四段中的平均请求时间间隔分别被映射为 0、1、2 和 3 这四个数值。

对于每一个生成的特征都有个唯一的标识符来对它们进行区别，在生成特征的过程中，若后面生成的特征的标识符与前面生成的特征的标识符相同，则认为这两个特征是同一特征，不对后面生成的特征进行存储。特征标识符的构成方式如下：

（1）特征模板 VR1 和 VR2 的构造方法是将抽取出的请求的资源按照请求在请求序列中出现的顺序用符号“_”连接起来构成一个字符串，然后对此字符串进行 MD5 摘要，最后用特征模板的名称加上符号“:”，再加上摘要字符串，再加上符号“_”，再加上抽取出的标记就构成特征标识符。

（2）特征模板 VT1 和 VT2 的构造方法是用特征模板的名称加上符号“:”，再加上抽取出的相邻请求的时间间隔或连续 5 个请求的评价时间间隔所在范围对应的数值，再加上符号“_”，再加上抽取出的标记就构成特征标识符。

（3）特征模板 E01 的构造方法是用特征模板的名称加上符号“:”，再加上抽取出的当前扫描位置的前一位置的标记，再加上“-”，再加上抽取出的当前扫描位置的标记。

由上面定义的特征标识符的构成方式可知特征标识符由特征模板的名称、以从观察序列上抽取的内容构成的标识和以从标记序列上抽取的内容构成的标识三部分组成，本文将后两种分别称之为观察标识和标记标识，其中观察标识并不是必需的，由于特征模板 E01 不对观察序列进行抽取，因此由其生成的特征的标识符不包含观察标识。

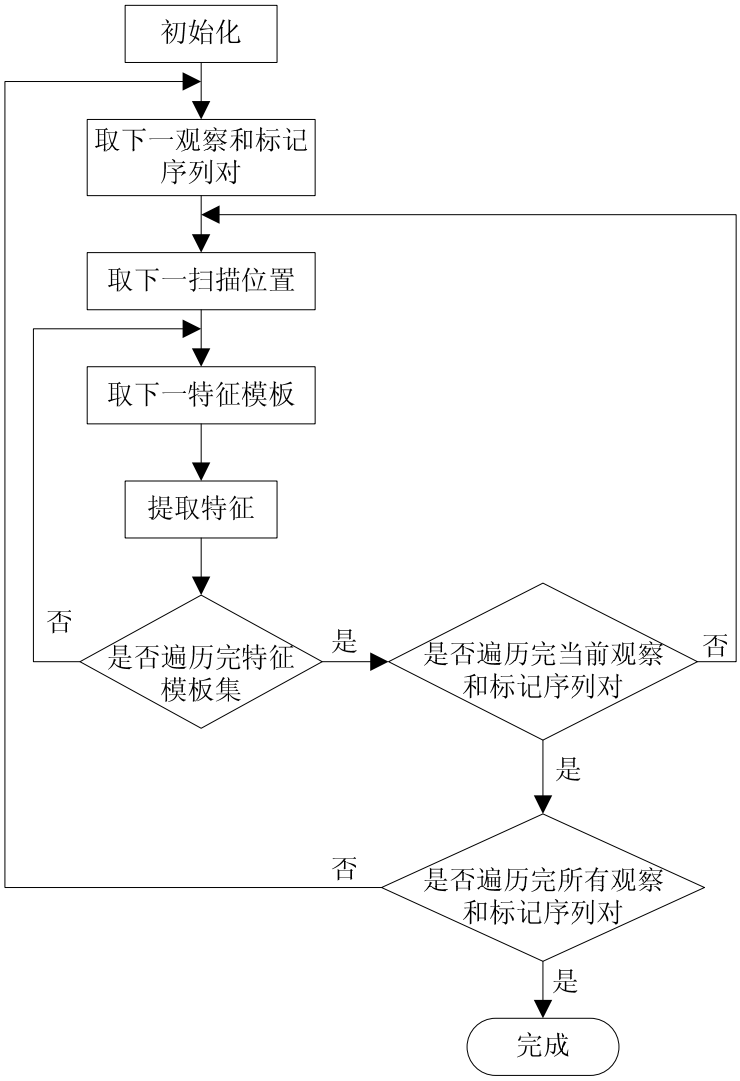

图 4.8 特征生成模块流程

特征生成模块的工作流程如图 4.8 所示，特征生成过程如下：

首先初始化特征模板。按行读取特征模板文件，忽略以“#”开始的注释行，对于其他的记录有特征模板的行，将读取到的字符串以字符“:”为分隔符分为两个子串，前一个子串为特征模板的名称，后一个子串为抽取规则。将抽取规则从字符“/”处分为两个子串，分别从这两个子串中依次提取数字存放到两个数组中，这两个数组分别表示要对观察序列和标记序列进行抽取的相对于当前扫描位置的所有相对位置；若抽取规则中不包含字符“/”，则根据特征模板的名称来将从抽取规则中提取出的数字存放到相应的数组中，并将另一个数组置空。最后将特征模板名称和两个相对抽取位置数组封装到一个对象中，称此对象为特征类型对象或特征模板对象。初始化完成之后就得到一个特征类型对象的列表，构成一个特征模板集。

然后遍历数据预处理模块得到所有用户的观察序列和标记序列对，对每个观察序列和标记序列对从前到后扫描，在每个扫描位置依次取特征模板集中特征类型对象用相应的抽取方法来抽取特征，将抽取出的特征的标识符、观察标识和从标记序列中抽取出的标记封装到一个特征对象中，然后以特征标识符为键，以特征对象为值，将抽取出的特征存放到一个哈希表中。

在对所有用户的观察序列和标记序列对完成特征抽取后，所有在训练数据中出现的特征都被以键值对的方式存储到一个哈希表中，以便在后面的参数估计模块中使用。

4.3 参数估计模块的实现

以条件随机场表示的正常用户访问行为模型由能够反映正常用户访问行为的特征和与每个特征相对应的参数组成，特征是由根据正常用户访问行为的特点及其与攻击行为的差异设定的抽取规则从训练数据中抽取得到的，特征的提取由特征生成模块来完成，而参数则需要通过计算来得到，根据已有的特征估算每个特征对应的最佳参数由参数估计模块来完成。

参数估计的方法是带惩罚的极大似然估计，即计算出一组参数使得带有惩罚项的对数似然函数达到最大值，使用二阶收敛的有限内存的 BFGS 算法来寻找使得对数似然函数最大化的参数。

参数估计的流程如图 4.9 所示，其计算过程如下：

首先为所有参数赋初值，这里将所有参数的初值设为 0；然后根据已有的参数和特征计算对数似然函数的值及其梯度，然后采用有限内存的 BFGS 算法利用已经计算出的似然函数的值及其梯度计算出一个新的参数向量，这个新的参数向量会使对数似然函数值更接近其最大值；然后判断是否满足收敛条件，若满足，则将计算出的参数和对应的特征输出到模型文件中，否则重复上述计算对数似然函数值及其梯度并利用这两个值计算出新的参数向量的过程，直到满足收敛条件；最后，当计算出的参数满足收敛条件，即得到条件随机场模型的最佳参数之后，将得到的以条件随机场表示的正常用户访问行为模型写入到模型文件中，即将能够确定模型的所有特征及其对应的参数写入到文件中，以便判别模块在对用户请求序列进行判别时使用。

收敛条件是梯度的范数与参数向量的范数的比值小于某个值，这个值体现了收敛的精度，本文取 0.001 作为收敛精度。

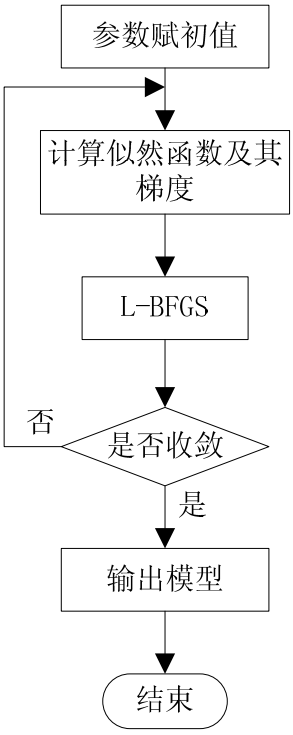

图 4.9 参数估计模块流程

对数似然函数和其梯度的表达式分别为式 3.7 和式 3.8，由这个两个式子可看出计算对数似然函数和其梯度需要遍历训练集中所有用户的观察序列和请求序列对，对每个观察序列和请求序列对的计算结果进行累加，为了简化计算，在计算每个观察序列和标记序列对的部分结果时采用前向-后向算法。对数似然函数值和其梯度的计算流程如图 4.10 所示，其具体计算过程如下：

首先根据已有的参数向量计算似然函数和梯度的惩罚项，即计算式 3.7 和式 3.8 中的第三项；然后遍历训练集中的所有观察序列和标记序列对，对每个观察序列和标记序列对计算式3.7和式3.8前两项对于单个序列对的部分值并累加到最终结果中去，这里将这两个部分值分别称之为序列似然函数值和序列梯度，遍历完之后即得到似然函数值及其梯度。对于每个序列似然函数和序列梯度的计算，首先从后往前扫描观察序列和标记序列对，计算每个位置的后向向量并将它们存放到一个数组中，为了避免溢出，需要将后向向量进行缩放，缩放的方式是将后向向量中的每个元素除以一个缩放因子，缩放因子取后向向量的所有元素之和，并且将每个后向向量的缩放因子存放到一个对应的数组中，以便后面在计算前向向量时以相同的缩放因子对它们进行缩放；然后从前往后扫描观察序列和标记序列对，在每个扫描位置计算前向向量、计算序列似然函数值和序列梯度中对应式 3.7 和式 3.8 中第一项的部分值并累加以及计算利用前向向量和后向向量计算序列梯度对应式 3.8 中第二项的部分

中间结果并累加，在前向遍历完成之后即可得到序列似然函数值对应式 3.7 中的第一项的值和序列梯度对应式 3.8 中的第一项的值；最后，计算归一化因子，其值等于最后一个位置的前向向量的各元素之和，并利用归一化因子计算序列似然函数值中对应式 3.7 的第二项的值，结合前向遍历时计算出的序列似然函数值中对应式 3.7 的第一项的值即可得到序列似然函数值，同时将前向遍历时计算得到的序列梯度中对应式 3.8 中第二项的中间结果向量中的每个元素除以归一化因子即得到序列梯度对应式 3.8 中第二项的值，用前向遍历是计算出的序列梯度对应式 3.8 中的第一项的值减去此值即可得到序列梯度。

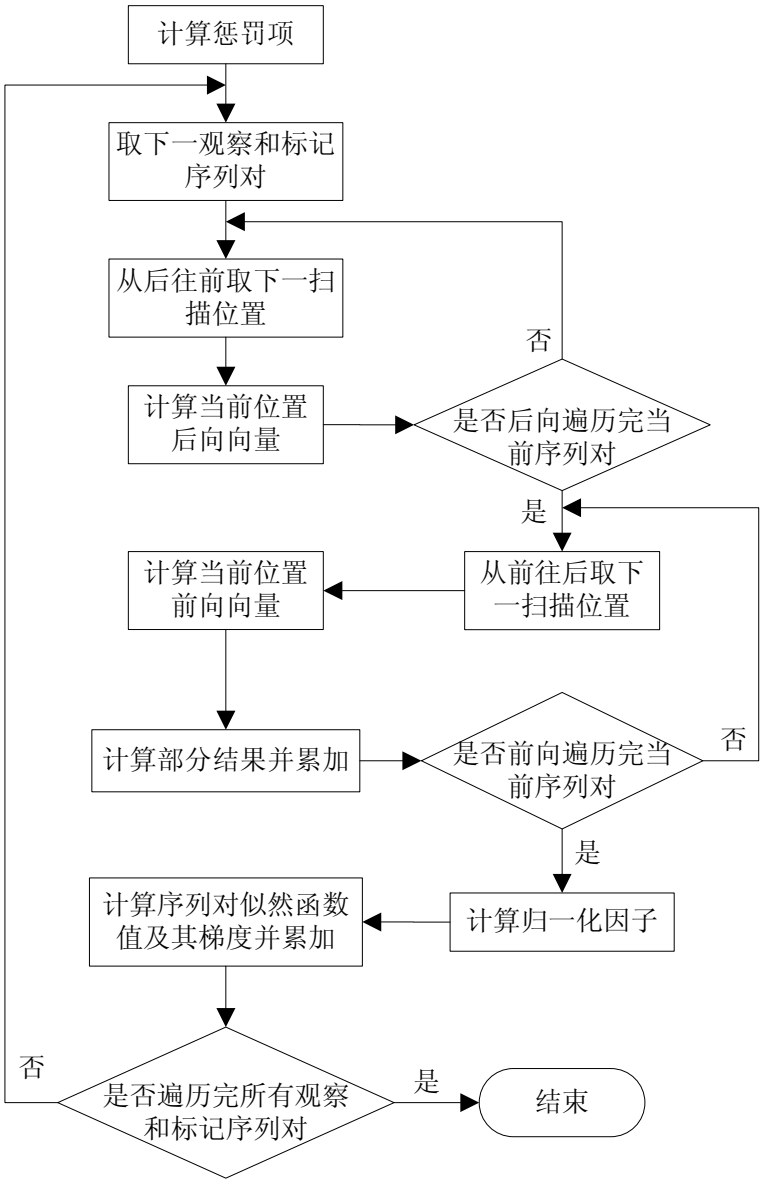

图 4.10 似然函数及其梯度计算流程

在将训练得到的以条件随机场表示正常用户访问行为模型输出到文件时，按行存储特征及其对应的参数，每行中以字符“|”将参数和特征分隔开来，在分隔符之前是参数，之后是特征标识符，并且由于在判别模块初始化模型时需要用到特征的个数，因此在模型文件的第一行存储从训练中抽取的特征的个数。训练得到的模型文件如图 4.11 所示。

```
19836
-0.1505718018527532|VR1:393d03d397d5c4e54654b9f1410cd793_0
0.016261160391685306|VR2:dc0b716f0d99701b43deb82c8fc93e32_1
-0.07727950194533834|VR2:77ae1cc677550a7ef0ec7d6137b10c5b_1
0.07727950194532808|VR2:77ae1cc677550a7ef0ec7d6137b10c5b_0
0.004033475844045256|VR2:8c2bb41d2c5aaf73c4b1fcac7d289de5_1
```

图 4.11 模型文件片段

4.4 判别模块的实现

判别模块的主要功能是利用从正常用户访问日志中训练得到的以条件随机场表示的正常用户访问模型来判别用户请求序列是由正常用户发送的序列还是由攻击者发送的序列。

判别模块分为模型初始化和对用户请求序列进行判别两个部分。模型初始化部分读取训练的得到的模型文件，从中解析出各个特征以及它们各自对应的参数并将它们存放相应的数据结构中以便在对用户请求序列进行判别时使用，模型初始化只需要在判别模块启动时执行一次，每次模型更新之后判别模块都要重新启动一次，以使用新的模型来对用户请求序列进行判别。对用户请求序列进行判别部分利用在模型初始化部分已经读入到内存中的训练所得的所有特征及其对应的参数来计算用户请求序列的平均熵，然后根据熵是否在正常范围内来判断该请求序列是正常序列还是攻击序列。

模型初始化的工作流程如图 4.12 所示，其具体过程如下：

按行读取模型文件，对于读取的第一行，将读取到的字符串转换为对应的整型数，然后新建一个以这个数为长度的数组用以存储参数；对于以后各行，首先根据分隔符“|”将读取的字符串分为两部分，前一部分为参数，后一部分为特征标识符，然后根据特征标识符的构成方式从中解析出观察标识以及从标记序列中抽取的标记，并将特征标识符、观察标识和从标记序列中抽取的标记封装到特征对象中，最后将特征对象中对应域的值设为当前特征标识符和参数行的计数，并将以特征标识

符为键，以特征对象为值的键值对存放到哈希表中，同时将参数字符窜转换为对应的数并将其存放到参数数组中以当前特征标识符和参数行的计数为下标的位置上。

特征标识符和参数行的计数从第一个读取的特征标识符和参数行开始为 0，每读取一个特征标识符和参数行并处理完之后加 1。每一个特征标识符和参数行的参数和特征标识符是对应的，将从每行中解析出的信息所封装成的特征对象的相应域设为该行的计数，并将该行所记录的参数存储到参数数组中以该行计数为下标的位置，这样对于每个特征对象都能在参数数组中找到对应的参数，从而保持原有的特征和其参数的对应关系。

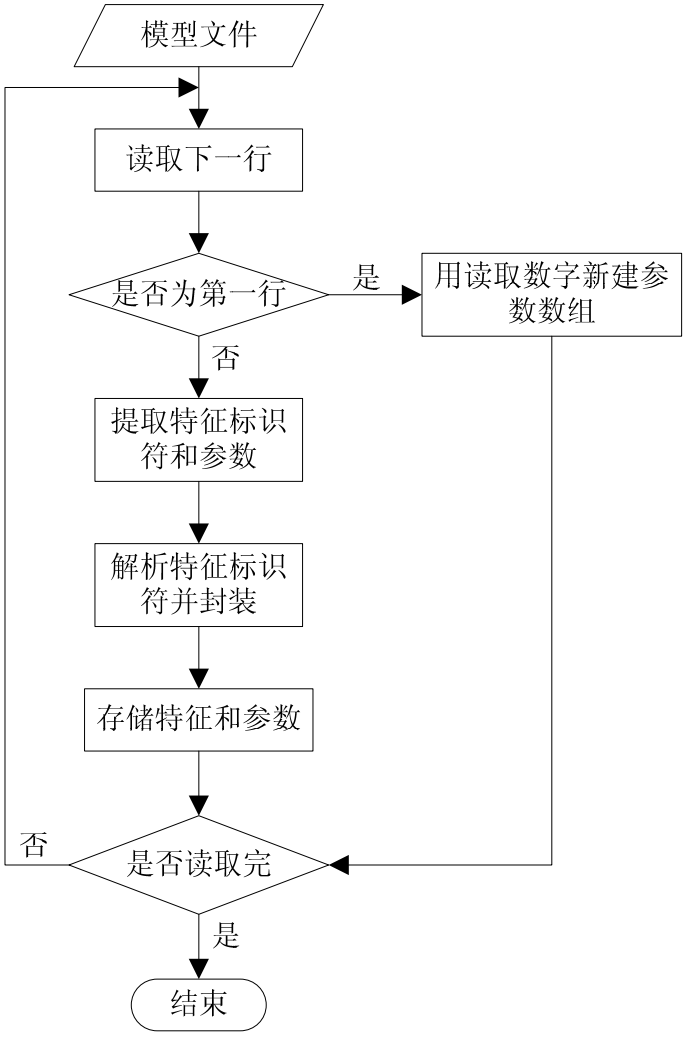

图 4.12 模型初始化流程

对用户请求序列进行判别部分的工作流程如图 4.13 所示，判别过程如下：

首先遍历用户请求序列，即观察序列，在每个位置根据前一位置的前向向量计算当前位置的前向向量，前向向量的初始值为 1，在请求序列的第一个位置由此初

始值来计算当前位置的前向向量；在遍历完成之后按照式 3.9 来计算请求序列的平均熵，对请求序列最后位置的前向向量的各元素求和即得观察序列的概率，将此概率除以请求序列的长度得到一个平均概率，计算此平均概率的自然对数再取反即得请求序列的平均熵；最后根据计算出请求序列的平均熵是否在正常范围内判别请求序列是正常序列还是攻击序列。

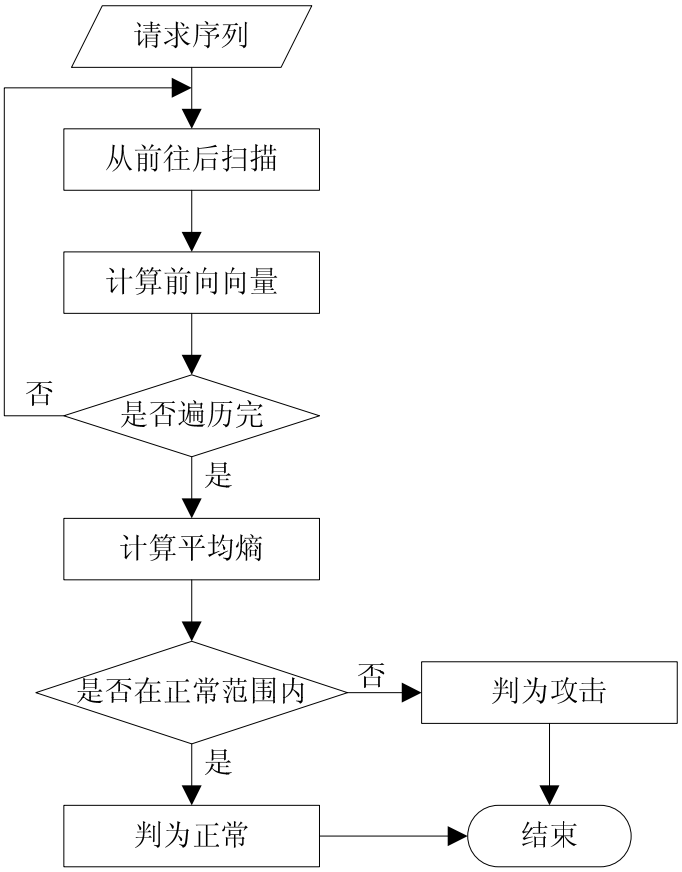

图 4.13 判别部分流程

请求序列平均熵的正常范围需要根据正常用户访问日志中所包含的所有用户的请求序列的平均熵的分布来确定。利用训练得到的以条件随机场表示的正常用户访问模型按上述方法计算训练集中所有用户请求序列的平均熵得到如图4.14所示的正常用户请求序列平均熵的分布，从图中可看出当平均熵大于 2.2 时，请求序列数随着平均熵的增大而急剧减小，在平均熵为 4.0 处出现了最后一个较为明显的请求序列数峰值，平均熵大于 4.0 的请求序列数极少，并且统计结果表明平均熵小于等于 4.0 的请求序列数占据了总的请求序列数的大约 97%，因此将请求序列平均熵的正常范围设为大于等于 0 并且小于等于 4.0。

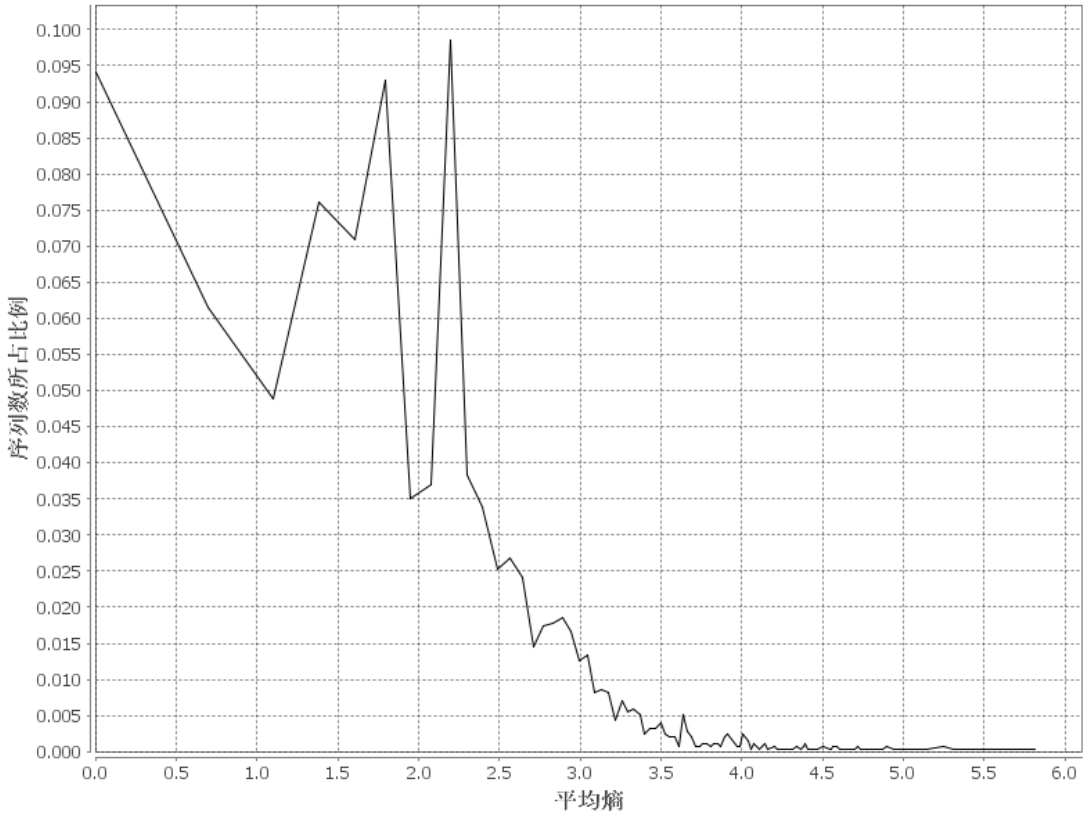

图 4.14 正常用户请求序列平均熵分布

4.5 攻击模拟模块的实现

为了验证本文所提出的基于条件随机场的应用层 DDOS 攻击检测方法有效性，本文设计了一个性能评测系统来对攻击检测系统的有效性进行评测，攻击模拟模块是性能评测系统的一部分，其主要功能是按攻击者构造攻击请求序列的方式模拟产生攻击序列，并将产生的攻击和一些正常用户的请求序列混合在一起构成测试数据集。

本文主要考虑的是随机 URL 攻击和重放 session 攻击这两种攻击方式，因此本文在构建训练集时只以两种攻击方式的请求序列构造方式模拟产生攻击序列，对这两种攻击方式进行模拟并构造测试数据集的具体实现过程将在以下两小节中分别介绍。

4.5.1 随机 URL 攻击模拟

模拟产生随机 URL 攻击序列并构造测试集的工作流程如图 4.15 所示，其模拟过程如下：

首先从日志中所包含的所有请求序列中随机选择一部分用户的请求序列作为测试序列，将这些请求序列存放到哈希表中，哈希表中的每个键值对以用户 IP 或域名加空格加请求序列类型标记“normal”为键，以请求信息列表为值，请求信息列表中的每个元素是记录有请求时间和请求的资源的字符串；然后从这些测试序列中随机选取一部分，将它们的 IP 或域名作为发送攻击请求的主机的 IP 或域名，并将这些 IP 或域名存放到数组中；然后从被选择为攻击主机的序列中选择一个序列的第一个请求的时间作为攻击开始时间；然后从哈希表中删除攻击主机的请求序列；然后遍历所有的攻击主机，为每个攻击主机构造攻击序列，并将以攻击主机的 IP 或域名加空格加请求序列类型标记“attack”为键，以攻击序列的请求信息列表为值的键值对存放到哈希表中；当遍历完成之后即得包含有所有测试数据的哈希表，最后将哈希表中的所有请求序列输出到测试集文件中，请求序列的存储方式是按行存储，首先存储请求主机的 IP 或域名，以后的每行依次存储请求序列的每个请求信息。

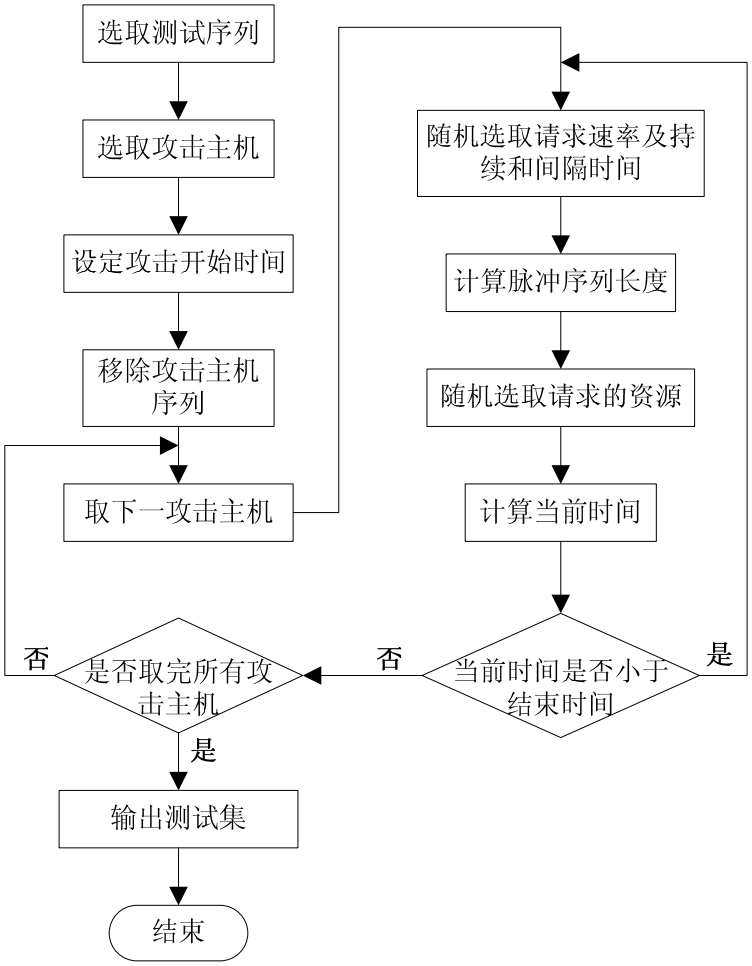

图 4.15 随机 URL 攻击模拟流程

对于每个攻击主机的攻击序列的构造是采用随机强度脉冲的方式，即在一个随机选择的持续时间内以一个随机选择的恒定速度发送请求，然后在间隔一段随机选择的时间后再次以上述方式发送请求，如此重复直到到达攻击结束时间，其具体构造方法如下：

首先将攻击开始时间加上一个随机选择的延时时间作为当前攻击主机的真正攻击开始时间，并将其设为当前时间，然后开始循环构造脉冲请求序列，每构造一个脉冲请求序列后将当前时间加上脉冲请求序列的持续时间和一个随机选取的间隔时间作为新的当前时间，并判断当前时间是否小于结束时间，若是则继续循环构造脉冲请求序列，否则就完成了攻击序列的构造，结束循环。在构造每个脉冲请求序列时，首先随机选择请求速率、脉冲请求序列持续时间和此脉冲请求序列与下一脉冲请求序列的时间间隔，然后根据请求速率和持续时间计算该脉冲序列所含的请求个数，即脉冲请求序列的长度，并从请求资源集中随机选取这一个数的请求资源构成请求资源列表，请求资源集为日志中所包含的所有请求资源，然后根据持续时间和脉冲序列的长度计算每个请求之间的时间间隔，最后遍历请求资源列表，以当前时间作为第一个请求的时间，之后的每个请求的时间为上一请求的时间加上请求时间间隔，将请求时间和请求资源组合在一起构成一条请求信息，遍历完成之后即得到一个脉冲请求序列。

4.5.2 重放 session 攻击模拟

模拟产生重放 session 攻击序列并构造测试集的工作流程如图 4.16 所示，其模拟过程与随机 URL 攻击的模拟过程类似，仅在攻击序列的构造方法上不同，为此本节不再赘述除构造攻击序列之外的流程。

攻击序列的构造过程如下：

首先从正常用户请求序列集中随机选取一个请求序列作为用于重放的序列，并将此序列中每个请求的信息，即请求时间和请求的资源封装到记录对象后存放于列表中，正常用户请求序列集为日志中包含的所有用户请求序列中序列持续时间小于攻击持续时间的一半的所有请求序列；然后遍历记录对象列表，将每个记录对象中保存的请求时间改变为当前请求与上一请求的时间间隔，对于序列中的第一个请求则将其请求时间改变为 0，如此便将请求序列中的相邻请求的时间间隔保存下来以便在重放请求序列时设置每个请求的时间，从而保证重放请求序列的相邻请求时间间隔与原序列一致；然后不断循环重放所选取的请求序列，直到当前时间大于或等于攻击结束时间时结束循环，得到最终的攻击序列。

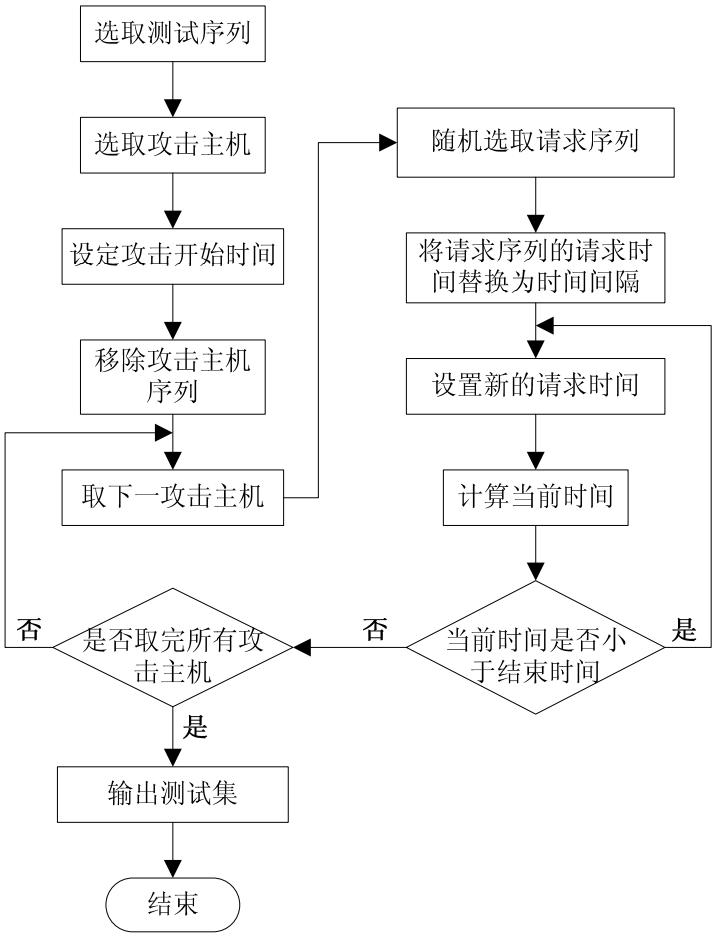

图 4.16 重放 session 攻击模拟流程

重放请求序列的方法是遍历记录对象列表将当前时间加上记录对象中记录的当前请求与上一请求的间隔时间作为请求时间，并将当前时间更新为计算所得的请求时间，然后将请求时间和当前记录对象中记录的请求的资源结合在一起构成一条请求信息存放到攻击序列列表中，最后，在遍历完成之后将当前时间加上一个随机选择的时间间隔作为新的当前时间。

4.6 评价模块的实现

评价模块的主要功能是利用从训练集中训练得到的以条件随机场表示的正常用户访问行为模型对测试集中的所有请求序列进行检测，并根据检测结果和测试集中已有的各请求序列实际是正常序列还是攻击序列的标记来计算评价检测系统有效性的两个参数指标：检测率和误检率，从而为评价本文所提出的利用条件随机场对正常用户访问行为建模的应用层 DDOS 攻击检测方法的有效性提供依据。

评价模块的工作流程如图 4.17 所示，其具体评价过程如下：

首先从测试集文件中读取待检测的请求序列。读取方式是按行读取测试集文件，根据文件中记录请求信息的格式来判断读取的内容是请求信息还是用户 IP 或域名加请求序列类型标记，即若读取的内容中包含字符“[”，则该行记录的是请求信息，否则该行记录的是用户 IP 或域名加请求序列类型标记，若读取的内容是用户 IP 或域名加请求序列类型标记，则新建一个列表，然后将以用户 IP 或域名加请求序列类型标记为键，以该列表为值的键值对存放到哈希表中，并且由于其后面的若干行记录的是对应该用户的请求序列，那么对后面读取的每行，从中解析出请求的资源和请求时间并将它们封转到一个记录对象中，然后将该记录对象添加到对应的列表中，直到读取到下一个记录用户 IP 或域名加请求序列类型标记的行，读取完成之后即得到一个存有测试集文件中所有请求序列的哈希表。

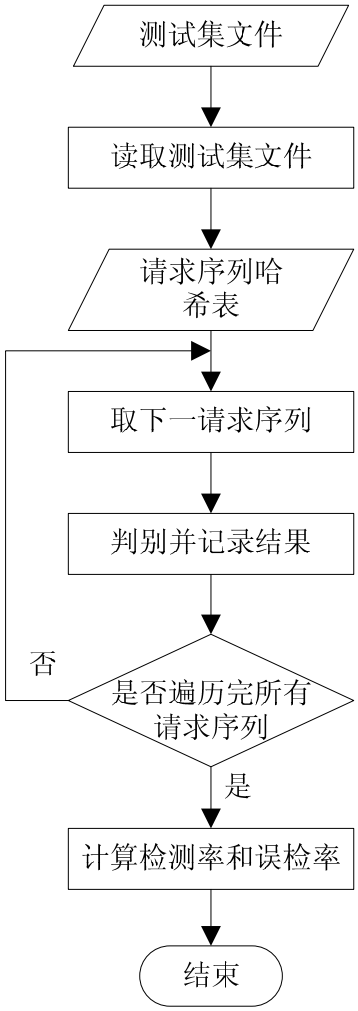

图 4.17 评价流程

然后遍历哈希表中的所有键值对，对于每个键值对，将其键以空格为分隔符分为两部分，前一部分为用户 IP 或域名，后一部分为该请求序列的类型，并将请求序列的类型存放到一个请求序列类型数组的第一个位置，然后利用判别模块对其值，即请求序列进行判别，将得到的请求序列类型存放到请求序列类型数组的第二个位置，然后将以用户 IP 或域名为键，以请求序列类型数组为值的键值对存放到哈希表中，遍历完成之后即得到一个存有所有测试序列的实际类型和经检测系统判别所得类型的哈希表。

最后根据所有测试序列的实际类型和经检测系统判别所得类型来计算检测率和误检率。

4.7 本章小结

本章详细介绍了整个攻击检测和评测系统所包含的数据预处理模块、特征生成模块、参数估计模块、判别模块、攻击模拟模块和评价模块的具体实现方法，描述了各模块的工作流程，并结合具体数据分析了数据预处理模块、特征生成模块和判别模块所涉及的几个阈值的设置。

5 实验与分析

本章将设计一个实验来验证本文所提出的基于条件随机场的攻击检测方法的有效性，并对实验结果进行分析。

5.1 系统性能评估标准

本文采用检测率和误检率攻击检测系统有效性的评估标准。检测率的定义是测试集的所有攻击序列中被攻击检测系统检测为攻击的序列数与测试集中的攻击序列的总数的比值，误检率的定义是测试集的所有正常序列中被攻击检测系统检测为攻击的序列数与测试集中的正常序列数的比值。检测率和误检率的计算公式分别如式5.1 和式 5.2 所示。

$$检测率=\frac{检测为攻击并实际为攻击的序列数}{测试集中实际为攻击的序列总数}\times 100\% \qquad （式 5.1）$$

$$误检率=\frac{实际为正常但被检测为攻击的序列数}{测试集中实际为正常的序列数}\times 100\% \qquad （式 5.2）$$

检测率反应了攻击检测系统对攻击的敏感度和检测攻击的能力，其值越高，则表明攻击检测系统能够检测出越多的攻击，检测攻击的能力越强。误检率体现了攻击检测系统的准确性，其值越小，则表明攻击检测系统将越少的正常序列错误地检测为攻击序列，准确性越高。

对攻击检测系统的评价要同时考虑检测率和误检率。若只考虑检测率，有可能出现攻击检测系统的检测率很高，甚至能够检测出全部的攻击，但同时将大量正常请求序列也检测为攻击，误检率也很高的情况，这样的攻击检测系统会严重阻碍正常用户的访问，显然是不适用的。另一方面，若只考虑误检率，有可能出现的情况是攻击检测系统的误检率很低，甚至不会将正常用户的请求序列错误地检测为攻击，但同时也将大量的攻击序列认为是正常用户的请求序列而让其通过，攻击检测系统的检测率也很低，这样的攻击检测系统会让大部分攻击到达服务器，没有防护作用。

一个有效的攻击检测系统要同时具有很高的检测率和较低的误检率，一般优先考虑检测率。

5.2 实验设置

本文所设计的实验的大致流程是首先利用真实的 WEB 服务器日志训练出一个以条件随机场表示的正常用户访问模型，然后根据随机 URL 攻击和重放 session 攻击的攻击序列构造方式模拟生成这两种攻击形式的攻击序列并和正常请求序列混合在一起构成测试集，最后对测试集中的所有请求序列进行检测，并利用检测结果和测试集中记录的请求序列的真实类型来计算攻击检测系统的检测率和误检率来评估检测系统的有效性。

本文所采用的日志是美国国家航空航天局肯尼迪航天中心的 WWW 服务器在 1995 年 8 月 6 日一天的用户访问日志，并从中选取 75%的用户请求序列作为训练数据，从剩下的 25%的用户请求序列中选取 20%的用户请求序列，将其原有的请求序列替换为模拟的攻击序列，并和剩下的 25%的用户请求序列中的其它的正常请求序列一起构成测试集。

本文分别对随机 URL 攻击和重放 session 攻击进行了模拟，分别构造了 3 个含有随机 URL 攻击序列的测试集和 3 个含有重放 session 攻击序列的测试集，然后用本文的攻击检测系统分别对这两类测试集中的所有请求序列进行检测并计算检测率和误检率来检验本文的攻击检测系统对这两种类型的攻击的有效性。

含有随机 URL 攻击序列和含有重放 session 攻击序列的测试集片段分别如图 5.1 和图 5.2 所示，图中显示了训练集中请求序列的记录方式，在记录有用户 IP 或域名的行后跟若干依次记录用该用户请求序列的每个请求的信息的行，每个请求序列都有一个表明该序列实际是正常序列还是攻击序列的标记，该标记与用户的 IP 或域名处于同一行，两者之间以一个空格隔开，并且从图中可看出随机 URL 攻击序列和重放 session 攻击序列各自的特点。

```
194.130.28.41 normal
[06/08/1995:08:05:27 -0400] "/shuttle/countdown/liftoff.html"
[06/08/1995:08:05:34 -0400] "/shuttle/countdown/video/livevideo2.gif"
[06/08/1995:08:05:34 -0400] "/images/NASA-logosmall.gif"
nb1-du5.polarnet.fnsb.ak.us attack
[07/08/1995:11:07:31 +0800] "/shuttle/missions/sts-70/images/KSC-95EC-1058.gif"
[07/08/1995:11:07:32 +0800] "/news/sci.space.news/1511  "
[07/08/1995:11:07:33 +0800] "/cgi-bin/imagemap/countdown69?181,288"
```

图 5.1 含随机 URL 攻击序列的测试集片段

```
user14.snowhill.com normal
[06/08/1995:21:48:32 -0400] "/shuttle/missions/sts-71/movies/movies.html"
[06/08/1995:21:48:35 -0400] "/shuttle/missions/sts-71/sts-71-patch-small.gif"
[06/08/1995:21:50:29 -0400] "/shuttle/missions/sts-71/movies/sts-71-launch-3.mpg"
ix-stp-fl1-01.ix.netcom.com attack
[06/08/1995:12:10:41 +0800] "/ksc.html"
[06/08/1995:12:10:44 +0800] "/images/ksclogo-medium.gif"
[06/08/1995:12:10:53 +0800] "/images/NASA-logosmall.gif"
[06/08/1995:12:10:54 +0800] "/images/MOSAIC-logosmall.gif"
[06/08/1995:12:10:58 +0800] "/images/USA-logosmall.gif"
[06/08/1995:12:11:06 +0800] "/images/WORLD-logosmall.gif"
[06/08/1995:12:14:05 +0800] "/ksc.html"
[06/08/1995:12:14:08 +0800] "/images/ksclogo-medium.gif"
```

图 5.2 含重放 session 攻击的测试集片段

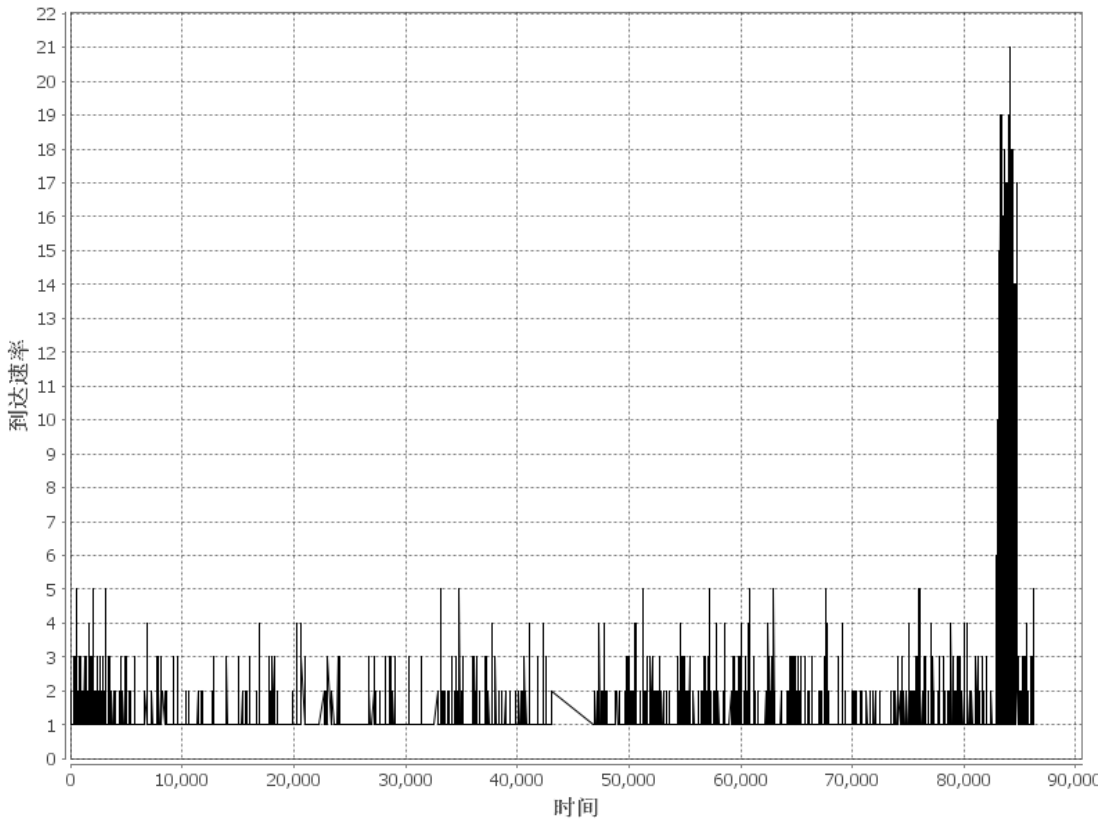

图 5.3 随机 URL 攻击测试集请求到达速率

在模拟随机 URL 攻击和重放 session 攻击产生攻击序列时，设定的攻击持续时间为 30 分钟。第一个随机 URL 攻击测试集和第一个重放 session 攻击测试集中，每秒钟到达服务器的请求的个数随时间变化的分布分别如图 5.3 和图 5.4 所示，其它测试集也具有类似的分布。从图中可看出在攻击持续时间内请求到达服务器的速率要

远大于其它时段内的请求到达服务器的速率，模拟的攻击序列已经构成了攻击。

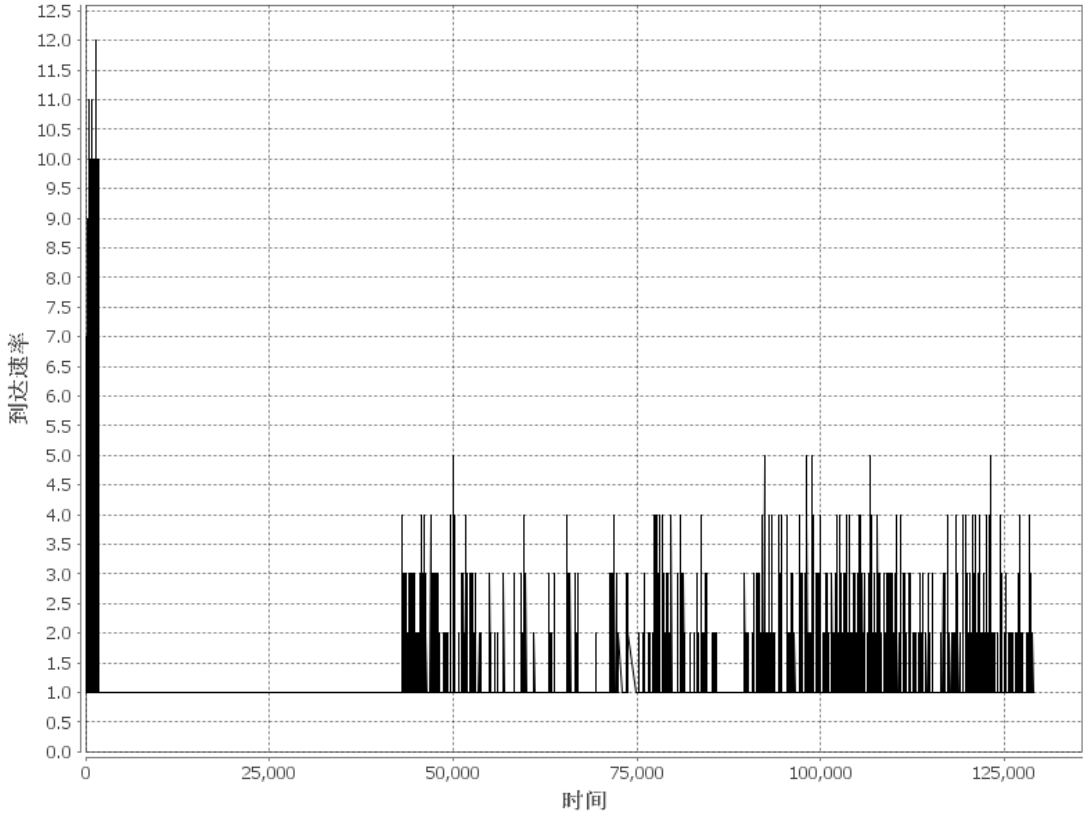

图 5.4 重放 session 攻击测试集请求到达速率

5.3 实验结果及分析

用攻击检测系统分别对含 3 个含有随机 URL 攻击序列的测试集和 3 个含有重放 session 攻击序列的测试集进行检测并计算检测率和误检率所得实验结果分别记录于表 5.1 和表 5.2 中。

表 5.1 随机 URL 攻击测试集的检测率和误检率

	测试集 1	测试集 2	测试集 3	平均
检测率	99.21%	99.37%	99.47%	99.35%
误检率	2.17%	2.94%	2.02%	2.38%

表 5.2 重放 session 攻击测试集的检测率和误检率

	测试集 1	测试集 2	测试集 3	平均
检测率	96.06%	96.84%	98.43%	97.11%
误检率	4.53%	2.47%	3.94%	3.65%

由以上实验结果可看出攻击检测系统在随机 URL 攻击测试集和重放 session 攻

击测试集上的检测率都大于 95%，并且误检率都小于 5%，说明文本所提出的基于条件随机场的检测方法是一种极为有效的方法。

攻击检测系统在两类测试集上的检测率和误检率具有一定的差异，在随机 URL 攻击测试集上的平均检测率比在重放 session 攻击测试集上的平均检测率高大约 2.24%，同时在随机 URL 攻击测试集上的平均误检率比在重放 session 攻击测试集上的平均误检率低大约 1.27%，说明攻击检测系统对随机 URL 攻击的检测效果要比对重放 session 攻击的检测效果好，这是因为重放 session 攻击的请求序列是截取正常用户的请求序列进行重放而构成的，其表现出来的行为与随机 URL 攻击的请求序列所表现出的行为相比更接近正常用户的访问行为，从而更难以被检测到，使得攻击检测系统的检测率降低，误检率升高。

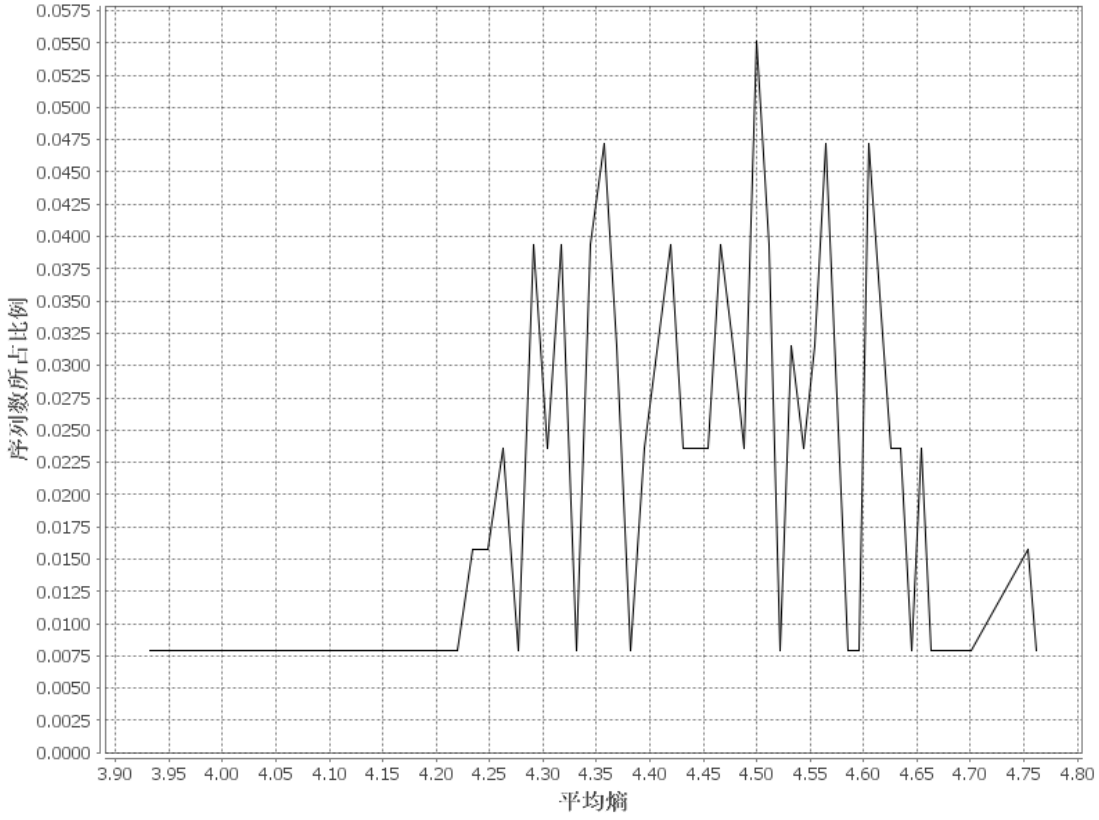

图 5.5 随机 URL 攻击的熵分布

本文的攻击检测方法是根据请求序列的平均熵来判别请求序列是攻击序列还是正常序列的，第一个随机 URL 攻击测试集和第一个重放 session 攻击测试集中的所有攻击序列的平均熵的分布分别如图 5.5 和图 5.6 所示，其它两个随机 URL 攻击测试集中的攻击序列的平均熵分布与第一个随机 URL 攻击测试集中的攻击序列的平均熵的分布类似，并且其它两个重放 session 攻击测试集中的攻击序列的平均熵分步

与第一个重放 session 攻击测试集中的攻击序列的平均熵分布也类似。从图中可看出重放 session 攻击序列的平均熵分布比随机 URL 攻击序列的平均熵分布更为集中，这是由这两种攻击序列的构造方式决定的，随机 URL 攻击序列的每个请求的资源相邻请求的之间的时间间隔都是随机选取的，因此其平均熵的分布较为分散，而重放 session 攻击通过重放同一个请求序列来构成攻击序列，因此其平均熵的分布较为集中。

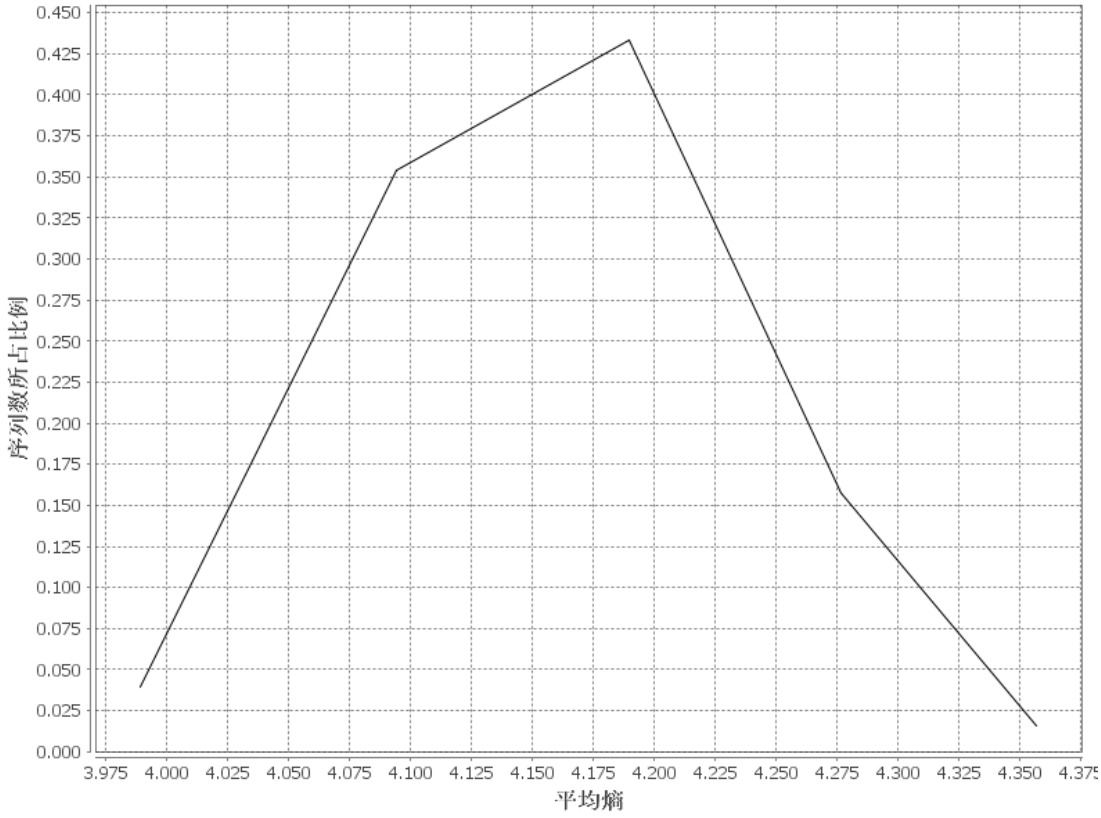

图 5.6 重放 session 攻击的熵分布

5.4 本章小结

本章首先提出了评价本文所提出的基于条件随机场的攻击检测方法有效性的标准，然后设计了一个实验来验证本文提出的攻击检测方法对随机 URL 攻击和重放 session 攻击的有效性，最后通过实验结果说明了本文提出的攻击检测方法是一种极为有效的方法，并对相关问题进行了分析。

6 总结与展望

6.1 研究总结

本文根据正常用户访问行为与被攻击者操控的僵尸主机向目标服务器发送的请求序列所表现出的访问行为的差异提出了一种对正常用户访问行为建模并根据请求序列所表现出的访问行为和正常用户访问行为模型的偏差来判断该请求序列是否是攻击序列的应用层 DDOS 攻击检测方法，在条件随机场相对于其它概率模型具有不需要在观察序列上做严格的独立假设和不存在标注偏置问题的优点的基础上提出用条件随机场来描述正常用户的访问行为，并定义了请求序列的平均熵来衡量请求序列所表现出的访问行为和正常用户访问行为模型的偏差。

本文根据所提出的检测方法实现了一个攻击检测系统，其大致工作流程是首先从正常用户访问日志中训练得到一个以条件随机场来描述的正常用户访问行为模型，然后根据此模型来计算待检测的请求序列的平均熵，最后根据计算所得的平均熵是否在正常范围内来判断该请求序列是正常序列还是攻击序列。

最后本文设计了一个性能评价系统来检验本文所提出的攻击检测方法对随机 URL 攻击和重放 session 攻击的有效性，检验结果表明本文提出的攻击检测方法能够极为准确地检测出攻击。

6.2 不足与展望

本文提出的基于条件随机场的应用层DDOS攻击检测方法经实验验证是一种极为有效地攻击检测方法，但是也存在一些不足之处，实验结果表明本文的检测方法对重放 session 攻击的检测效果没有对随机 URL 攻击的检测效果好，因此下一步的工作是改进现有的攻击检测方法，提高对重放 session 攻击这类更为隐蔽的攻击的检测能力。此外，检测出攻击之后如何对攻击序列做进一步的处理也是一个重要的研究课题。

致谢

在硕士研究生阶段，我得到了许多帮助、鼓励和关怀，在此，借这些许文字聊以寄抒我的感激之情。

首先要感谢我的导师谭运猛老师，感谢他在这两年半的时间里在学业上对我的悉心指导以及在生活中对我的关怀，他的远见卓识和深刻的洞察力让我少走了许多弯路，他不仅是我学业上的导师，也是我的人生导师，他以他的言传身教向我传授舍得之道。

感谢廖盛斌老师，感谢他将我带到一个新的世界，让我看到了全然不同的风景，感谢他在我遇到困难时为我解惑答疑，他就如灯塔一般为我在黑暗中前行指明了方向。

感谢马登邑同学和苏晓锋同学，在与他们的讨论中我对理论的理解更为深刻和清晰，还要感谢实验室的其他同学，感谢他们的鼓励，感谢他们的笑容，感谢他们使我的研究生生活更加多姿多彩。

感谢那些将服务器日志发布到网络上的组织，感谢他们为我提供了实验数据，在此对他们的开放精神致以崇高的敬意。

最后感谢我的家人，感谢他们默默看着我前行的温暖目光。

参考文献

[1] J. Mirkovic, P. Reiher. A taxonomy of ddos attacks and defense mechanisms. ACM SIGCOMM Computer Communications Review, 2004. 34(2): 39~54

[2] T. Peng, C. Leckie, K. Ramamohanarao. Survey of Network-based Defense Mechanisms Countering the Dos and DDoS Problems. ACM Computing Surveys (CSUR), 2007. 39(1): 3

[3] Xun Wang, Sriram Chellappan, Phillip Boyer et al. On the Effectiveness of Secure Overlay Forwarding Systems under Intelligent Distributed DoS Attacks. IEEE Transactions on Parallel and Distributed Systems, 2006. 17(7): 619~632

[4] D. Moore, G. Voelker, S. Savage. Inferring internet denial-of-service activity. ACM Transactions on Computer Systems (TOCS), 2006. 24(2): 115~139

[5] Rocky K. C. Chang. Defending Against Flooding-based Distributed Denial of service Attacks: A Tutorial. Communications Magazine, IEEE, 2002. 40(10): 42~51

[6] C. Douligeris, A. Mitmkotsa. Ddos Attacks and Defense Mechanisms: Classification and State-of-the-art. Computer Networks: The International Journal of Computer and Telecommunications Networking, 2004. 44(5): 643~666

[7] F. Kargl, J. Maier, M. Weber. Protecting web servers from distributed denial of service attacks. In: Proceedings of 10th International World Wide Web Conference. May 2001. 130~143

[8] Yu Chen, Kai Hwang, Yu-Kwong Kwok. Filtering of Shrew DDoS Attacks in Frequency Domain. In: Proceedings of the IEEE Conference on Local Computer Networks, 30th Anniversary. 2005. 786~793

[9] G. Carl, G. Kesidis, R. Brooks et al. Denial-of-Service attack detection techniques. IEEE Internet Computing, 2006. 10(1): 82~89

[10] Geer D. Malicious bots threaten network security. IEEE Computer, 2005. 38(1): 18~20

[11] 诸葛建伟, 韩心慧, 周勇林等. 僵尸网络研究. 软件学报, 2008. 19(3): 702~715

[12] LAU F., RUBIN S. H., SMITH M. H. et al. Distributed denial of service attacks. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. 2000. Vol. 3. 2275~2280

[13] Stephen M. Specht, Ruby B. Lee. Distributed Denial of Service: Taxonomies of

Attacks, Tools and Countermeasures. In: Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems. September 2004. 543~550

[14] L. Meyer, W. T. Penzhorn. Denial of service and distributed denial of service today and tomorrow. In: Proceedings of IEEE 7th AFRICON Conference. Vol.2, 2004. 959~964

[15] A. Hussain, J. Heidemann, C. Papadopoulos. A Framework for Classifying Denial of Service Attacks. In: proceedings of ACM SIGCOMM. Karlsruhe. August 2003. 99~110

[16] A. Keromytis, V. misra, D. Rubenstein. SOS: Secure overlay services. In: Proceedings of SIGCOMM. Pittsburgh, PA. Aug 2002. 61~72.

[17] Robert Stone. CenterTrack: An IP Overlay Network for Tracking DoS Floods. In: Proceedings of the 2000 USENIX Security Symposium. Denver, CO. July 2000. 199~212

[18] John Ioannidis, Steven M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In: Proceedings of Network and Distributed System Security Symposium. San Diego, CA. February 2002.

[19] Hal Burch, Bill Cheswick. Tracing Anonymous Packets to Their Approximate Source. In: Proceedings of the 2000 USENIX LISA Conference. New Orleans, LA. December 2000. 319~327

[20] Stefan Savage, David Wetherall, Anna Karlin et al. Practical network support for IP traceback. In: Proceedings of the ACM SIGCOMM Conference. Stockholm, Sweeden. August 2000. 295~306

[21] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, et al. Hash-based ip traceback. In: Proceedings of the ACM SIGCOMM. San Diego, CA. August 2001. 3~14

[22] R. Mahajan, S. Bellovin, S. Floyd et al. Controlling high bandwidth aggregates in the network. ACM Computer Communications Review, 2002. 32(3): 62~73

[23] Jelena Mirković, Gregory Prier, Peter Reiher. Attacking DDoS at the source. In: Proceedings of ICNP. Paris, France. 2002. 312~321.

[24] D. Whyte, E. Kranakis, P. Van Oorschot. ARP-Based Detection of Scanning Worms within an Enterprise Network. In: Proceedings of Annual Computer Security Applications Conference (ACSAC 2005). Tucson, AZ. Dec. 2005. 5~9

[25] M. Farahmand, A. Azarfar, A. Jafari et al. A Multivariate Adaptive Method for Detecting ARP Anomaly in Local Area Networks. In: Proceedings of International Conference on Systems and Networks Communication (ICSNC'06). Oct. 2006. 53~53

[26] L. R. Rabiner, B. H. Juang. An Introduction to Hidden Markov Models. IEEE ASSP MAGZINE, 1986. 3(1): 4~16

[27] Y. Yasami, M. Farahmand, V. Zargari. An ARP-based Anomaly Detection Algorithm Using Hidden Markov Model in Enterprise Networks. In: Proceedings of Second International Conference on Systems and Networks Communications. Cap Esterel. 2007. 69~75

[28] Denial P. Huttenlocher, Gregory A. Klanderman, William J. Ruchlidge. Comparing images using the hausdorff distance. IEEE Transaction on Pattern Analysis and Machine Intelligence, 1993. 15(9): 850~863

[29] Yann Labit, J. Mazel. HIDDeN: Hausdorff distance based Intrusion Detection approach dedicated to Networks. In: proceedings of The Third International Conference on Internet Monitoring and Protection (ICIMP 2008). Bucharest, Romania. 2008. 11~16

[30] J. Udhayan, R. Anitha. Demystifying and Rate Limiting ICMP hosted DoS/DDoS Flooding Attacks with Attack Productivity Analysis. In: Proceedings of IEEE International Advance Computing Conference (IACC 2009). Patiala, India. March 2009. 558~564

[31] Haining Wang, Danlu Zhang, Kang Shin. Detecting SYN flooding attacks. In: Proceedings of the IEEE Infocom. New York, NY. June 2002. 1530~1539

[32] V. A. Siris, F. Papagalou. Application of anomaly detection algorithms for detecting SYN flooding attacks. In: Proc. IEEE Global Telecommun. Conf. (IEEE GLOBECOM 2004). Dallas, TX. 2004. vol.4, 2050~2054

[33] Xu Rui, Ma Wenli, Zheng Wenling. Defending against UDP flooding by negative selection algorithm based on eigenvalue sets. In: Proc of IAS'09. Xi'an. Aug 2009. 342~345

[34] Takeshi Yatagai, Takamasa Isohara, Iwao Sasase. Detection of HTTP-GET Flood Attack Based on Analysis of Page Access Behavior. In: Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing.

2007. 232~235

[35] 肖军, 云晓春, 张永铮. 基于会话异常度模型的应用层分布式拒绝服务攻击过滤. 软件学报, 2010. 33(9): 1713~1724

[36] S. Ranjan, R. Karrer, E. Knightly. Wide area redirection of dynamic content by internet data centers. In: Proceedings of INFOCOM 2004. March 2004. Volume 2, 816~826

[37] J. Jung, B. Krishnamurthy, M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In: Proc. 11th IEEE Int. World Wide Web Conference. Honolulu, Hawaii. May 2002. 252~262

[38] W. G. Morein, A. Stavrou, C. Cook et al. Using graphic Turing Tests to counter automated DDoS attacks against web servers. In: proceedings of ACM CCS, 2003. Washington, DC. October, 2003. 8~19

[39] S. Kandula, D. Katabi, M. Jacob et al. Botz-4-Sale: Surviving Organized DDos Attacks That Mimic Flash Crowds. In: Proceedings of the USENIX Symposium on Network Systems Design and Implementation. Boston, MA. May 2005. Volume 2, 287~300

[40] D. Gavrilis, I. Chatzis, E. Dermatas. Flash crowd detection using decoy hyperlinks. In: proceedings of IEEE International Conference on Networking, Sensing and Control. London. April 2007. 466~470

[41] Michael Walfish, Mythili Vutukuru, Hari Balakrishnan et al. DDoS defense by offense. In: Proceedings of ACM SIGCOMM. Pisa, Italy. September 2006. 303~314

[42] S. Ranjan, R. Swaminathan, M. Uysal et al. DDoS-shield: DDoS-resilient scheduling to counter application layer attacks. IEEE/ACM Transactions on Networking, 2009. 17(1): 26~39

[43] Yi Xie, Shun-Zheng Yu. A Large-Scale hidden Semi-Markov Model for Anomaly Detection on User Browsing Behaviors. IEEE/ACM Transactions on Networking, 2009. 17(1): 54~65

[44] Y. Guédon. Estimating hidden semi-Markov chains from discrete sequences. Journal of Computational and Graphical Statistics, 2003. 12 (3): 604~639

[45] A. Ratnaparkhi. A Maximum Entropy Model for Part-of-Speech Tagging. In: Proc. EMNLP. New Brunswick, New Jersey. 1996.

[46] Adam L. Berger, Vincent J. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics, 1996. 22(1): 39~48

[47] S. Della Pietra, V. Della Pietra, J. Lafferty. Inducing features of random fields. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997. 19(4): 380~393

[48] Richard H. Byrd, Jorge Nocedal, Robert B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. Math. Program., 1994. 63(2): 129~156