江 蘇 大 學

JIANGSU UNIVERSITY

本 科 毕 业 论 文

基于半监督学习的 SDN 网络异常检测系统设计与实现

Design and implementation of SDN network anomaly detection system based on semi machine learning

学院名称：计算机科学与通信工程学院

专业班级：信息安全 1601 班

学生姓名：徐丙磊

指导教师姓名：李致远

指导教师职称：副教授

2020 年 6 月

基于半监督学习的 SDN 网络异常检测系统设计与实现

专业班级：信息安全 1601 班　学生姓名：徐丙磊

指导教师：李致远　　职称：副教授

摘要 使互联网用户免受网络攻击和其他威胁的侵害是当今网络运营商面临的最突出的安全挑战之一，其中分布式拒绝服务（DDoS）是最广泛的网络攻击之一，而且因为其灵活性、突发性及低成本等特点，想要有效的防御这种攻击非常困难，DDoS 攻击会导致系统因资源耗尽而停止工作。软件定义网络（SDN）是近来新出现的一种网络技术，提供了前所未有的可编程性，该网络架构允许网络运营商动态配置和管理其基础架构。但由于将控制平面单独作为一种设备独立出来，在面对 DDOS 这类泛洪攻击的时候，SDN 网络更加容易受到眼中的破坏，大量的 Packe_in 及 Packe_out 数据包将频繁传输于 OpenFlow 交换机与控制器，在数据量较大的时候将占用所有的队列资源，使得正常数据无法传输。

在本文中，我们提出了一种轻量级针对异常流量的检测系统，它利用机器学习方法（即决策树/随机森林）在基于 SDN 的 Internet 服务提供商（ISP）网络中防御 TCP-SYN 和 ICMP 泛洪攻击。与同类型异常检测系统相比，我们所设计的系统占用的系统资源更少，检测速度较快，准确率也较高。通过部署在虚拟机中的 SDN 环境中，我们实现了所设计的系统，并评估了该系统的准确性。通过广泛的实验，结果表明该系统可以在不影响良性流量的情况下，方便快捷地检测出我们所模拟的恶意流量，且平均准确率在 90%以上。

关键词 软件定义网络 机器学习 分布式拒绝服务攻击 泛洪攻击 流量检测 异常检测

Design and implementation of SDN network anomaly detection system based on semi supervised learning

Abstract Protecting Internet users from cyber attacks and other threats is one of the most prominent security challenges facing network operators today. Among them, distributed denial of service (DDoS) is one of the most extensive network attacks, and because of its flexibility, Sudden and low-cost characteristics, it is very difficult to effectively defend against such attacks, DDoS attacks will cause the system to stop working due to resource exhaustion. Software-defined networking (SDN) is a recently emerging network technology that provides unprecedented programmability. This network architecture allows network operators to dynamically configure and manage their infrastructure. However, because the control plane is independently used as a device, when faced with flooding attacks such as DDOS, the SDN network is more vulnerable to damage in the eyes. A large number of Packet_in and Packet_out packets will be frequently transmitted to OpenFlow switches and controllers , When the amount of data is large, it will occupy all the queue resources, so that normal data cannot be transmitted.

In this article, we propose a detection system for abnormal traffic, which uses machine learning methods (ie, decision trees / random forests) to perform TCP-SYN and ICMP flooding in SDN-based Internet service provider (ISP) networks attack. By deploying a test platform, we implemented the designed system and evaluated the accuracy of the system. Through extensive experiments, the results show that the system can detect malicious traffic with an accuracy of more than 90% without affecting the benign traffic.

Key words Software Defined Network Machine Learning Distributed Denial of Service Attacks Flood Attacks Traffic Detection Intrusion Detection System

目录

第一章 绪论

1.1项目研究背景及意义

近年来，随着互联网技术不断地更新迭代，云计算、5G 技术等新兴技术为人们带来便利的同时，也催生了大量的流量数据。各大应用服务器每天需要处理的流量数据高达数百 TB，这也对网络流量的管理监控带来了非常严峻的挑战！

流量的不断增加意味着服务器需要投入更多的资源来处理相应的数据，这也意味着对网络整体性能和安全性能的要求不断提高。因为这些日益增加的流量中很可能裹挟着黑客发起的恶意攻击流量！借以消耗服务器的内部资源达到攻占或瘫痪应用服务的目的。

如对网络安全威胁最大的 DDOS（distributeddenial of service attacks），即分布式拒绝服务攻击，正是恶意黑客利用 TCP 协议在设计上缺陷，利用三次握手的漏洞，通过发送大量的虚假 SYN 半链接数据包，恶意消耗服务的处理队列资源。使目标主机或相应服务器无法正常地为其他合法的用户提供服务。尤其是攻击者利用多台傀儡主机的情况下，向受害的主机发送的恶意攻击数据包数量将是极其巨大的，破坏力非常惊人。而且 DDoS 发起非常简单但危害性极大，难以被常规防御软件快速检测和有效防御。

SDN（Software Defined Network），也就是近年来刚兴起的软件定义网络，是一种控制集中式的新型网络技术，相较于传统的分布式网络架构而言，SDN 技术解耦了传统网络中的控制平面和数据平面，通过软件定义的方式可以更加快速地配置，管理和编程网络。在这种集中式网络管理架构中，底层网络的基础结构是从应用程序中抽象出来的。SDN 提供了跨数据中心，园区和广域网的自动化和可编程性选择。

而由于 SDN 网络的上述特点，一旦遭受 DDoS 攻击除了会危害被攻击主机、恶意消耗处理队列资源之外，还会引起 OpenFlow 交换机的流表项大量增长，数量众多的 Packet_in 数据包将会发送到控制器当中[1]，对整个 SDN 网络体系造成二次伤害。 在这个过程当中，控制器以及被攻击的主机，甚至与其所连接的 OpenFlow 交换机都会受到严重的影响。所以相较于传统网络，类似 DDoS 的网络攻击对于 SDN

架构的危害更大!

异常流量监控是一种利用流量收集、分析评估、检测预测等多种手段为网络的安全管理提供可靠参考、提高网络安全性能的技术。但针对新兴的 SDN 网络采取合适有效的流量监控方法能够有效地降低维护网络体系的投资成本和运行成本。建立一种快速、准确、有效的异常流量检测系统是 SDN 网络安全的重点和难点问题[3]。

1.2国内外研究现状

针对 SDN 的异常流量检测，国内外诸多学者都进行了相应的研究工作。最早是由 Braga 等人提出的利用 SDN 基于流的功能来检测 DDoS 攻击[5]。Gude 等则提出了一种基于定期从 NOX 控制器收集网络流量数据的 DDoS 攻击检测方法[6]。使用自组织映射分析网络流功能，然后将其分类为正常或恶意。使用的网络流量功能包括成对流量百分比，每个流量平均值的数据包、每个流量平均值的字节、每个流量平均值的持续时间，单流量增长以及不同端口的增长。但是在监控和处理大量数据时，特别是在集中式控制器环境中，控制器的性能会下降。

Mehdi 等人评估了各种传统异常检测方法在 SDN 上的实现，提出的框架包括四种异常检测算法，可检测家庭和办公室网络中的恶意流量[7]。该算法在 NOX 控制器上实现，包括带有基于信用的速率限制的 TRW-CB 算法，最大熵检测器 NETAD。但主要工作集中在基于家庭环境的网络流量上，没有讨论针对所检测到的网络异常的缓解策略。

C. Yu Hunag 等人提出了一种简单的方法来检测 DDoS 攻击。DDoS 防御程序通过监视 OpenFlow 交换机中的流数来检测 DDoS 攻击。一旦流量超过预定义的阈值，控制器就将其视为 DDoS 攻击，并插入流规则以丢弃数据包[8]。

在 Dotcenko 等人的工作中，设计实现了一种基于模糊逻辑的系统来识别参与恶意网络扫描的主机。结合了 TRW-CB 和限速算法以及模糊逻辑推理用来检测恶意流量[9]。该实验利用了一个 OpenFlow 交换机和四个主机的小型拓扑，在 Beacon 控制器和 Mininet 上进行测试的。

Piedrahita 等人在 2015 年提出了一种名为"FlowFence"的拥塞避免技术[10]，用于

缓解恶意流量的攻击。FlowFence 体系结构包括 POX 控制器和运行应用程序的 OpenFlow 路由器，这些应用程序用于监视接口的带宽消耗，平均带宽使用量用于检测拥塞状况。该实验是在未来互联网测试平台（FITS）上进行的，该拓扑具有在两台物理计算机、两个 OpenFlow 路由器以及四台虚拟主机。

上述所有的工作均使用本机 OpenFlow 统计信息。此方法需要访问流表中每个流条目的计数器。使用 OpenFlow 收集统计信息会极大的增加大型网络控制平面的开销。

T.V. Phan 等人使用了一种基于人工神经网络的方法在 OpenFlow 交换机中结合了分布式自组织映射系统[11]，以应对洪泛攻击。该系统还解决了控制平面的其他问题，例如性能瓶颈和控制器过载。在该系统中安全模块安装在 OpenFlow 交换机中，安全应用程序则部署在应用程序层中。

Han 等人在提出了一种名为"OverWatch"的具有协作智能的 DDoS 攻击防御框架，以保护特定网络中的主机和服务器。OverWatch 实现了一种轻量级的流量监视算法，该算法基于网络流量特征和不对称特征来检测 DDoS 攻击。控制平面利用基于机器学习的 DDoS 攻击分类器和僵尸网络跟踪算法来确定攻击类型和僵尸网络位置[12]。

为了解决意外的网络负载波动和分布式控制器环境中的可伸缩性的问题，Shang 等人提出了一种面向服务的负载平衡机制[13]。实现了一种负载均衡算法，该算法考虑了流量请求，当前负载和空闲控制器的传播延迟。当前负载超过预定值且流量请求次数超过预定平均值时，控制器将触发过载状态。然后将流量请求分配给空闲控制器或平均负载值较小的控制器。Zaalouk 等人提出了一种基于协调器的体系结构[14]，用于通过实现 SDN 控制功能来开发网络安全应用程序。OrchSec 提供了不同级别的监视功能，例如基于整体网络安全要求的不同级别。使用 OrchSec 框架可以抵御诸如 ARP 欺骗/高速缓存，DoS / DDoS 等网络攻击。

1.3 本章总结

虽然上述解决方案在应对部分异常流量攻击的时候具备较好的成效，但可以看出仍存在许多的缺陷：

（1）所设计系统的拓展性、迁移性较差，无法满足新业务部署需求；

（2）在一定程度上过度依赖硬件设备，且实验设备的使用成本较高；

（3）实时监控性较差，无法实现对整个 SDN 网络的实时监控管理，利用率、实用性低。

针对上述的问题，在本文中我们提出了一种新型的异常流量检测系统，利用机器学习方法（即决策树、随机森林），在基于 SDN 的虚拟网络拓扑中进行实验，通过训练所收集到的大量恶意 TCP-SYN 泛洪攻击流量数据，建立了相应的机器学习有效模型，并将其内嵌在图形化界面当中，以实现恶意流量的实时监控。

通过部署测试平台，我们实现了所设计的检测系统，且经过多次的广泛实验检验评估了系统的准确性，结果表明该系统可以在不影响良性流量的情况下实时监控恶意流量，准确率在 90%以上。

第二章 研究背景和相关工作

2.1 SDN

2.1.1 SDN 简介

在传统网络结构中，每个单个网络设备通常必须执行三个不同的活动，这些活动分别映射到网络的三个不同平面：数据，控制和管理。

数据平面负责处理传输流量，传输流量决定如何处理到达入口接口的数据包。该平面也被称为转发平面，因为它主要是指用于确定适当的出口接口的转发表。从数据包的角度来看，数据平面通常处理终端用户生成的数据包，这些数据包由网络设备转发到其他终端设备。

控制平面负责收集，处理和管理网络信息，以便确定转发行为。该平面通常包括各种表以及在这些表上工作的协议套件。因此控制平面处理用于网络自身或其他网络设备生成/接收的分组。在控制平面上通常会运行诸如路由接口状态管理、可达性信息交换等典型协议，以便为数据平面提供服务。

管理平面用于与设备进行交互或监控，以达到管理网络的目的。除了支持接口，网络（IP 子网）和控制平面协议的配置之外，管理平面还运行其自己的协议套件（例如 SNMP）。在传统的网络设备中，数据平面活动由专用硬件（或“高速代码”）执行，而控制平面操作则由设备 CPU 处理。

而作为一种新兴的网络技术，SDN 被认为是对上述传统分布式网络的重构，它利用开放软件替换硬件，使网络可编程，带来了极大的灵活性。不同于传统网络中各设备同时具备数据，控制和管理三个平面，现在在 SDN 网络中可以将其进行分离，将复杂的、消耗资源极大的控制逻辑从路由器，防火墙等网络元素中剥离出来，并形成逻辑上集中的控制器，以调节跨网络的数据包转发。这种架构消除了传统分布式网络体系结构的复杂和静态性质，可以拥有单独的逻辑控制平面，该逻辑控制平面可以与数据转发硬件（交换机）分开运行，并且可以在单独的服务器上运行。并且这种架构的另一个明显的好处是可以随时更改解耦的控制平面软件进而充分利用功能强大的计算机。

2.1.2 SDN 结构

在 SDN 网络中，可以分为三个不同的层：

• 应用层：包含着重于网络服务扩展的解决方案，这些解决方案主要是与控制器通信的软件应用程序。

•控制平面层：主要为一个逻辑集中的 SDN 控制器，全局负责维护当前的网络拓扑结构。它还可以通过 API 请求与应用程序层服务相连接，通过标准协议对网络设备进行统一的管理和监视。

• 数据平面层：主要涉及物理网络设备，包括以太网交换机和路由器。提供符合行业标准的可编程和高速硬件和软件。

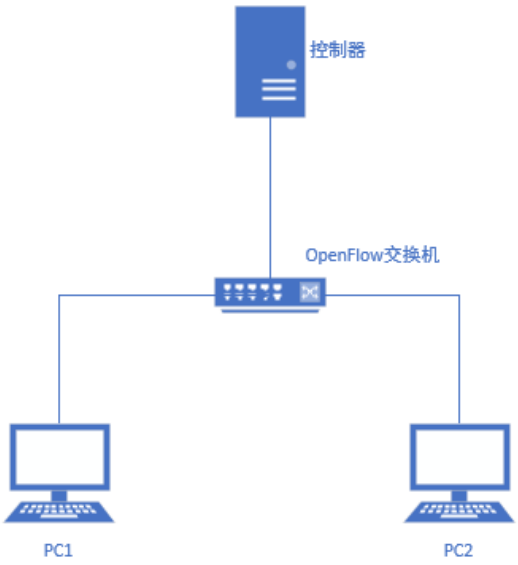

图 1 SDN 结构图

在底层的数据平面，物理网络由硬件转发设备组成，这些设备存储网络数据平面的转发信息库（FIB）状态（例如 TCAM 数目和已配置的端口速度），以及相关的元数据，包括数据包，流量和端口计数器。物理网络的设备可以被分组为一个或多个单独的控制器域，其中每个域具有至少一个物理控制器。OpenFlow 平面接口或基于标准的协议（通常称为“南向协议”， 例如 OpenFlow，PCEP，SNMP，OVSDB 等）定义了控制器平台与数据平面设备（例如物理和虚拟交换机以及路由器）之间的控

制通信。

控制平面层是 SDN 网络的核心，由每个域的控制器实现，该控制器收集分布在每个控制域中的物理网络状态。该组件也被称为“网络操作系统”（NOS），因为它使 SDN 能够以全局网络的形式向控制应用程序实例（在应用程序层中运行）呈现物理网络状态的抽象视图。

北向开放式 API 是指控制器的软件模块和 SDN 应用程序之间的软件接口。这些接口都可以在开源社区中查找，使用者可以利用这些 API 与 SDN 控制器进行交互。

应用程序层涵盖了一系列应用程序，可满足不同的客户需求，例如网络自动化，灵活性和可编程性等。SDN 应用程序的某些领域包括流量工程，网络虚拟化，网络监视和分析，网络服务发现，访问控制等。每个应用程序实例的控制逻辑可以作为单独的过程直接在每个域内的控制器硬件上运行。

2.2 机器学习

2.2.1 全监督学习

全监督学习是一种简单的机器学习模型，在这种学习模型中，需要给出输入和输出变量，并使用标记数据训练算法。

在全监督学习中，需要使用“标签 ”正确的数据训练机器。这意味着在进行训练之前需要人工对其进行标记。虽然在一定程度上增加了工作量，但全监督的优点是其准确度较高，计算复杂度较低。常见的全监督学习包括支持向量机，神经网络，K-近邻，随机森林和决策树。

所有的全监督学习算法在本质上都是比较复杂的算法，而从类型上来看主要分为分类或回归。

分类被用于预测给定的输入值的结果，根据训练结果预测输出的分类当中。而且分类模型大多数情况下适用于非实数值形式的结果。例如通过训练输入数据尝试预测“患病”或“健康”等分类标签。

但是当输出变量为实数值形式时，大多数情况下则使用回归预测给定样本的结果。例如，回归模型用于处理输入数据以预测降雨量，人体身高身高等情景。

线性回归，逻辑回归，CART，朴素贝叶斯和 K 最近邻（KNN）是全监督学习的典型算法。

组合是另一种监督学习形式，这种形式意味着通过组合多个各自较弱的机器学习模型的预测，以对新样本产生更准确的预测。

2.2.2 无监督学习

无监督学习是另一种典型的机器学习形式，这种形式指的是在进行机器学习的过程中，所使用的训练数据集没有进行分类，而且大多数情况下是没有或者无法人工添加标，需要机器自行学习。因此无监督学习的主要特征就是机器能够自行对数据进行学习分类，而无需提前对数据进行处理添加标签。

无监督学习的主要通过给机器提供大量的变化数据，并允许其从数据中学习，以提供以前未知的见解并识别隐藏的模式。因此，不一定要定义无监督学习算法的结果。而是它通过给定的数据集所进行学习的过程。

例如 Apriori，K-means，PCA 等算法是典型的无监督学习算法。

2.2.3 半监督学习

任何全监督学习算法最大的缺点是，数据集必须由机器学习工程师手工标记，这种方法准确度虽然较高但这个过程成本较高，尤其是在处理大量数据的时候。而任何无监督学习的最大的缺点是其应用范围有限。

为了克服上述这些缺点，引入了半监督学习的概念。在这种类型的机器学习模型中，同时根据标记和未标记数据的组合来训练算法。通常此组合将包含非常少量的标记数据和大量未标记的数据，这种算法的基本过程是程序员首先将使用无监督学习算法对相似数据进行聚类,然后使用现有的标记数据来标记其余未标记的数据。这种算法的典型用例在它们之间具有一个共同的属性——未标记数据的获取相对便宜，而标记所述数据则非常昂贵。

2.2.4 经典算法

1）KNN

与其他的将数据集分为训练集和测试集的监督学习算法不同，K 最近邻算法通常不划分测试集，而是将整个数据集作为训练集进行机器学习训练。

当新数据实例需要结果时，KNN 算法遍历整个数据集以找到新实例的 k 个最近实例，或与新记录最相似的 k 个实例，然后输出均值结果（针对回归问题）或模式（最常见的课堂）的分类问题。k 的具体数值是由用户指定的。

K 最近邻算法的应用场景较多，例如欧几里得距离和汉明距离之类的度量来计算实例之间的相似性。

2）Logistic 回归

线性回归预测是连续值（即，以厘米为单位的降雨），逻辑回归预测是在应用转换函数后的离散值（即，学生是否通过/未通过）。

Logistic 回归最适合于二进制分类：y＝0 或 1 的数据集，其中 1 表示默认类别。例如，在预测事件是否会发生时，只有两种可能性：事件发生（我们将其表示为 1）或事件不发生（0）。因此，如果我们预测病人是否生病，我们将使用 1 数据集中的值标记患病的病人。

逻辑回归以其使用的转换函数命名，该函数称为逻辑函数 h（x）＝1 /（1＋ex）。这形成了 S 形曲线。

在逻辑回归中，输出采用默认类别的概率形式（与线性回归不同，线性回归是直接产生输出的）。由于这是一个概率，因此输出在 0-1 的范围内。因此，例如我们要预测患者是否生病，我们已经知道生病的患者用表示 1，因此，如果我们的算法将 0.98 的得分分配给患者，则认为患者很有可能生病了。

使用逻辑函数 h（x）= 1 /（1 + e ^ -x）通过对 x 值进行对数转换来生成此输出（y 值）。然后应用阈值以强制将此概率转换为二进制分类。

3）分类树和回归树

分类树和回归树有时称为 CART，是决策树的一种简单形式，其中建模树是二进制的，仅使用算法和数据结构。

该树上有两种类型的节点：

分支节点，代表单个输入变量，并在变量上提供单个分割点（假设其数值）。叶子节点，代表两个输出变量。

分类树和回归树的终端节点是叶节点，而根节点和内部节点则是非终端节点。每个非终端节点通常代表一个输入变量（x）和该变量的分割点，叶节点表示输出变量（y）。该模型按如下方式进行预测：遍历树的拆分以到达叶节点并输出在叶节点处存在的值。

当机器运行算法时，通过跟随分支节点拆分直到到达叶节点（即预测或类值输出）来进行预测。

分类树和回归树易于学习和使用，并且对于所有问题都非常准确。由于不需要特别准备数据，因此这些方法的实施速度特别快。

2.2.5 算法选择

本系统整体结构较为简单，主要功能为识别正常/异常流量，且在训练数据集的收集过程中使用的是模拟流量生成工具 Hping3，对恶意流量的标记相对较为简单。因此在进行相关实验后，我们从 KNN、Logistic 回归、决策树三种经典算法中选择使用决策树作为机器学习所用算法，对所收集的数据流量进行二分类。

2.3 入侵检测

2.3.1 入侵检测系统类型

常见的入侵检测系统类别众多，按照不同的类型大致可以分为以下几种：

1.按照未知攻击/已知攻击分类 ：

基于签名的入侵检测：这类的检测系统主要针对已知的攻击，通过实时流量收集模块，将传入的流量与已知的攻击模式（称为签名）进行比较，通常需要建立专

门的已知攻击特征数据库。这种类型的检测系统一般准确度较高，但是很难检测到新的攻击类型，对于新攻击的检测准确率很低。且这种系统一般由专门的供应商负责发布更新攻击数据库（类似于防病毒软件）

基于异常的入侵检测：这类的检测系统主要针对未知的攻击，由于我们无法预测各种未知攻击的行为特征,所以也无法提前设置相应异常特征数据库。但相对的，我们可以使用机器学习来创建一个模拟常规活动的模型，该模型中的行为视为正常流量数据，然后将实时捕捉到的数据流量行为与现有正常模型进行比较，若无法分类到正常流量模型中则视为异常流量。但这种模型需要提前收集大量的正常数据行为特征，否则误报率较高。

2.按照检测系统行为分类：

主动：也被称为入侵检测和防御系统。该类系统发现入侵攻击会生成警报、日志条目以及用于更改配置以保护网络的命令，能够自动防御恶意行为。

被动：被动的检测系统仅检测恶意活动并生成警报或日志，但不采取防御措施，一般需要人工干预进行防御，但系统较为简单，不会占用大量的服务器资源。

3.通过攻击检测发现的位置分类：

网络入侵检测（NIDS）：这种类型的检测系统属于全局策略性的系统，由一个或多个组件组成，主要用于监视所有进出所监控网络的流量。

主机入侵检测（HIDS）：这种类型的检测系统运行在网络中与组织的 Internet / Intranet 连接的所有设备上。他们可以检测到源自网络拓扑内部的恶意流量（例如当恶意软件试图从组织中的主机传播到其他系统时）

2.3.2 入侵检测类型选择

由于本课题是基于仿真 SDN 环境进行设计实现，鉴于在仿真环境中的局限性，因此本系统主要用于被动地检测已知攻击，用模拟流量工具 Hping3 由系统内部生成 DDOS 攻击为例，演示本系统对于恶意流量的检测效果，并将相应的检测结果及预测准确率实时输出到图形化界面当中。

2.4 本章总结

在本章 2.1 及 2.2 节当中我们主要阐述了本系统所涉及的 SDN、机器学习相关原理，介绍了全监督、无监督、半监督这三种常见机器学习模型的优缺点，及机器学习中经典的 KNN、Logistic 回归、决策树算法，并在综合比较之后选取决策树作为本系统的实现算法。在 2.3 节当中，我们介绍了入侵检测系统的常见类型，并针对课题研究实现的实际环境进行了检测类型的选择，在下一章当中我们将进行详细的介绍实现。

第三章 基于半监督学习的 SDN 异常检测系统的方案实现

3.1 流量收集

3.1.1 Tshark 简介

TShark 是 Wireshark 的面向终端的版本，旨在在不需要或不提供交互式用户界面时捕获和显示数据包，因此需要调用终端输入相应的命令及参数运行，它所支持功能选项与 wireshark 相同，甚至在一定程度上比 wireshark 更加的便捷、强大！

除了能够实时抓包外，它还支持从本地直接读取 pcap 文件，进行相应的分析，并可以通过设置筛选特征值，将需要的数据内容打印在终端中，当然也可以直接输出成文件，支持 txt、csv、json 等多种文件格式。与其他协议分析器相比，在 TShark 中可过滤的字段更多，并且可用于创建过滤器的语法更加丰富。

它的常见命令较多，诸如 tshark [-i <捕获接口> |-] [-f <捕获过滤器>] [-2] [-r <输入文件>] [-w <输出文件> |-] [选项] [<过滤器>]等等。

可以使用-f 或-R 选项分别指定捕获或读取过滤器，在这种情况下，必须将整个过滤器表达式指定为单个参数，或者可以在选项参数之后用命令行指定参数，在这种情况下，过滤器参数之后的所有参数都被视为过滤器表达式。

3.1.2 Tshark 参数

-c <捕获数据包数>

该命令主要用于设置捕获实时数据流量时要读取的最大包数，如果是用于读取捕获文件，该命令也可以设置读取的最大数据包数，如-c 30。

-e <字段>

如果选择了-T 命令，即需要将数据包输出成相应的文件时，则可以通过-e 命令将一个字段添加到要显示的字段列表中，当然该命令可以在命令行上多次使用。但是只要选择了-T 字段选项，就必须至少提供一个字段。

-E <字段打印选项>

该命令同为辅助参数命令，即选择-T 字段时，可以通过-E 来控制字段的打印。

常见选项有：

header＝y | n，可以理解为打印表头，如果为 y，则将打印设置的-e 字段为表头。

separator= / t | / s | <字符>，主要用于设置字段分隔符。如/s 即为选择空格作为分隔符，若需要自定义分隔符，可以自行选择命令行能够接受的符号。

quote＝d | s | n 设置引号字符以用于包围字段。d 使用双引号，s 单引号，n 不使用引号（默认值）。

-i | --interface <捕获接口> |

该命令主要用于设置捕获数据包的网络接口名称，具体的接口可以通过-D 命令显示输出。

如果未指定接口，则 TShark 将自动搜索接口列表，并进行相应的选择，一般会首先第一个非环回地址。

该命令可以多次出现。从多个接口捕获时，捕获文件将以 pcap 格式保存。

-r |-读取文件<infile>

通过该命令可以从本地直接读取数据包数据，支持格式较多，可以是任何通用的捕获文件格式（包括压缩文件），如 pcap。

-T ek | fields | json |

该命令可以与-e -i 等多个命令配合使用，主要用于将实时捕获或从本地读取的 pcap 文件转换成其他格式，如 json 或 csv 文件等。

3.1.3 实时抓包实现

通过上述对 Tshark 工具的介绍，我们可以通过调用终端在后台实现对流量数据包的实时捕获。

在经过大量实验测试之后，我们发现在正常情况下每捕获十个数据包便进行分析效果最佳，因此在本模块中我们便将实时捕捉数设置为 10，并通过以下命令实现：

tshark -i any -c 10 -w 1.pcap

但由于我们的 SDN 仿真环境是基于 Ubantu 系统所搭建的，其中涉及到权限问题，在 root 用户权限下捕捉到的数据包普通用户无法调用，因此需要将存到本地的数据包释放权限，具体命令为：

chmod 666 1.pcap

至此实时捕捉数据包的环节已经完成，但存在本地的 pcap 数据包无法直接用于机器学习，因此需要将其转换成 csv 文件，以便机器学习模块读取，我们通过以下命令实现。

```
tshark -r 1.pcap -T fields -e frame.time_delta -e tcp.len -e icmp -e tcp.analysis.flags -e openflow_v4.type -e openflow_v4.length -E header=y -E separator=, -E quote=d -E occurrence=f > 2.csv
```

通过多种命令参数的组合，我们设置筛选了数个用于机器学习的字段。

在后续的图形化界面设计当中，我们同时增加了实时将捕获到的数据包打印输出到图形化界面当中，但与用于机器学习的字段不同，我们用于显示的字段更偏于显示我们所关心的源地址、端口号等具体信息，因此需要单独通过命令再次输出 pcap 文件，我们可以通过以下命令实现：

```
tshark -r 1.pcap -T fields -e frame.time -e ip.src -e tcp.srcport -e tcp.dstport -e openflow_v4.type -E header=y -E separator=, -E quote=d -E occurrence=f > test.csv
```

同样由于操作系统环境问题，上述命令需要通过 os.system 命令来调用 Ubantu 终端实现。

至此涉及到 Tshark 工具的模块已经全部介绍完毕。

3.2 机器学习训练模型

3.2.1 采集数据集

由于本系统是基于仿真 SDN 环境，所用操作系统为 Ubantu，因此为了方便后续机器学习所建模型的准确性，我们使用了模拟流量生成工具 Hping3 在所建立的

SDN 拓扑内部发起 TCP-SYN 泛洪攻击，并通过命令设置随即生成攻击源地址，以便对真实攻击源地址进行伪装。

Hping3 是面向命令行的 TCP / IP 数据包汇编器/分析器，支持 TCP，UDP，ICMP 和 RAW-IP 协议，而且具有跟踪路由模式，在覆盖通道之间发送文件的功能以及许多其他功能。

Hping3 的功能有很多，包括但不限于进行防火墙测试、网络测试等等，在本文中我们则使用该攻击模拟恶意黑客发起的恶意攻击，具体步骤如下：

1)运行所建立的 SDN 网络拓扑，并对虚拟主机、OpenFlow 交换机进行相应的配置，OpenFlow 协议选择 1.3 版本。

2)通过命令链接远程 Ryu 控制器，链接成功后可以看到在终端界面中有数据包的输出。

3)运行 Wireshark 工具，监听 any 端口，以便于收集途径整个网络的数据包。

4)启动 Hping3 工具，通过设置相应的参数，我们设定攻击模型为 TCP-SYN 泛洪攻击，发送频率为每秒 10 个半链接数据包，并通过--rand-source 命令随机伪造攻击源地址。

在发送一定攻击数据包之后，我们关闭了 Hping3 工具，并继续收集了一部分正常流量，训练数据集最终收集了 2000 条流量数据，用来进行机器学习的训练。

3.2.2 处理数据集

经过 3.2.1 节我们所收集到的流量以 pcap 数据包的形式存在本地，但 pcap 类型的数据包我们无法直接用于机器学习，所以需要对其进行一定的处理，详情可见 3.1.3 中我们的处理过程。

通过 Tshark 工具我们从原始流量数据中筛选了数个特征值，包括 frame.time_delta,tcp.len,icmp,tcp.analysis.flags,openflow_v4.type,openflow_v4.length 等特征。

通过命令将其转换为 csv 文件之后，我们需要人工对其打上标签，因为我们在模拟攻击流量的时候，已经设置过随机伪造源地址，所以能够很轻松的筛选出攻击

流量，当然也鉴于该种原因我们并没有选择源 IP 地址作为训练特征值。

我们在 csv 文件中额外另加了一列，列名为 an，所有的攻击流量该列值设定为 1，正常流量标记为 0。

3.2.3 训练数据集

Scikit-learn 是 Python 中提高一系列机器学习算法的接口库，以其快捷方便的特性广受欢迎。但其需要建立在 SciPy（科学 Python 库）的基础上，因为机器学习设计到很多的数学运算公式，例如矩阵、线性表等。

Scikit-learn 安装较为简单，可以直接通过 pip 命令下载，但同时需要一起下载 numpy 和 pandas 库，以便读取文件和进行线性计算。

首先通过 pd.read_csv("ex.csv")命令从本地直接读取 csv 文件，并将其存入 data 列表当中，随后将 frame.time_delta,tcp.len,icmp,tcp.analysis.flags,openflow_v4.type,openflow_v4.length 等特征划分到 X 中，将 an 划分到 Y 中。

但现在的特征数据当中含有空缺值，需要使用 x['tcp.len'].fillna(0, inplace=True)命令对缺失值进行处理，将空白值填充为 0。

在进行训练的时候，为了方便检验训练效果，我们将数据集划分为训练集和测试集，其中测试集占 25%。

但没经过特征工程处理的数据集会影响训练效果，因此我们可以通过 fit_transform 命令将其进行标准化、归一化处理。然后便可以直接用 sklearn 内嵌的决策树算法对数据进行训练。

3.2.4 异常流量预测

通过 3.2.3 节的模型训练，我们已经得到了一个较为可靠的机器学习模型。

为了检测模型的准确率，我们又额外测试了五个数据集，在这五个数据集中攻击流量占比分别为 80%、60%、50%、40%以及 20%，用模型进行预测后分别得到了准确率、精准率、召回率及 F1-Score 等四个评估机器学习性能的数据，具体数据如下图所示：

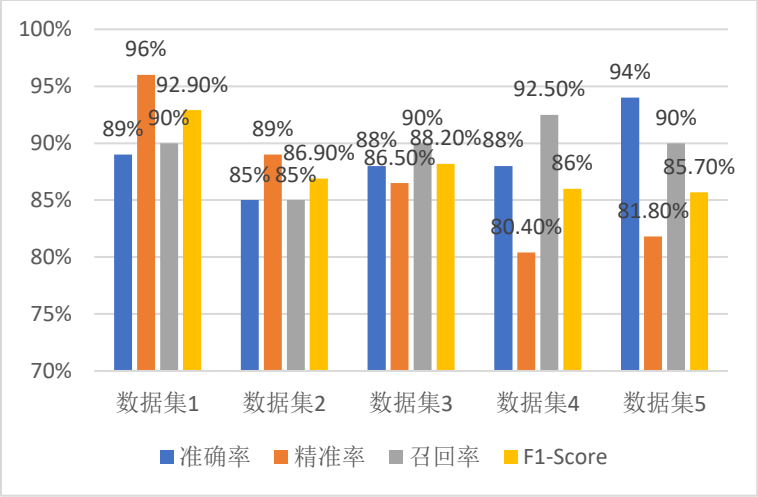

图 2 系统性能测试

从上图中我们可以看到系统预测准确率最低为 85%，最高为 94%，平均准确率为 88.8%；精准率最低为 80.4%，最高为 96%，平均精准率为 86.74%；召回率最低为 85%，最高为 92.5%，平均召回率为 89.5%；F1-Score 最低为 85.7%，最高为 92.9%，平均 F1-Score 为 87.94%。

由上述数据可以看出，本系统在攻击流量占比较高的情况下，性能水平最高。

在系统的实际应用当中，我们需要从本地直接读取经过 Tshark 处理筛选过的 csv 文件，随即将数据按照补缺、标准化归一化处理之后，输入到模型当中，然后将计算的攻击/正常状态输出到图形化界面当中。

3.3 图形化界面设计

3.3.1 界面设计

界面的设计我们采用的是 PyQt5-designer，可以较为直观的直接进行设计，系统的主界面如下图所示：

图 3 图形化界面

其中最上方为实时输出到界面中的数据包表头，包括时间、源 IP 地址、源端口号、目的端口号及 OpenFlow 类型，中间的文本框用于显示数据。

点击左下方的“开始检测”按钮之后。系统便开始运行，在经过抓包、解析、预测等多个步骤之后，会根据抓到的数据包预测当前的网络状态，包括“受到攻击”和“网络正常”两种状态。而准确率则是为了防止错误预警专门设置的，可以按照可信度来理解，在训练过程中，如果 10 个数据包中 9 个被系统预测为攻击流量，则输出的准确率则会显示为 90%。

右下方的分析间隔则是为了防止抓包模块和分析模块同时读取同一个文件产生死锁，预先设定的窗口值，默认为 10 秒，使用者也可以手动修改。

3.3.2 界面实现

在 PyQt5-designer 中初步设计好界面后，我们存储在本地的是 UI 文件，需要用命令将其转换为 py 文件，然后我们就可以将抓包模块和分析模块增添到相应的位置，并与显示窗口绑定。

其中为了能够达到实时显示抓包文件，我们在 3.1.3 节中提到过，专门用 Tshark 进行过处理，生成了一个名为 test.csv 文件，通过读取该文件，将其中的数据按照列表数组的形式存储起来。然后使用 append 命令追加显示到文本框中。

3.4 本章总结

在本章中，我们按照系统实现的不同阶段，依次介绍了设计实现流量收集、机器学习、界面设计等三个主要的步骤。其中在流量收集模块中，我们所使用的主要工具为 Wshark，通过 Wshark 我们可以将实时捕捉的数据包进行筛选转换，生成用于机器学习的 csv 文件。然后我们便可以在机器学习模块中利用收集到的数据集进行训练或预测，相应的运算结果则会通过接口实时输出的图形化界面当中。至此，我们的系统已经设计完毕，可以进行测试使用了。

第四章 实验评估

4.1 SDN 环境搭建

4.1.1 mininet 搭建

Mininet 是一个轻量级仿真 SDN 测试平台，可以通过相对轻量级的技术将网络拓扑结构进行虚拟化，可以非常轻松地创建一个 SDN 拓扑结构，而且可以直接进入建立的 host 中执行终端命令，发送数据包等等，完全可以与实际物理机媲美。

因为在 GitHub 上有 Mininet 完整的安装源码，因此只要我们的虚拟机配置安装完 git 后便可以直接从 Github 上克隆源码到本地。

在 Ubantu 系统中 Mininet 安装命令有多种，如下图所示：

```
# install.sh -a      ##完整安装（默认安装在home目录下）
# install.sh -s mydir -a        ##完整安装（安装在其他目录）
# install.sh -nfv       ##安装Mininet+用户交换机+OVS（安装在home目录下）
# install.sh -s mydir -nfv          ##安装Mininet+用户交换机+OVS（安装在其他目录下）
```

图 4 Mininet 安装命令

这里我们采用第三种，直接在终端中输入命令后便开始自动安装，安装完成界面如下图所示：

```
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/home/sdn/oflops/example_modules/snmp_cpu'
make[2]: Leaving directory '/home/sdn/oflops/example_modules/snmp_cpu'
make[2]: Entering directory '/home/sdn/oflops/example_modules'
make[3]: Entering directory '/home/sdn/oflops/example_modules'
make[3]: Nothing to be done for 'install-exec-am'.
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/home/sdn/oflops/example_modules'
make[2]: Leaving directory '/home/sdn/oflops/example_modules'
make[1]: Leaving directory '/home/sdn/oflops/example_modules'
Making install in cbench
make[1]: Entering directory '/home/sdn/oflops/cbench'
make[2]: Entering directory '/home/sdn/oflops/cbench'
 /bin/mkdir -p '/usr/local/bin'
  /bin/bash ../libtool   --mode=install /usr/bin/install -c cbench '/usr/local/bin'
libtool: install: /usr/bin/install -c cbench /usr/local/bin/cbench
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/sdn/oflops/cbench'
make[1]: Leaving directory '/home/sdn/oflops/cbench'
Making install in doc
make[1]: Entering directory '/home/sdn/oflops/doc'
make[1]: Nothing to be done for 'install'.
make[1]: Leaving directory '/home/sdn/oflops/doc'
Enjoy Mininet!
```

图 5 Mininet 安装成功

为了再次确认我们已经成功安装了 Mininet，我们可以输入一些简单的命令来进

行测试：

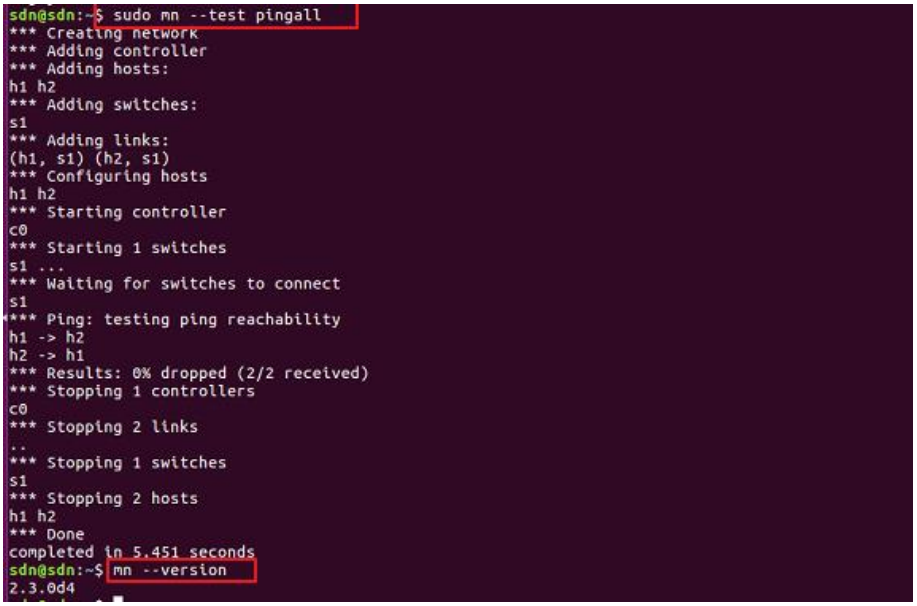

图 6 测试 Mininet 是否安装成功

至此我们的 Mininet 已经安装完毕，可以进入下一步的安装过程。

4.1.2 ryu 搭建

Ryu 是一个用 Python 编写的 OpenFlow 控制器，其中包含了大量的 API，开发人员可以根据项目需要直接调取相应的功能以便方便快捷的进行网络设备的部署及功能实现。

Ryu 支持的协议众多，例如 OpenFlow 协议 Ryu 完全支持多个版本，像 1.0,1.1,1.3 等多个版本均在支持范围内。

Ryu 可以使用 pip（pip install ryu）或直接从 GitHub 源码进行安装，但是在安装的同时会导致在安装时从网络上下载大量 Python 依赖项。

Ryu 的更新频率较高，大约每个月都有一个新版本，并且依赖项的数量（以及这些依赖项的最低版本）逐渐增加。

在运行安装命令之后，我们同样可以通过一些简单的命令来测试是够安装成功：

```
@ubuntu:~/opt/ryu/ryu/app$ ryu-manager simple_switch.py
oading app simple_switch.py
oading app ryu.controller.ofp_handler
nstantiating app simple_switch.py of SimpleSwitch
nstantiating app ryu.controller.ofp_handler of OFPHandler
```

图 7 安装 Ryu

4.1.3 实验拓扑结构

由于 Mininet 的便捷性，我们可以直接通过调用 MiniEdit.py 通过图形化界面直接进行网络拓扑的设计，如下图所示：

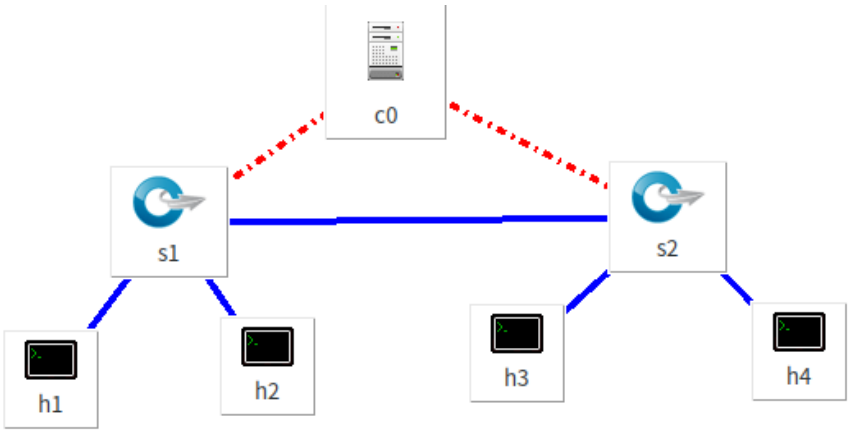

图 8 SDN 拓扑结构

为了避免占用虚拟机的内存资源，我们所搭建的网络拓扑较为简单，其中包含一个控制器，两个 OpenFlow 交换机以及三个 Host 用户主机。

其中 h1 的 IP 地址为 192.168.0.1，h2 的 IP 地址为 192.168.0.2。h3 的 IP 地址为 192.168.1.1，h4 的 IP 地址为 192.168.1.2。

由于 OpenFlow 的 1.0 和 1.3 版本有一定的差异，因此我们直接在 edit 中指定了该拓扑结构所使用的 OpenFlow 协议版本为 1.3。

且由于我们安装了 Ryu 控制器，因此需要在 Mininet 中指定控制器为远程控制器，在运行拓扑之后我们可以通过命令接入到 Ryu 控制器，随即在终端中便可以看到部分流表的刷新，如下图所示：

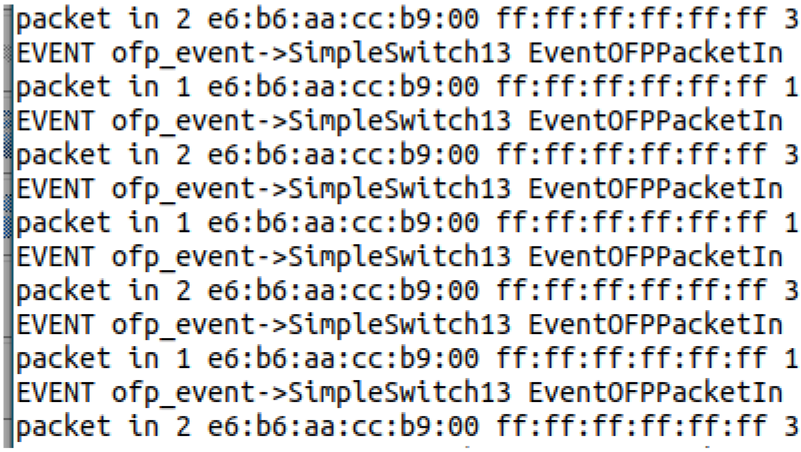

图 9 Ryu 流表刷新

4.2 攻击检测

4.2.1 攻击流量检测

首先我们需要将我们的系统运行在 h1 中，如下图所示：

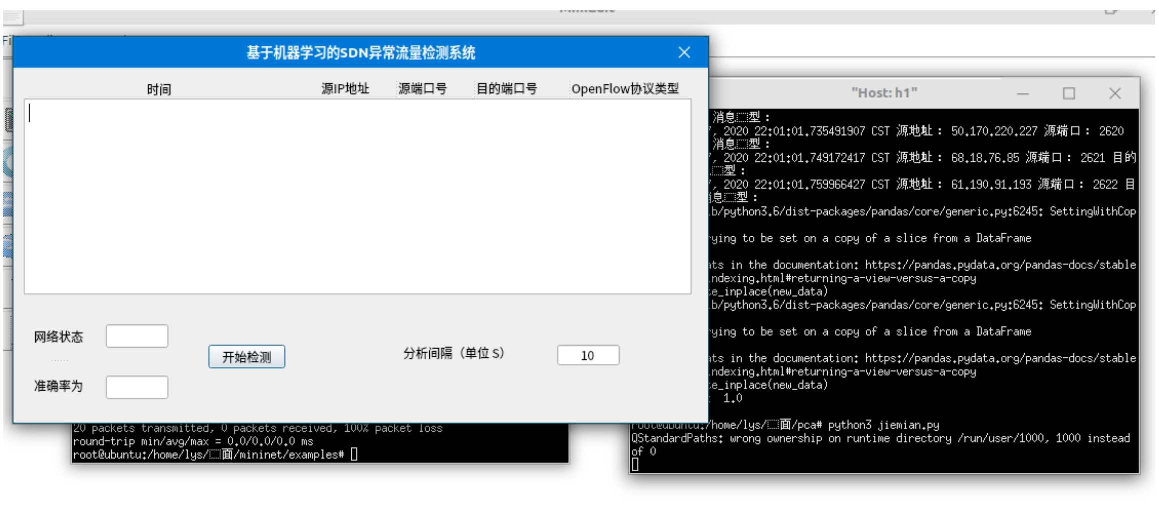

图 10 系统运行界面

图形化界面的具体解析可以参考 3.3.1 节，这里我们就不做过多解释。

在点击开始检测按钮后，系统将开始运行，同时在终端界面中我们可以看到抓包模块已经开始监听：

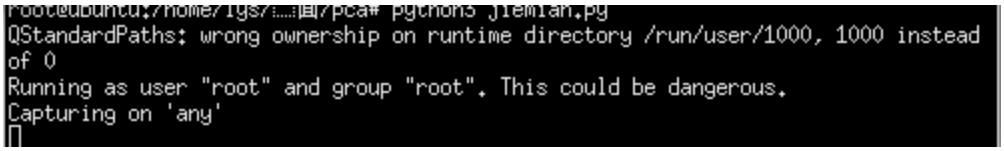

图 11 系统开始抓包

我们此时打开 h2 的终端端口，运行 Hping3 工具，指定向 192.168.0.1 发起 TCP-SYN 泛洪攻击，且随机生成攻击源地址，为了避免短时间内虚拟机内存被消耗完毕导致系统崩溃，我们指定输出 20 个数据包将自动结束运行，如下图所示：

为了检测攻击数据包是否成功发送，我们同时开启了 Wireshark 进行监听，可以看到成功的捕捉到了我们所发出的攻击数据包：

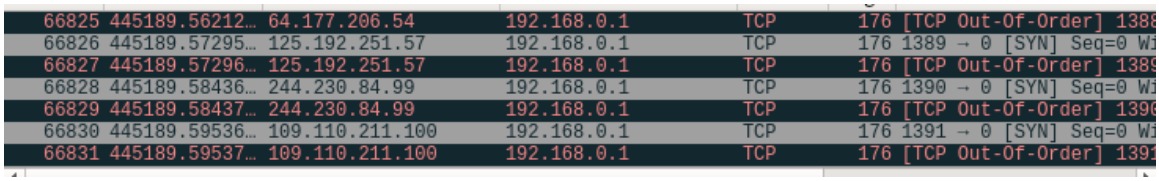

图 12 Wireshark 监控数据

在经过一定的分析处理之后，我们的检测系统中直接输出了对应的检测结果及准确率，如下图所示：

图 13 系统运行结果 1

我们可以看到效果比较明显，系统成功检测到了我们所发出的攻击流量，且及时给出了“收到攻击”的报警信息，下方的准确率也直接显示为 1.0（100%）!

4.2.2 正常流量检测

由于我们的 SDN 网络是在本地环境中运行的，无法与 Internet 相连，因此在没有本地环回访问的情况下很难有正常的数据交互，但是为了测试本系统是否能够监测正常流量，这里我们以 Ping 命令为例，在 h2 中向 h1（即 192.168.0.1）发起 Ping 命令，如下图所示：

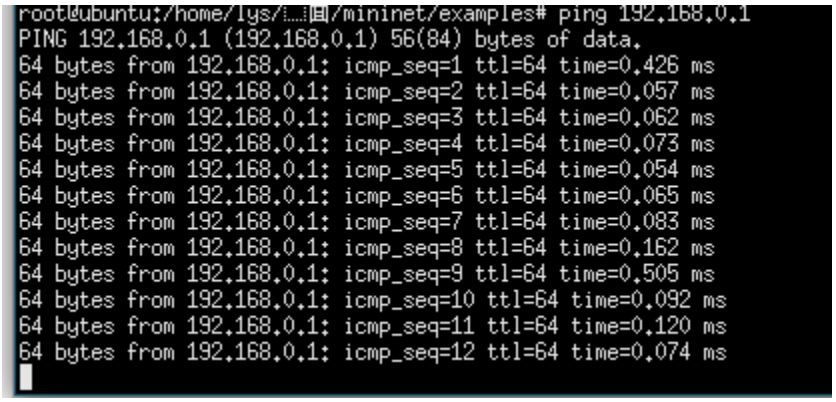

图 14 h2 Ping h1

从上图我们可以看到，我们成功通过 h2 的 ping 命令与 h1 建立连接并得到了相应的回复，在经过短暂的分析之后，我们的系统也直接给出了分析结果：

图 15 系统运行结果 2

由于我们监听的是 any 端口，所以系统直接将 h1 与 h2 的数据包都显示在了界面当中，而且根据系统的分析也成功的检测出当前的流量数据为正常流量，攻击准

确率为 0（及攻击的可能性为 0）。

4.2.3 攻击&正常流量检测

为了验证我们所设计的系统在正常网络环境中，同时拥有正常访问数据流量及攻击流量的情况下，能否成功检测出攻击流量，我们同时发起 Ping 命令及 Hping3 模拟攻击，系统的检测结果如下：

图 16 系统运行结果 3

可以看到我们的系统成功检测到了四个正常的 ICMP 数据包以及六个攻击流量数据包，并且成功给出了“受到攻击”的预测，60%的准确率也与实际情况相同！

4. 3 本章总结

在本章当中，我们着重介绍了安装配置 SDN 环境所涉及到的 Mininet 及 Ryu 控制器，并对安装工程进行了相关的介绍。

在 4.2 节中的我们分别对检测系统进行了攻击流量检测、正常流量检测以及攻击+正常流量三种检测，而我们的检测系统也表现出了良好的性能，三种检测都给出了成功的预测报警，且与实际情况相同。

第五章 结论

本文从互联网安全环境出发，系统的分析了当前互联网所遭受的攻击威胁，诸如 DDOS 等泛洪攻击为众多互联网公司带来了极大的威胁，也在很大程度上影响了正常用户的访问体验。除此之外，泛洪攻击对于近年来炙手可热的软件定义网络威胁更大，这种控制集中的网络结构很容易被泛洪攻击所影响，导致服务器资源被非法占用，影响普通用户无法访问通信。

因此本文设计实现了一种在 SDN 环境下的异常流量检测系统，主要用机器学习的方法来检测 DDOS 等泛洪攻击。在仿真 SDN 环境中我们利用流量生成工具 Hping3 模拟恶意流量，对我们网络拓扑中的 host 主机进行泛洪攻击，同时利用 Wireshark 对数据流量进行收集，在收集了大量数据之后利用 Tshark 工具出数据包进行筛选处理，筛选了五个影响最大的数据特征，再将其转换成可以用于机器学习的 CSV 文件后，利用决策树的方法对数据集进行机器学习，成功得到了入侵检测的攻击预测模型！

为了系统的美观及可操作性，我们利用 PyQt5 为其设计了相应的图形化界面，以便于系统能够直接运行在 Ubantu 系统当中，随时可以进行异常流量检测，而且为了方便检测实时流量数据，我们还在图形化界面当中增添了实时流量包显示的界面，能够及时地将捕捉到的数据包直观的显示界面上，包括源 IP 地址、端口等信息。

在经过攻击流量检测、正常流量检测以及攻击流量+正常流量检测之后，本文所设计的系统表现出了良好检测性能，能够在不影响正常流量的情况下成功检测出网络中的恶意流量，并在界面中给出相应的预警信息！

致谢

感谢江苏大学四年来对我的培养，感谢本科所有老师对我的指导，尤其感谢李致远老师我的毕业设计中给予的指导，从选题、开题、构思、资料收集等多个方面到最后论文的定稿给予的支持和帮助！使我对软件定义网络方面有了更深的了解，也使我产生了极大的兴趣。

在备考复试的过程中，因为时间的问题导致不能及时跟进毕设，非常感谢李老师的理解和耐心，使我能够放心的在参加完复试后全力完成毕业设计！在此，谨向李老师致以衷心的感谢和崇高的敬意！

也非常感谢我的室友刘亚圣在 SDN 环境搭建及图形化设计过程中对我的指导和帮助，使我能够更好的完成本次毕业设计！

最后也非常感谢在百忙之中抽出时间对本文进行审阅、评议和参加本人毕业答辩的各位老师！请各位老师批评指正！

参考文献

[1] 刘振鹏, 贺玉鹏, 王文胜,等. SDN 环境下的 DDoS 攻击检测方案 DDoS Attack Detection Scheme under SDN Environment[J]. 武汉大学学报(理学版), 2019, 065(002):178-184.

[2] 史振华, 刘外喜, 杨家烨. SDN 架构下基于 ICMP 流量的网络异常检测方法[J]. 计算机系统应用, 2016, 025(004):135-142.

[3]王燚. SDN 网络流量监控系统设计与实现[D].电子科技大学,2018.

[4] Bawany N Z , Shamsi J A . SEAL: SDN based secure and agile framework for protecting smart city applications from DDoS attacks[J]. Journal of Network & Computer Applications, 2019.

[5] R. Braga, E. Mota, A. PassitoLightweight DDoS flooding attack detection usi ng NOX/OpenFlow Proceedings - Conference on Local Computer Networks, LC N (2010), 10.1109/LCN.2010.5735752 408–15

[6] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M.Casado, N. McKeown, et al. NOX: towards an operating system for networksSIGCOMM Computer Communicat ion Review, 38 (2008), pp. 105-110, 10.1145/1384609.1384625

[7] S.A. Mehdi, J. Khalid, S.A. Khayam Revisiting traffic anomaly detection using software defined networking International Workshop on Recent Advances in Intrusion Detection, Springer, Berlin, Heidelberg (2011), pp. 161-180, 10.1007/978-3-642-23644-0_9

[8] C. Yu Hunag, T. Min Chi, C. Yao Ting, C. Yu Chieh, C. Yan Ren A no vel design for future on-demand service and security International Conference on Communication Technology Proceedings, ICCT (2010), 10.1109/ICCT.2010.5689156 385–8

[9] S. Dotcenko, A. Vladyko, I. Letenko A fuzzy logic-based information sec urity management for software-de fi ned networks. Advanced communication Tech nology (ICACT)2014 16th International Conference on (2014), pp. 167-171, 10.110 9/ICACT.2014.6778942

[10] A.F.M. Piedrahita, S. Rueda, D.M.F. Mattos, O.C.M.B. Duarte FlowFence: a denial of service defense system for software defined networking Global Information Infrastructure and Networking Symposium (2015), 10.1109/GIIS.2015.7347185 GIIS 2015

[11] T.V. Phan, N.K. Bao, M. Park Distributed-SOM: a novel performance bottleneck handler for large-sized software-defined networks under flooding attacks J. Netw. Comput. Appl., 91 (2017), pp. 14-25, 10.1016/j.jnca.2017.04.016

[12] B. Han, X. Yang, Z. Sun, J. Huang, J. Su OverWatch : a cross-plane DDoS attack defense framework with collaborative intelligence in SDN Security and Communication Networks,Hindawi (2017)

[13] F. Shang, L. Mao, W. Gong Service-aware adaptive link load balancing mechanism for Software-Defined Networking Future Gener. Comput. Syst. (2017), 10.1016/j.future.2017.08.015

[14] A. Zaalouk, R. Khondoker, R. Marx, K. Bayarou OrchSec: an orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions IEEE/IFIP NOMS IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World 2014 (2014), 10.1109/NOMS.2014.6838409