[SDN 中基于注意力机制的异常检测系统]

设计文档

所在赛道与赛项：B-EP1。

一、目标问题与意义价值

软件定义网络(SDN) 是一种创新的网络架构与管理方法，通过将网络控制平面与数据转发平面进行解耦，实现网络的集中控制和编程的灵活性。通过将控制平面和数据转发平面分离，SDN 实现了网络的集中控制和可编程性。网络管理员可以通过控制器对整个网络进行集中管理和编程，而无需逐个配置每个网络设备。这样可以简化网络管理任务，并提供更好的灵活性和可扩展性。SDN 还支持网络的动态性和自动化。通过控制器的编程接口，管理员可以编写应用程序和策略来实现网络的自动化配置、流量工程、安全策略等功能。这种灵活性和自动化能力为网络创新和应用提供了更多的可能性，同时也为云计算、大数据和物联网等新兴技术的发展提供了支持。

虽然 SDN 架构可以提供更好的灵活性和可编程性，但也引入了一些新的安全风险和挑战。SDN 架构中的控制器作为网络的中枢，控制器的安全性至关重要。攻击者可能试图入侵控制器，以获取对网络的完全控制和篡改控制器的指令和策略。为解决这些问题，我们设计了一种基于注意力机制的网络攻击检测系统，系统共分为两个阶段：线下训练和线上检测。系统中通过使用注意力机制完成对数据特征的提取，训练时使用获取到的网络攻击数据进行训练，在训练完成后可部署到网络中做到对攻击数据的多分类检测，实现判定数据流是否属于攻击以及对应的攻击类型。该系统的主要目标是通过一种端到端的检测模型，无需进行复杂的人工处理，即可实现对攻击的检测与分类。

例如，在需要统一管理复杂网络的校园网络中，SDN 控制器可以通过提供集中管理和自动化，改进安全性和提升整个网络的应用级服务质量，从而使校园网络受益。同时，SDN 技术也适合应用于云网络。云服务提供商通常需要在同一物理基础设施上创建和管理多个逻辑网络，通过使用 SDN，网络管理员可以更容易地实现网络隔离，提供更高级别的网络服务，如网络功能虚拟化（NFV）。在安全性方面，我们的检测系统可以部署在云网络的 SDN 控制层，以提高对攻击的检测速率和准确性。

总的来说，SDN 通过解耦网络的控制平面和数据转发平面，引入中心控制器和编程接口，实现了网络的集中控制、可编程性和自动化。这种新的网络架构和管理方法为网络管理人员带来了更大的灵活性和控制能力，并为网络创新和应用提供了更多的机会。但随着应用范围的扩大，相应的网络安全威胁也随之提高，我们的异常检测系统通过利用注意力机制的独特优势，使用注意力机制进行细粒度的特征提取，可以提高对攻击的检测准确率。

二、设计思路与方案

2.1 系统的总体框架

系统的总体设计结构采用线下训练和线上检测两个模块。提取到的数据流需要先进行特征的初步统计，再将各类攻击数据进行模型训练，线上检测阶段只需要使用训练好的模型即可。系统的总体架构如图 2 所示。

以下将基于 OpenFlow 通信协议下的工作流程详细说明攻击检测过程。

1、信息采集

攻击检测与分类依赖于各个交换机的流表信息，SDN 网络在信息采集模块是基于 Ryu 控制器添加了 monitor 方法对于网络中交换机的流表进行实时采集。Monitor 方法在实现二

层交换机的基础上，对交换机的上线和下线情况进行监控，包括请求端口信息和流表信息。通过对网络设备的持续性监控，获取到流表的具体信息后，将采集到的流表信息进行解析保存，得到流表的统计数据以及流表的详细信息。

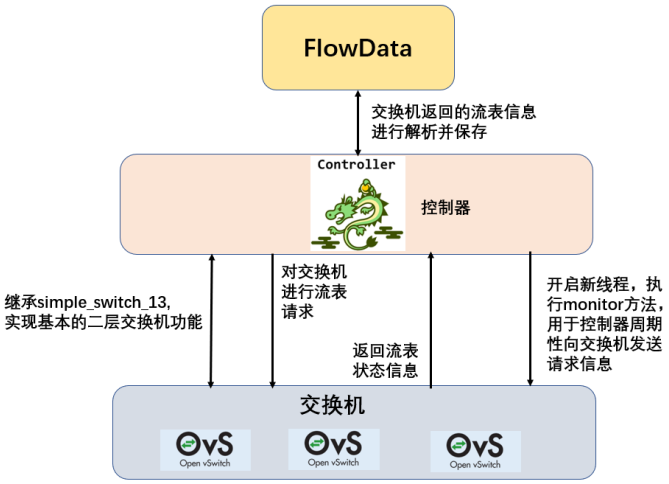

图 1　monitor 算法采集流表信息流程

2、在线下使用采集到的数据进行模型的训练。训练完成后，模型参数固定，将模型部署到控制器当中进行线上检测攻击类别。通过防御模块来调用北向接口，更新流表规则，从而达到对攻击数据的防御。

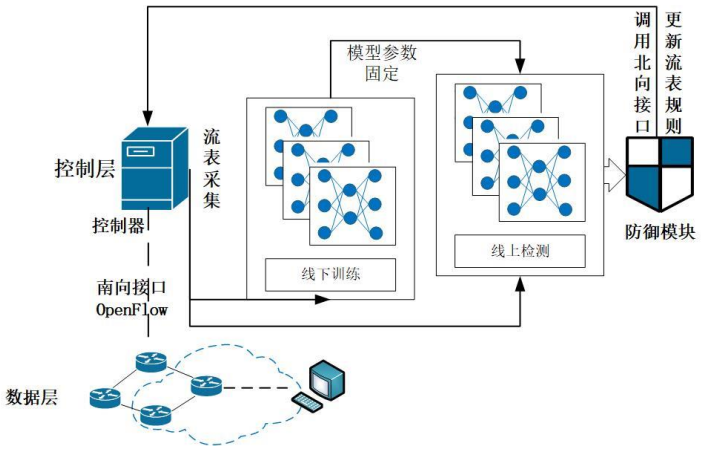

图 2 攻击检测模型总体架构图

三、方案实现

3.1 基于注意力机制的攻击检测模型

检测模型共分为三个部分：数据预处理、特征提取、输出类别。通过前期的数据预处理，将采集到的信息处理为方便模型输入的向量值，随后将其输入到基于注意力机制的编码器中进行特整提取，最后通过线性层和 Softmax 输出检测类别。模型的整体框架如图 2 所示。

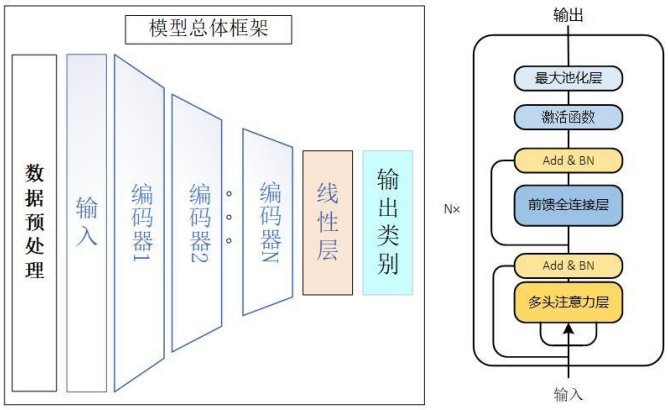

图 3 基于注意力机制的攻击检测模型总体框架(左)、编码器架构(右)

3.2 预处理

对于采集到的统计数据，需要先进行数据预处理，为了防止模型过拟合，我们隔离了统计数据中对应的 IP 以及端口信息，同时对于离散型变量(协议类型、标志信息等)和连续型变量(流持续时间、流的平均每秒字节数等)分别进行处理。

离散型变量：由于其种类固定，可使用词向量嵌入的方式将每个类别映射到一个稠密的实值向量，这些向量可在训练过程中进行学习和优化。

连续型变量：由于统计信息中存在过大值与过小值，采用简单的最大最小归一化等无法准确反应统计数据的分布情况，所以采用反正切归一化将数值变换到指定范围[0,1]内，便于模型的输入。具体公式如下：

$$Y = \frac{\arctan(x)*2}{\pi} \quad (1)$$

完成归一化操作后，需要对连续型变量进行向量映射，通过对连续性变量T_j^{num} 与向量W_j^{num} 进行主元素乘法操作，实现对连续型变量的向量化操作。

$$T_j^{num} = b_j^{num} + x_j^{num} * W_j^{num} \quad (2)$$

其中 x_j^{num} 代表第 j 个连续型变量，W_j^{num} 为向量映射矩阵，b_j^{num} 为第 j 个特征偏差。最后将连续型变量与离散型变量向量化后的值进行合并后，统一做为模型的输入

$$T = stack[T_1^{num}, T_2^{num}, \dots, T_k^{num}, T_1^{cat}, T_2^{cat}, \dots, T_k^{cat}] \quad (3)$$

数据向量化映射图如图 4 所示。

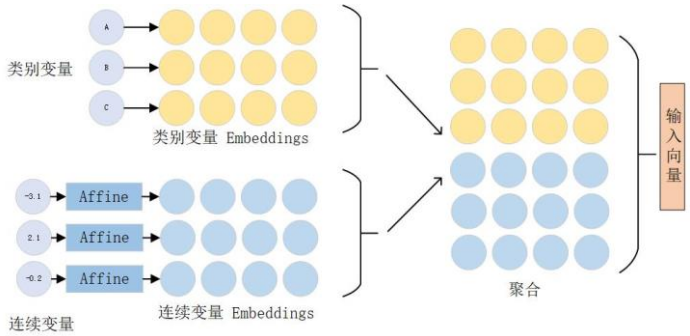

图 4 数据向量化映射图

3.3 编码器架构

编码器的主体架构采用“金字塔”式架构，利用注意力从高层信息到低层信息，逐步学习区域相似性，并填充特征图。编码器中共包含 N 组子编码器，通过利用注意力机制不断优化获取到的特征信息。详细过程如图 3 所示。

3.3.1 编码器层

编码器架构采用“金字塔”式架构，通过堆叠多个编码器模块来实现多尺度特征提取。每个编码器模块多头注意力层、残差连接、前馈全连接层和池化层组成，用于逐渐减小特征图的尺寸和增加感受野。这样，底层编码器会捕获更细节的信息，而高层编码器会具有更大的感受野和更抽象的语义信息。

多头注意力层：注意力机制（Attention Mechanism）是在计算能力有限的情况下，将计算资源分配给更重要的任务，同时解决信息超载问题的一种资源分配方案。在神经网络学习中，一般而言模型的参数越多则模型的表达能力越强，模型所存储的信息量也越大，但这会带来信息过载的问题。那么通过引入注意力机制，在众多的输入信息中聚焦于对当前任务更为关键的信息，降低对其他信息的关注度，甚至过滤掉无关信息，就可以解决信息过载问题，并提高任务处理的效率和准确性。自注意力模块首先将输入 X 经过线性变换得到 query Q，key K 和 value V 三个矩阵，将输入序列投影到该矩阵上，得到自注意力输出向量。注意力权重是通过计算 query 和 key 相关性再进行 softmax 得到的，通过缩放点积注意力(scaled dot-product attention)对相似度进行计算,自注意力计算公式如下：

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) * V \quad (4)$$

其中，d_k是矩阵 Q、K 的列数，即向量维度。

多头注意力机制其实就是将原始的输入序列进行多组的自注意力处理过程；然后再将每一组自注意力的结果拼接起来进行一次线性变换得到最终的输出结果。通过并行使用多个自注意力模块，使得不同的头部关注不同的信息。多头注意力机制能够形成多个子空间且注意力机制在不同的空间中分布不同，所以将 Q,K,V 投影到不同的向量空间，从而使模型可以学习到输入数据之间的各个角度关系。多头注意力模块公式为：

$$MultiHead(Q,K,V) = Concat(head_1, \dots, head_h)W^O \quad (5)$$

$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (6)$$

残差链接层：残差链接（Residual connections）是在深度神经网络中引入的一种连接方式，旨在解决深层网络训练过程中的梯度消失和梯度爆炸问题，并加快网络的训练速度和提高模型性能。残差链接通过在网络的不同层之间引入跨层的直接连接，将底层的输入信号与高层的输出信号相加。这样做的目的是将底层的信息直接传递到高层，使得梯度能够更容易地流经网络并保持其有效性。具体来说，残差链接通过跳过一部分网络层的计算，允许底层的信息直接流向更高层，从而提供了一种捷径，使得梯度能够更容易地传播。残差连接层公式如下：

$$Res(x) = x + F(x) \quad (7)$$

其中，$F(x)$ 为多头注意力层的输出.

前馈全连接层：在残差链接层结束后，进入到前馈全连接层，前馈全连接层将输入信号与权重矩阵进行矩阵相乘，并加上偏置向量，然后通过激活函数进行非线性变换，最终输出转换后的特征表示。公式如下：

$$FeedForward(x) = Relu(x \cdot W + b) \quad (8)$$

最大池化层：通过在输入数据的局部区域中选择最大值作为输出，以减少特征图的尺寸，并提取出主要的特征信息。

3.3.2 输出层

模型在编码器的最后通过拼接一个线性层 和 Softmax 层，通过获取编码器层对信息的特征提取，最后使用线性层实现对所有信息的统一。具体公式如下：

$$Y = Softmax(Linear(x)) \quad (9)$$

$$Softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} \quad (10)$$

3.4 损失函数设置

损失函数使用交叉熵损失函数，具有非负性和目标最小化的优点，通过最小化交叉熵损失函数，可以使模型的预测概率尽可能接近真实标签的概率分布，从而提高模型在多分类任务中的准确性。

$$loss(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i) \quad (11)$$

3.5 模型中其他参数设置

整体模型的采用 Pytorch 搭建完成，使用 AdamW 作为优化器。AdamW 是对 Adam 优化算法的一种改进。AdamW 算法结合了 Adam 算法的自适应学习率和权重衰减（Weight Decay），可以可以更好地处理权重衰减和自适应学习率之间的相互影响，提高模型的训练效果和泛化能力。

四、系统运行结果/应用效果

系统实验环境是运行在 VM 虚拟机上，操作系统为 Ubantu20.04 64 位系统，SDN 环境是基于 Mininet 仿真平台搭建网络拓扑。此外，实验使用 hping3 攻击模拟网络中 DDoS 攻击流量。SDN 采用了 OpenFlow13 版本协议、Ryu 控制器和 OpenvSwitch 交换机。本文使用了线性拓扑，包含一台 Ryu 控制器，和 6 台 OpenvSwitch 交换机和通过交换机连通的 18 台主机，网络拓扑如图 5 所示。

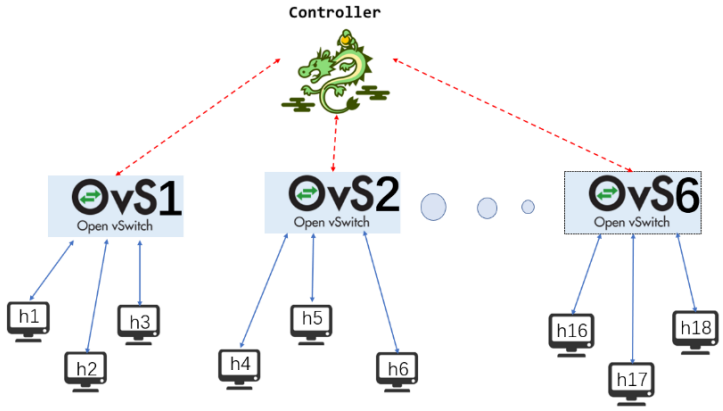

图 5　SDN 网络拓扑

4.1 系统初始

系统启动前要先运行 SDN 环境，在服务端 C4 目录下输入 ryu-manager ofctl_rest.py c4_switch.py 启动 Ryu 控制，打开 REST API 服务；在 sFlow-rt 目录下输入./start.sh 打开 sFlow-rt 服务。在客户端 C4 目录下运行编辑的 Mininet 脚本。服务器端、客户端搭建 SDN 环境操作如图 6、图 7 所示，系统初始运行界面如图 8 所示。

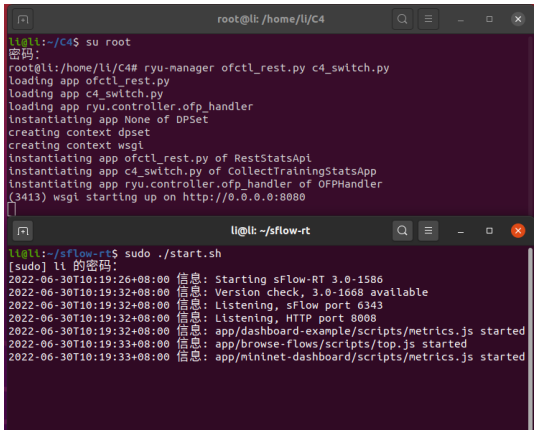

图 6 服务器端 SDN 环境搭建

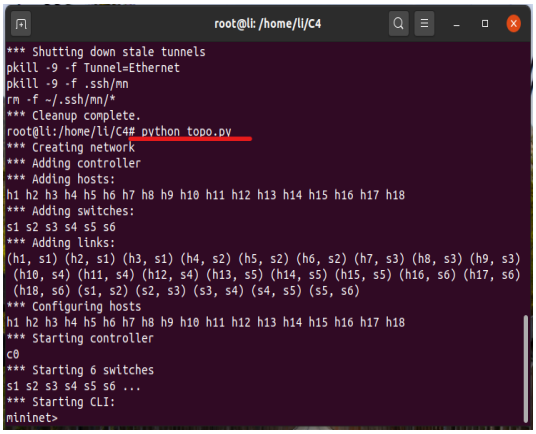

图 7 客户端 SDN 环境搭建

图 8 系统初始页面

4.2 异常检测

我们模拟了在正常情况下的数据访问，并进行异常检测，系统返回结果如图 9 所示:

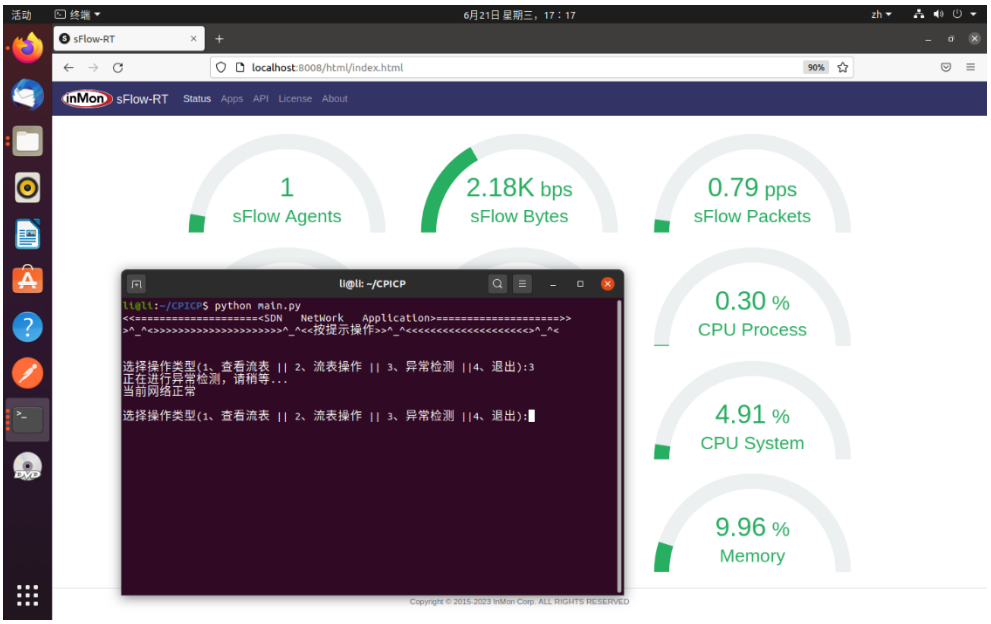

图 9 正常情况下进行检测

通过模拟 DDOS 攻击时进行异常检测，系统返回结果如图 10 所示

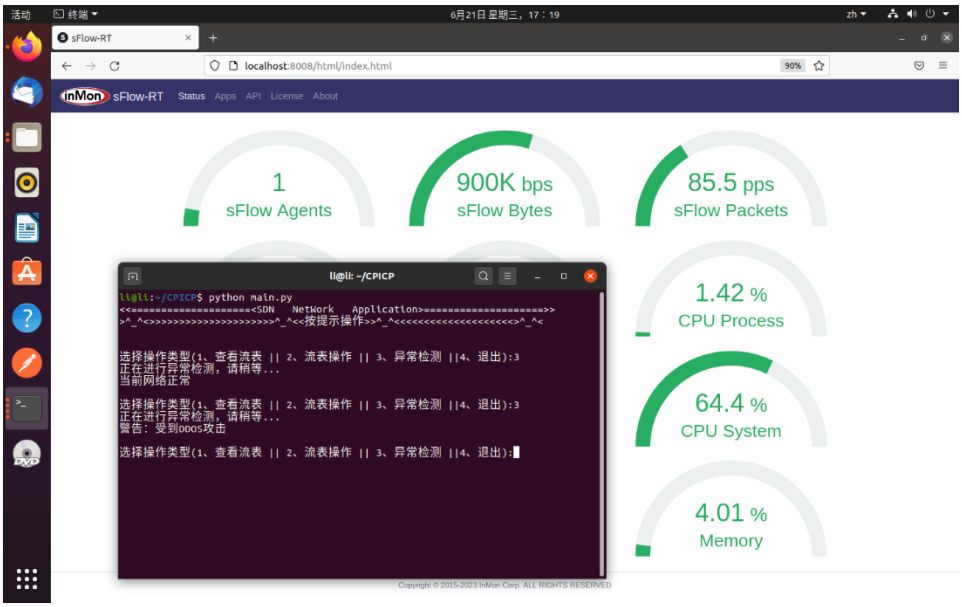

图 10 模拟 DDOS 攻击后进行异常检测

4.3 使用数据集进行检测

为了更丰富的展示我们提出的对异常的多分类效果，我们在三个公开数据集上进行了实验，分别为 CIC_IDS_2017，INSDN 以及我们自己模拟的 DDOS 攻击数据集。

4.3.1 评价指标：

相关缩写说明：

TP（True Positives）表示真正例，即模型正确预测为正例的样本数量。这是模型正确地将正例判断为正例的情况。

FP（False Positives）表示假正例，即模型错误地将负例判断为正例的样本数量。这是模型将负例错误地判断为正例的情况。

FN（False Negatives）表示假反例，即模型错误地将正例判断为负例的样本数量。这是模型将正例错误地判断为负例的情况。

精准率：

$$\text{precision} = \frac{TP}{TP+FP} \quad (12)$$

召回率：

$$\text{recall} = \frac{TP}{TP+FN} \quad (13)$$

F1 分数：

$$\text{F1} = \frac{2*(precission*recall)}{precission+recall} \quad (14)$$

4.3.2 实验结果

CIC_IDS_2017 中共包含 9 个类别，囊括了常见的一些攻击信息，关于 CIC_IDS_2017 的实验结果如表 1 所示。

指标 攻击类型	precision	recall	f1-score
DDoS	0.99899194	1.00000000	0.99949571
DoS GoldenEye	0.99325301	0.99903054	0.99613340
DoS Hulk	0.99769585	0.99483056	0.99626114
DoS Slowhttptest	0.98570067	0.98102467	0.98335711
DoS slowloris	0.98400000	0.98751115	0.98575245
FTP-Patator	0.99827139	0.99568966	0.99697885
Infiltration	1.00000000	0.81818182	0.90000000
PortScan	1.00000000	0.98076923	0.99029126
SSH-Patator	0.99346405	0.99184339	0.99265306
平均值	0.99459743	0.97209789	0.98232478

表 1 在 CIC_IDS_2017 数据集上的实验结果

INSDN 数据集是真实环境下当 SDN 受到 DDOS 攻击时采集到的，其中共包含 2 个类别，分别为正常和异常数据，关于 INSDN 的实验结果如表 1 所示。

指标 攻击类型	precision	recall	f1-score
BEGIN	0.99808838	0.99845543	0.99827187
DDOS	0.99917686	0.99898107	0.99907896
平均值	0.99863262	0.99871825	0.99867541

表 2 在 INSDN 数据集上的实验结果

模拟数据集是真实环境下当 SDN 受到 DDOS 攻击时采集到的，其中共包含 2 个类别，分别为正常和异常数据，关于模拟数据集的实验结果如表 1 所示。

指标 攻击类型	precision	recall	f1-score
BEGIN	0.99997070	1.00000000	0.99998535
DDOS	1.00000000	0.99998578	0.99999289
平均值	0.99998535	0. 99999289	0. 99998912

表 3 在模拟攻击数据集上的实验结果

通过在三个真实数据集，以及实际环境测试中可以得出，我们所设计的模型可以很好的检测出攻击信息，并根据攻击信息判别攻击类别。在三个数据集上的实验平均准确率都达到了 99%以上，具备一定的防御效果。

五、创新与特色

1、系统在信息采集使用基于 Ryu 的 monitor 算法实现了网络的流量监控，加快了系统采集速率。

2、设计了一种新颖的异常检测模型，通过利用注意力机制的强大能力，实现对攻击信息的获取，并据此判别攻击类别，通过实验证明我们所提出的模型具有很高的准确率。使系统可以精准定位攻击信息，判别攻击类型，根据攻击信息进行进一步的防御。

3、系统为 python 脚本实现，各个模块独立，在实际应用中只需改变数据接口即可适用，对 SDN 环境要求低，有很强的扩展性。