针对 VxWorks 操作系统的漏洞复现方法研究

摘要：随着嵌入式系统和物联网技术的快速发展，VxWorks 操作系统作为广泛应用于各种嵌入式设备和 PLC（可编程逻辑控制器）的实时操作系统，其安全性日益受到关注。论文致力于研究 VxWorks 操作系统的漏洞复现方法，特别是针对 CVE-2015-7599 这一特定漏洞进行深入分析。通过对该漏洞的根因进行静态分析，揭示其产生的根本原因；利用动态调试技术，详细剖析了该漏洞的利用路径；进一步，在 PLC 设备上模拟了漏洞的利用过程，并研究了远程权限提升的方法。论文希望通过对 VxWorks 操作系统的已知漏洞进行分析，发现其脆弱环节，理清 VxWorks 系统安全研究方向，为后续安全研究提供参考。

关键词：VxWorks；嵌入式系统；漏洞复现；漏洞利用；PLC

1 引言

VxWorks 嵌入式操作系统作为最具代表性的实时操作系统之一，被广泛应用于多个领域，包括工业控制、航空航天、通信设备等重要领域[1]。随着工业自动化程度不断提高，VxWorks 操作系统的安全性日益受到关注，研究并分析 VxWorks 系统中的安全漏洞，对于保障工业控制系统的安全具有重要意义。然而，由于 VxWorks 特殊的架构和应用场景，其漏洞复现研究相对较少，这为展开相关研究提供了动力和空间。

本研究旨在通过对 VxWorks 操作系统中 CVE-2015-7599 漏洞的深入分析，探讨针对 PLC 设备的漏洞利用技术。通过静态分析和动态调试，揭示漏洞的根本原因及其利用路径，并实现出相应的漏洞利用程序。研究成果将有助于提升工业控制系统的安全性，为今后相关领域的研究提供理论和技术支持，从而推动信息安全技术的发展。

2 相关工作

国外方面，部分安全研究人员和团队已经开始关注 VxWorks 操作系统的安全性，并在漏洞挖掘和利用方面进行了一些研究。2010 年，HD Moore 提出并分析了关于 VxWorks 操作系统使用的 WDB RPC 远程调试协议中存在的漏洞[2]。2015 年初，加拿大的研究人员 Yannick Formaggio 公布了两个 VxWorks 系统安全漏洞。其中一个高危漏洞是缓冲区溢出类漏洞，如果攻击者以极高速度向 VxWorks 系统发送用户名和密码

的时候，内置在系统内部的 FTP 服务器会产生环形缓冲区溢出，导致系统崩溃[3]。在 2019 年的 Black Hat 黑客大会上，来自 IOActive 的研究人员公开了波音 787 客机的 CIS/MS 组件多个漏洞。CIS/MS 组件采用了 VxWorks 系统，位于飞机的两路通讯总线的边界上，该组件存在漏洞的二进制文件未开启 NX、调用栈保护等防护措施，导致攻击者通过一个常见的内存漏洞就能够劫持程序执行，而攻击者可以通过远程、物理等多种方式实现成功入侵[22]。之后，攻击者可以利用这些漏洞对波音客机的机身网络总线进行渗透，向机身的关键系统发送恶意指令，造成安全威胁。研究人员还公开了四个可供利用的漏洞，并将其组成了攻击链，以 TFTP 服务栈溢出获得执行权限，继而通过 VxWorks 的内核漏洞提权，获得飞机内部网络的控制权。此外，SecNiche Security Labs 也曾发表文章讨论了部分使用 VxWorks 操作系统的固件中存在的漏洞以及漏洞成因分析[23]。

国内方面，对 VxWorks 操作系统漏洞的研究也逐渐兴起。Liu 等人利用固件分析技术对嵌入式操作系统进行反向分析来搜索系统漏洞，然后采用改进的模糊测试方法对嵌入式操作系统进行模糊测试，从而对漏洞的安全状况和级别进行评估[24]。Wei 等人利用了 CVE-2015-7599 漏洞，验证了其设计的漏洞挖掘框架的有效性[4]。Zhang 等人分析了 VxWorks 操作系统中存在的输入验证、权限与访问控制、信任管理和加密等安全漏洞，描述了这些漏洞的风险、攻击途径和受影响的系统[5]。

国内外对 VxWorks 操作系统漏洞研究的关注度逐渐增加，但相关的系统性研究成果尚有待进一步完善。未来的研究方向可以包括对已知 VxWorks 漏洞的深入分析与复现、漏洞利用技术的研究、安全加固措施的提出以及漏洞复现工具和框架的开发等方面，以期丰富 VxWorks 漏洞复现研究的成果和方法。

3 模型设计

CNN 实际上早在大规模图像特征识别和分类的应用中就取得了很大的成功，是一种非常经典的前馈神经网络模型。CNN 主要由卷积层（Convolutional Layer），激活函数（Activation Function），池化层（Pooling Layer），全连接层（Fully Connected Layer），损失层（Loss Layer）组成[6]。卷积层通过运用不同的卷积核在输入图像上滑动，做卷积运算而得到一组平行的特征图谱。紧跟在卷积层后面的激活函数增强整个网络的分非线性特征。池化层将输入的图像划分成若干个矩阵区域，每个子区域输出一个统计值，如平均值、最大值等。实际上池化层是一种下采样方式，通过池化层对原始数据进行维度的降低，从而避免过拟合。全连接层中的每个神经元都与前一层的所有神经元进行全连接，来整合所有卷积层或池化层的局部信息?损失层则是 CNN 的最后一层，通过损失函数来计算模型预测结果和期望输出的误差，并且利用反向传播算法计算 CNN 各层中参数的梯度，通过梯度信息逐层更新所有参数，从而优化整个网络。图 1 展示了一个用作图像分类的经典 CNN 模型：VGG16 的结构示意图。目前 CNN 模型已经广泛的应用在乳腺病理图像的研究中，并且在医学图像分割、医学图像识别中有着很大的应用

空间和前景。

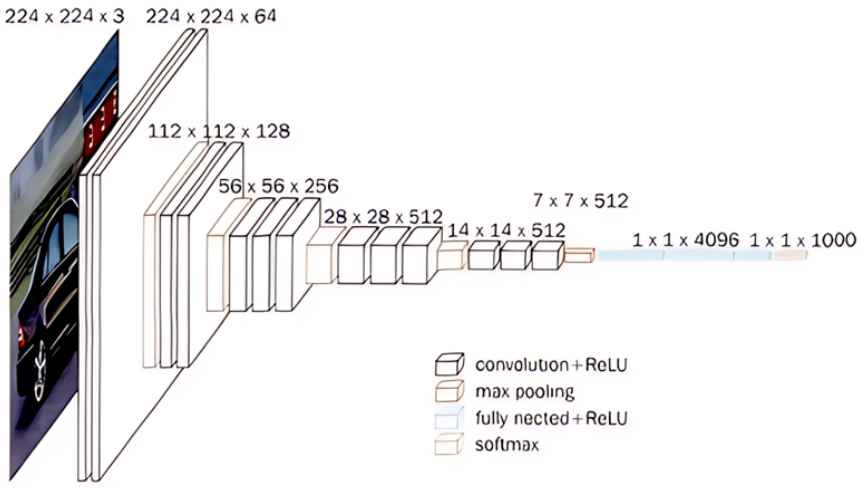

图 1: 经典 CNN 模型：VGG16

3.1 YOLO 算法

YOLO 算法首先运行和计算的速度非常快，能够达到实时的要求，因此在用于处理视频和图像的时候可以快速的得到结果。此外，使用全图作为 Context 信息，背景错误（把背景错认为物体）比较少。这样可以保证在识别物体的同时，保证一定量的准确性。同时，YOLO 算法的泛化能力很强，即对新鲜样本的适应能力很强，在引入新样本时，如果和过去的样本有相同的规律，网络也能给合适的输出[7]。YOLO 算法的核心思想是把一整张图片作为网络的输入，直接在输出层回归边界框 (Bounding box) 的位置和边界框所属的类别。YOLO 利用一个单一的卷积神经网络到整体形象的这种方法将输入图像分为子区域并用它们的类预测多个边界框每个区域的概率。不像传统的 R-CNN 要求许多网络对于所有提取的区域，YOLO 利用整个乳房医学影像，以便在训练和测试时间内用单个网络完全编码预测器及其方面的上下文信息。

3.1.1 UNet 网络

文本预处理是自适应漏洞库构建的基础，预处理效果的优劣直接影响到漏洞库的质量。原始漏洞数据通常以 HTML、JSON 或半结构化文本形式存在，需通过多阶段清洗与转换实现知识抽取，因此本模型采用正则表达式实现轻量化、高精度的文本清洗与关键特征抽取。

U-Net 网络是一个典型的编码器-解码器结构，是对 FCN 的改进，其结构如图 7 所示。在解码器中进行特征融合时，U-Net 采用通道维度叠加的方式，形成更厚的特征图，然后通过卷积进行特征融合[8]。为了充分利用编码器阶段提取到的特征，每个解码器阶段都与编码器阶段进行跨越连接，来弥补上采样时丢失掉的细节信息。基于小步长的逐步上采样和每个阶段的跨越连接，U-Net 体现出了强于 FCN 的性能。预处理结果以

JSON 格式封装，包含标准化字段以支持后续语义向量化与检索。上述文本预处理过程不仅改善了原始数据的异构性问题，还通过语义结构化增强了漏洞特征的可检索性，为后续向量化与数据库嵌入提供了高质量的输入。

3.1.2 语义向量化

Rpcbind 服务主要功能是将 RPC 程序号映射到网络端口号。它通常监听在一个端口 111，并维护一个数据库，记录所有本地可用的 RPC（远程过程调用）服务及其对应的端口号。当客户端希望与某个 RPC 服务通信时，它首先会查询 rpcbind 以获取该服务的端口号。这种端口映射功能极大地简化了客户端与服务器之间的通信[9]。客户端不需要预先知道每个 RPC 服务的具体端口号，只需通过 rpcbind 获取相关信息即可。这不仅提高了通信的灵活性和效率，还减少了配置和维护的复杂性。

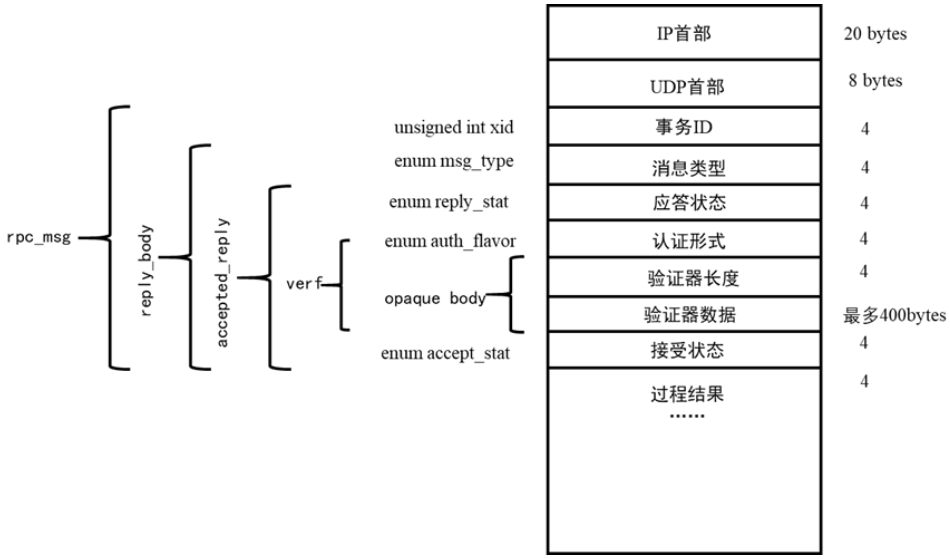

图 2: 封装在 UDP 中成功应答的 RPC 应答格式

尽管 rpcbind 在网络服务中发挥着重要作用，但它也会成为攻击目标。攻击者可能通过 rpcbind 获取服务器上运行的服务信息，从而发起进一步的攻击。本论文所研究的 CVE-2015-7599 漏洞就是 rpcbind 服务中的一个安全漏洞，允许攻击者通过精心构造的请求远程获取服务信息，甚至可能导致任意代码执行。

3.1.3 向量数据库嵌入

向量数据库的核心原理在于将非结构化数据通过嵌入模型转化为稠密的高维向量，再通过相似度检索完成语义级的信息查找[10]。以文本为例，传统的关键词匹配难以捕捉语义间的深层联系，而嵌入模型如 BERT 或 text-embedding-3-large 可以将一句话编码为固定维度的语义向量，使语义相似表达在向量空间中距离更近。向量数据库通常采用高效的近似最近邻（ANN）算法，如 HNSW 或 IVF，通过构建图索引或聚类索引结构，在大规模数据中实现毫秒级的相似向量查找。相比传统关系型数据库，向量数据库更适

合处理复杂语义、模糊匹配、多模态查询等场景，广泛应用于智能问答、推荐系统、搜索引擎和知识图谱等系统中。

向量数据库的效果很大程度上取决于嵌入模型和索引机制的选择。嵌入模型负责将原始数据转化为语义向量，不同任务适合不同模型，例如通用语义搜索常用 BERT 或 OpenAI 提供的嵌入接口，医学或法律等垂直领域则依赖专门训练的预训练模型。生成的向量维度也会影响性能，维度过高会增加计算与存储开销，维度过低则可能无法有效表达语义差异[11]。为了在大规模数据中快速检索相似结果，常见的做法是使用 HNSW 或 Faiss 等近似最近邻算法构建高效索引结构。实际系统中，通常还会结合向量检索与属性过滤，对查询结果进行二次筛选，以提升精度与可控性。随着多模态应用的发展，越来越多的系统支持将文本、图像、音频等多种类型的向量统一管理，构建更复杂的语义检索框架。

3.2 安全性设计

安全性是本软件设计的核心之一，特别是在控制端与目标端的通信过程中。利用基于 AES-128 加密的控制连接和更为复杂的功能连接加密机制，确保了数据传输的机密性和完整性。此外，每次会话采用一次性会话密钥，进一步提高了安全性。控制端和目标端在建立连接时都需经过严格的身份验证，防止非法设备的接入，保障系统的安全运行，AES 加解密流程如图 3 所示。

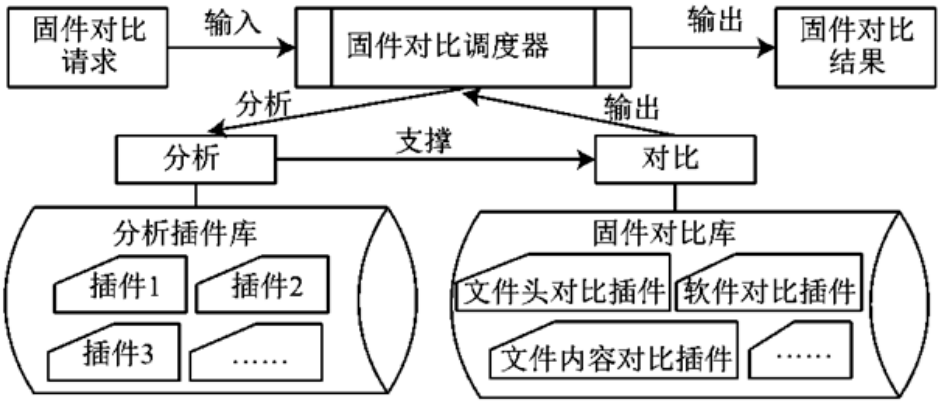

图 3: 安全性设计

系统在通信协议中引入了时间戳和消息摘要校验机制，确保每条指令在有效时间窗内完成验证，并具备可追溯性[12]。通信过程中所有敏感操作指令均经过双向认证和签名确认，避免指令被伪造或篡改。为了应对潜在的弱口令风险，目标端强制启用基于策略的口令复杂度校验机制，并支持设备级别的访问白名单配置。整体安全框架在保证通信效率的同时，兼顾灵活性与可扩展性，能够适应工业控制环境下多种部署形态与安全要求。

3.3 静态分析

大语言模型在抽取关键特征后，会将增强后的语义向量输入自适应漏洞库进行检索匹配。为了衡量特征向量之间的相似程度，系统采用最近邻搜索（Nearest Neighbor Search, NNS）算法，并以余弦相似度作为评估指标。通过计算向量之间的夹角余弦值，判断它们在语义空间中的方向接近程度，从而识别出对应的攻击特征类型。余弦相似度计算如公式（1）所示。

$$\text{similarity} = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \cdot \|\vec{B}\|} \tag{1}$$

其中，$\vec{A}$ 和 $\vec{B}$ 分别表示增强后的攻击特征向量和自适应漏洞库中的向量。余弦相似度的值范围为 $[-1, 1]$，值越接近 1，表示两个向量越相似。

静态分析是一种在不实际执行程序代码的情况下，通过词法分析、语法分析、控制流分析、数据流分析等来深入探索程序代码特性、结构、行为及潜在问题的过程[13]。与动态分析不同，静态分析不依赖于程序的运行时环境或状态，而是直接对源代码或编译后的可执行程序进行分析。静态分析的优势在于其全面性和系统性[14]。静态分析技术可以对程序中所有的路径进行覆盖，甚至可以检查很难执行到的代码片段，往往可以发现实际运行难以触及的错误。但同时由于静态分析不运行程序，无法营造程序真实的运行环境，可能会产生漏报和误报。

3.4 动态分析

3.4.1 单步跟踪

跟踪调试法也是动态分析的关键技术之一，通常借助调试器对程序运行状态进行实时监控，通过单步执行、断点设置、寄存器查看和内存监控等手段，有助于深入理解漏洞利用代码的具体行为逻辑，验证其触发条件与执行路径，并进一步评估其对目标系统造成的潜在影响和危害程度。

单步跟踪是属于细粒度的分析方法，通过逐步执行代码，监控每一个操作的执行结果[15]。这种方法能够详细观察漏洞复现和利用过程中的每一个系统调用、寄存器状态、堆栈变化和内存访问。通过单步跟踪，我们可以精确地识别代码中的关键操作，例如缓冲区溢出、堆喷射、代码注入等，以及这些操作是如何影响系统的正常功能的[16]。单步跟踪的主要优势在于其能提供对漏洞利用代码执行的深入理解，但其缺点是分析过程相对耗时，并且需要较高的技术知识来解读跟踪数据。

在实际漏洞验证过程中，调试器用于分析程序的异常行为，也被用来定位关键指令执行点、系统调用入口以及内存读写异常区域[17]。结合符号调试信息和断点控制，研究人员能够捕捉漏洞触发前后的上下文状态，辅助识别缓冲区溢出、堆喷射、格式化字符串等典型漏洞类型的利用链。通过调试日志的记录，还可以复现实验流程，便于后续复查与复测。

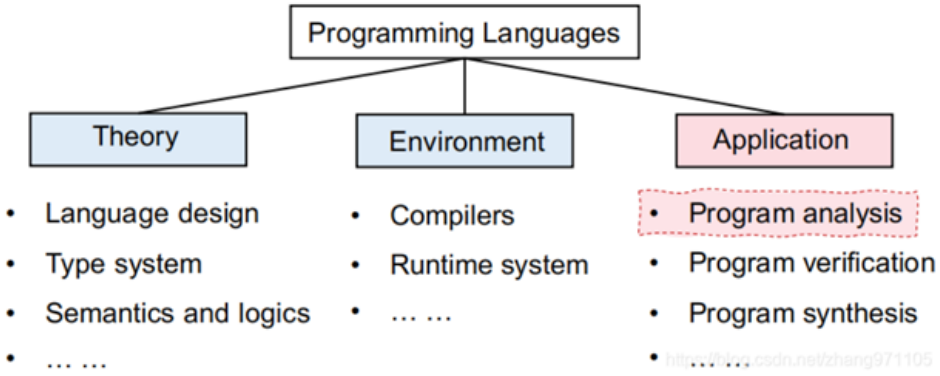

图 4: 单步跟踪

在多源语义结果存在偏差的场景中，系统引入了一个辅助判定模块，专门用于对主模型输出进行约束与修正[18]。该模块首先通过一套基于领域语料构建的术语标准体系，对输出结果中存在歧义或非标准表达的部分进行统一转换，从而消除术语混乱带来的干扰。当模型判断出现模糊边界时，该模块不依赖原始标签本身，而是回溯关联的协议语义、时间行为序列与载荷内容，构建上下文环境进行一致性比对。在面对未知或未显式建模的问题类别时，系统不完全依赖主模型的生成结果，而是引入基于人工知识的条件规则网络，从多个独立特征中提取逻辑约束，以纠正模型可能存在的泛化偏移[19]。最终，辅助模块将主模型的语义输出作为初始参考，结合自身独立的匹配判断过程，生成融合得分，自动完成最优标签的选取。最终标签的选取过程如公式（2）所示。

$$\hat{y} = \arg\max_{y_i \in Y} (\alpha \cdot P_{\text{model}}(y_i) + (1 - \alpha) \cdot S_{\text{rule}}(y_i)) \tag{2}$$

其中，$\hat{y}$ 表示所有候选标签中加权得分最大的标签，α 表示动态权重系数（默认值为 0.7），$P_{\text{model}}(y_i)$ 表示主模型输出的置信度概率值，$S_{\text{rule}}(y_i)$ 表示规则引擎的匹配度评分。

3.4.2 内核 hook

系统钩子方法侧重于监视代码执行过程中的系统 API 调用，特别是那些关键的系统函数，如内存操作、文件访问和网络通信等[20]。通过安装钩子拦截这些 API 调用，分析者不仅可以观察到哪些系统资源被访问，还能了解访问的具体参数和结果[21]。系统钩子的优势在于其较高的效率，然而，这种方法可能无法提供像单步跟踪那样的详细执行细节，并有可能遗漏一些非标准 API 的调用或直接执行的机器码，如图 5 所示。

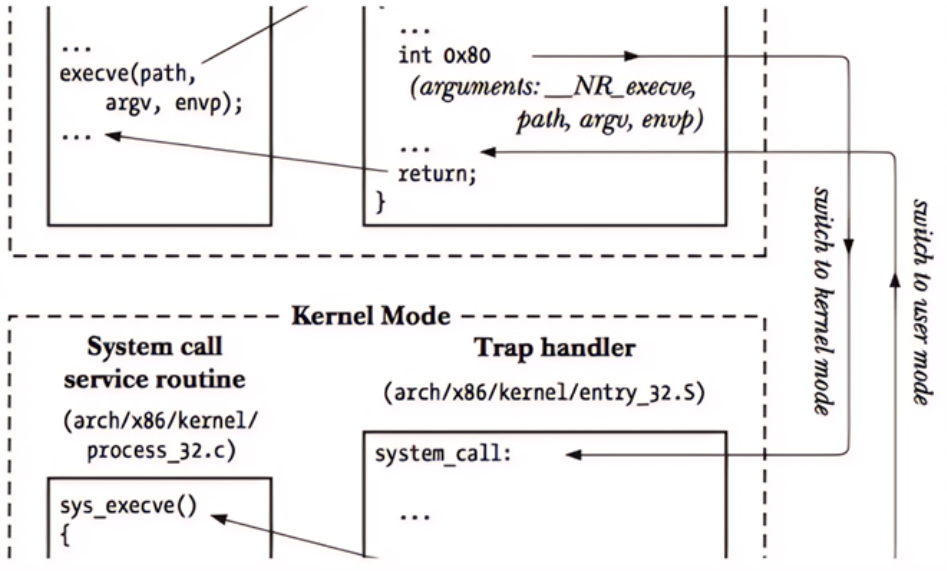

图 5: 内核 hook

4 实验结果与分析

4.1 实验设置

4.1.1 数据集

本部分数据主要来源于公开漏洞库（如 CVE、CNVD）以及多个厂商披露的 IoT 设备安全公告，结合 NVD 中筛选出的近三年涉及嵌入式设备的高危漏洞。为便于模型处理，首先对每条漏洞记录进行结构化转换，提取包括漏洞描述、影响组件、攻击路径、触发条件、利用代码片段等语义字段，并通过人工标注或规则引擎划分漏洞类型标签。最终构建出包含约 3000 条结构化漏洞样本的数据集，覆盖缓冲区溢出、命令注入、认证绕过、路径遍历等八类典型漏洞类型。

4.1.2 评估指标

为全面评估模型性能，实验从以下三个维度设计评估指标。

1. 采用 Token 准确率，如公式 (3)，衡量 LLM 提取关键特征与人工标注的匹配程度。

$$\text{Token Accuracy} = \frac{\sum_{i=1}^{N} I(\hat{y}_i = y_i)}{N} \tag{3}$$

其中，$\hat{y}_i$ 表示模型预测的第 i 个标记，y_i 表示人工标注的第 i 个标记，N 表示总标记数。$I(\cdot)$ 表示指示函数（预测正确时为 1，否则为 0）。

2. 通过精确率、召回率和 F1 分数综合评估分类性能。

精确率（Precision）：衡量模型预测为攻击的样本中真实攻击的比例，如公式（4）

所示。

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

召回率（Recall）：衡量真实攻击样本中被模型正确识别分类的比例，如公式（5）所示。

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

F1 分数：综合精确率与召回率的调和平均值，用于平衡两者在类别不平衡场景下的表现，如公式（6）所示。

$$F1\text{-}score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

其中，TP 表示真实攻击样本被正确分类为攻击的数量，FP 表示正常流量被误判为攻击的数量，FN 表示真实攻击样本被错误分类为正常的数量。精确率高表明模型的误报率低，召回率高表明模型的漏报率低。

从模型在特征分类任务中的准确性与处理效率上，本文实验采用总体分类准确率、平均分类准确率和处理时间来进行评估。

总体分类准确率：所有测试样本中正确分类的样本占比，如公式（7）所示。

$$\text{总体分类准确率} = \frac{\text{正确分类样本数}}{\text{总测试样本数}} \times 100\% \tag{7}$$

平均分类准确率：所有攻击类别准确率的算术平均值，用于避免类别不平衡对总体指标的影响，如公式（8）所示。

$$\text{平均分类准确率} = \frac{1}{N} \sum_{i=1}^{N} \text{第} i \text{ 类准确率} \tag{8}$$

其中 N 为攻击类别总数。

处理时间：单条攻击流量样本从特征输入到分类完成的平均耗时（单位：毫秒/样本），用于评估模型的计算效率。

4.2 结果分析

4.2.1 语义解析能力评估

本实验通过设定标准标签体系，对输入语句中嵌套结构、指代关系及专业术语的解析准确性进行评估，涵盖实体抽取、关系识别与语义归类三个维度。评估过程中，引入多组典型攻击描述样本，比较不同模型在复杂句式与模糊表达下的解析表现，从而衡量其在实际威胁情报处理任务中的应用价值。

具体实验中，本实验选取了 GPT-3[20]、GPT-4[21]、Claude 2.0[?]、Bard[?] 等大语言模型进行对比测试。针对测试集中的每一条攻击流量样本，依照表 1 中设定的提示模板，将其输入至各类大语言模型，获取模型生成的攻击特征摘要。随后，将模型生成的摘要

与人工标注的特征标签进行对齐，按语义维度对输出结果中的 Token 匹配情况进行统计，最终计算各维度上的 Token 级准确率指标，用于量化不同模型在攻击语义抽取任务中的性能表现。实验结果如表 1 所示。

表 1: 不同大语言模型 Token 准确率对比

大语言模型	攻击向量（%）	协议异常（%）	上下文（%）	总体 Token 准确率（%）
GPT-3	87.1	85.4	83.2	87.1
GPT-4	93.3	90.9	90.5	91.2
Claude 2.0	86.2	82.1	81.8	85.5
Bard	89.8	88.4	87.6	88.9

各类大语言模型在处理攻击语义信息时表现存在显著差异。GPT-4 在多项关键字段的识别上准确率最高，尤其在捕捉攻击路径和协议层异常方面展现出较强的解析能力，体现出其大规模参数模型在复杂语义理解任务中的优势。相较之下，Claude 2.0 在整体提取效果上表现较弱，面对新型攻击模式的适应性仍有待提升。Bard 在攻击路径识别方面表现尚可，但其输出一致性与稳定性略显不足，影响了整体应用的可靠性。

4.2.2 分类性能评估

为评估模型的分类效果，本实验基于混合数据集，首先通过训练集完成模型参数优化与语义关联学习，随后在测试集上得出攻击类型的分类性能指标如图 5 所示。

图 6: 攻击类型分类性能指标

实验证明，模型在识别结构清晰、语义稳定的攻击类型时表现稳定，如命令注入、DNS over HTTPS 和 DDoS 等类型均实现了超过 90% 的识别准确率和召回率，展现出良好的泛化能力。而对于默认口令尝试和 Smuggled ICMP，这类表现形式高度多样、上下文依赖性强的攻击，召回率相对偏低，易因术语不一致或语义模糊导致分类失准。

为提升对新型威胁的响应能力，系统还引入了相似度动态调节机制，结合实验确定了最优检索阈值，以增强语料库在未知标签下的泛化标注能力。在验证集上对阈值范围（0.6-0.9）进行网格搜索后，得出不同的相似度阈值下，模型在测试集上的精确率（Precision）、召回率 (Recall) 以及 F1 分数的变化如图 6 所示。

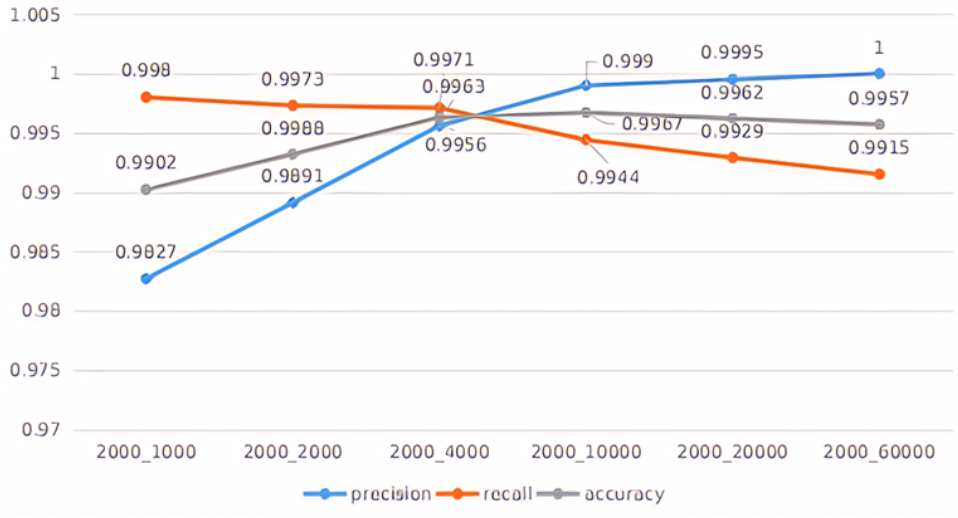

图 7: 相似度阈值对分类性能影响

模型在相似度阈值为 0.75 时取得了最优表现，F1 显示出较好的识别平衡性。进一步分析发现，若设置阈值偏低，系统倾向于捕捉更多边界模糊样本，尽管召回能力增强，却引发明显的误报增长；而当阈值调高，虽然误判减少，但对变异攻击样本的识别能力大幅下降。最终选定的 0.75 阈值在过滤干扰项与保持检测敏感性之间达成了良好折中，不仅提升了识别质量，也为模型的增量学习与特征扩展提供了高置信输入。

4.2.3 分类准确率评估

为验证本模型在分类准确率与处理时间上的优越性，本节选取了循环神经网络（Recurrent Neural Network，RNN）、长短时记忆网络（Long Short-Term Memory，LSTM）和自编码器（Autoencoder，AE）+LSTM 与本文方法进行对比试验，以评估各方法在攻击特征分类任务中的分类准确率、处理时间等性能指标。各方法在攻击特征分类任务中的性能对比如表 2 所示。

表 2: 不同方法在攻击特征分类任务中的准确率与处理时间对比

方法	分类准确率（%）					总体分类准确率（%）	平均分类准确率（%）	处理时间（ms/样本）
	命令注入变种	DNS-DoH	DDoS	默认口令尝试	Smuggled ICMP			
RF	88.9	85.2	90.0	82.1	76.3	88.1	87.4	0.6
RNN	90.1	86.5	91.6	82.9	77.9	89.3	88.7	4.5
LSTM	91.3	87.7	92.4	84.4	79.2	90.5	89.0	4.9
AE+LSTM	92.5	89.1	93.5	85.7	80.1	90.9	90.2	5.8
本文模型	**95.9**	**92.0**	**94.8**	**85.1**	**84.0**	**94.5**	**93.7**	**1.9**

本研究提出的攻击特征识别方法在整体分类性能上表现优越，其中攻击类别识别

准确率达到 94.5%，平均准确率为 93.7%，明显优于传统机器学习算法和常规深度网络架构。在命令注入变体和大规模拒绝服务攻击等结构规范、语义清晰的样本上，模型识别准确率分别为 95.9% 和 94.0%，凸显了其在语义解析与模式抽取方面的能力。对于默认口令类攻击与 Smuggled ICMP 等上下文复杂、表达差异较大的类别，尽管存在一定挑战，但准确率仍较传统方法有显著提升，说明模型在处理复杂变异样本、尤其是 IoT 领域中的新型威胁方面具备较强的泛化能力与适应潜力。

5 结束语

本文详细研究了 VxWorks 操作系统中的整数溢出漏洞（CVE-2015-7599），通过静态和动态分析确定了该漏洞的具体成因，并深入探讨了漏洞在实际工控环境中，尤其是 PLC 设备上的利用方式。通过静态分析详细揭示了 CVE-2015-7599 漏洞 VxWorks 操作系统中这一整数溢出漏洞的内部机制，并通过动态调试方法展示了漏洞的触发过程以及攻击者可能利用该漏洞进行攻击的方式。实现了针对 PLC 设备的漏洞利用程序，包括信息探测、读取配置信息、远程控制等多个模块，这些程序在实验环境中进行了测试，成功验证了其可行性与效果。通过这些实验，本研究验证了漏洞的实际影响，确认了攻击者能够利用该漏洞对 PLC 设备进行广泛的控制。

在总结本文的研究成果的同时，我们也必须承认目前对于工控系统安全性的挑战仍然严峻。尽管本研究提出了较为有效的漏洞分析方法和利用技术，但攻防技术的迅速发展要求我们不断更新和改进防御手段。

参考文献

[1] 任秋洁, 韩英. 面向 VxWorks 系统的嵌入式安全研究 [J]. 电脑知识与技术, 2020, 16 (22): 26-27.

[2] 孙润春. 基于 VxWorks6.8 操作系统的关键安全防护技术的研究 [D]. 北京邮电大学,2018.

[3] 王禹超. 基于二进制静态插桩的 VxWorks 固件网络协议模糊测试技术 [D]. 北京邮电大学,2022.

[4] 由忠宇. 基于 ARM Cortex 应用处理器和 VxWorks 操作系统的车载显示平台设计与实现 [D]. 北京邮电大学,2018.

[5] 尹鹏. 基于 VxWorks 嵌入式操作系统的网络数据通信研究 [D]. 西安电子科技大学,2017.

[6] 武华, 刘军伟. 基于 VxWorks 的多任务程序设计 [J]. 微机发展,2011(9):163-166.

[7] 李大山. 嵌入式网络存储服务器的优化研究 [D]. 上海交通大学,2010.

[8] 陈和平, 李国仿, 高丽. 基于 RSA 算法的 RPC 认证机制研究与实现 [J]. 武汉科技大学学报 (自然科学版),2004(04):409-411.

[9] 王玉婷. 基于实证的设计模式检测的研究 [D]. 安徽大学,2019.
[10] 张琦. 程序内存安全性动态分析和模糊测试技术 [D]. 南京航空航天大学,2022.
[11] 李伟文. 基于动态分析的模糊测试入口程序定位方法研究 [D]. 广州大学,2023.
[12] 尹小康, 蔡瑞杰, 杨启超, 等. 基于静态和动态混合分析的内存拷贝类函数识别 [J/OL]. 软件学报:1-24[2024-06-06].
[13] 柳淑玉. 缓冲区溢出漏洞挖掘技术研究 [D]. 武汉纺织大学,2012.
[14] 王禾. 跳板攻击场景下的嵌入式系统的攻击路径分析 [D]. 西安电子科技大学. 2021
[15] Zhang, Yongguang and Wenke Lee. "Intrusion detection in wireless ad-hoc networks." ACM/IEEE International Conference on Mobile Computing and Networking (2000).
[16] Bruce Schneier and Phil Sutherland. 1995. Applied Cryptography: Protocols, Algorithms, and Source Code in C (2nd. ed.). John Wiley & Sons, Inc., USA.
[17] Ferguson N, Kelsey J, Lucks S, et al. Improved cryptanalysis of Rijndael[C]//Fast Software Encryption: 7th International Workshop, FSE 2000 New York, NY, USA, April 10–12, 2000 Proceedings 7. Springer Berlin Heidelberg, 2001: 213-230.
[18] 龚广, 李舟军, 忽朝俭, 等.Windows 内核级 Rootkits 隐藏技术的研究 [J]. 计算机科学,2010,37(04):59-62.
[19] 李馥娟, 王群.Rootkit 攻防机制与实现方法 [J]. 电信科学,2018,34(12):33-45.
[20] 杨彦, 黄皓.Windows Rootkit 隐藏技术研究 [J]. 计算机工程,2008(12):152-153+156.
[21] 孙飞. 基于内核对象管理器的钩子型 Rootkit 检测技术研究 [D]. 哈尔滨工业大学,2021.
[22] Liu J, Guangyuan F U, Hailong L I. Research on VxWorks System Vulnerability Mining and Security Assessment[C]//2nd International Conference on Artificial Intelligence and Engineering Applications (AIEA 2017). 2017: 634-643.
[23] Zheng W, Zhou Y, Wang B. Design and Implementation of VxWorks System Vulnerability Mining Framework Based on Dynamic Symbol Execution[C]//Proceedings of the 9th International Conference on Computer Engineering and Networks. Springer Singapore, 2021: 801-811.
[24] Zhang Z, Lv Z, Mo J, et al. Vulnerabilities analysis and solution of VxWorks[C]//2nd International Conference on Teaching and Computational Science. Atlantis Press, 2014: 94-97.