FSA (2000)

Sekar

A fast automaton-based method for detecting anomalous program behaviors

有限状态机计算十分复杂，sekar改进了FSA在恶意软件识别中的方案，使得计算复杂度降低，可以更快的学习数据，并执行更准确的检测。

PAYL（IDS）（2004）

Anomalous Payload-Based Network Intrusion Detection

将正常发往某个主机特定端口的数据搜集统计，计算字符频率，在异常检测时，收集数据包负载，统计字符频率，比较这两个统计分布，计算马氏距离，如果差别比较大，判定为恶意流量。之后进行一些流量的处理。优点就是，针对首次出现的病毒，具有一定的防御能力（首次出现的恶意攻击，攻击负载之前未出现过，所以其概率分布与正常流量的概率分布可能差别较大。（加密流量可能不能识别？）

Information Flow（2009）

**Effective** and **Efficient Malware Detection at the End Host**

**没有给出特定的系统名称，提出了基于信息流识别软件恶意行为的概念**

一种基于信息流的恶意软件检测方法。将正常软件日常行为，与操作系统之间的信息流采集整理，分片，检测时将待测软件与系统之间的信息流做比较，相差较多，表示恶意行为。

一种恶意代码行为特征提取方法

<https://patents.google.com/patent/CN102054149A/zh>

**恶意软件识别与IDS概览**

IDS可分为两类：一是基于主机的，二是基于网络的

基于主机的入侵检测系统(HIDS)在网络的每个主机上部署本地代理。HIDS使用本地代理和应用程序日志或原始系统调用作为数据源来检测恶意进程、修改关键系统配置文件(例如注册表项)、权限升级和任何其他违反系统策略的未经授权的操作。

HIDS所提供的保护是不够的，因为它们仅限于一个主机上。

NIDS监视和分析数据包或流级别上的网络流量，并试图检测异常，例如未经授权的访问或DDoS攻击。在数据包级别上，IDS执行所谓的深度分组检查(DPI)，它同时分析每个包的报头和有效荷载。虽然检测数据包的有效载荷是非常有用的，但在当今高速通信网络中，这种方法不仅耗时、效率低，而且计算成本也很高。

而且更重要的一点，在对包加密，例如暗网和VPN技术的使用，对payload的分析近乎是不可能的。NIDS的主要缺点之一是网络的可扩展性，即随着网络的规模和复杂性的变化，NID的调整能力。

根据其检测机制，可以将IDS分类为：

(一)误用或基于签名

(二)基于异常或行为

(三)混合。

基于误用或签名的系统维护与已知攻击相对应的预定义签名(模式)数据库，并通过将它们与审计数据流进行比较来执行检测。有许多基于签名的开源idss在企业领域中得到了广泛的应用，例如Snort、Bro和Suricata。此外，现有的各种安全解决方案包括已知供应商提供的网络入侵检测，例如AlienVault提供的统一安全管理(USM)、Cisco提供的下一代iPS(Ngips)和Fireye的network security。但这种检测方式只能检测已知攻击，严重依赖于特征库。并且极易容易绕过。

异常检测是指在数据集中识别不符合“正常”行为的实例的问题。如果与“正常”行为的偏差超过了预先定义的或动态计算的阈值，则行为被视为异常或异常值。异常检测广泛应用于各种应用，如信用卡、保险、医疗等欺诈检测，网络安全入侵检测，安全关键系统故障检测，敌人活动军事监控等。基于异常的IDS不依赖于先前定义的模式，但它们的目标是对正常行为/流量进行建模，以检测异常，从而能够同时检测已知和未知的攻击。这样做的代价要不断的进行模型的调整。

文献中有很多研究试图将上述技术结合起来，以继承两者的优点，提高检测率，最大限度地减少假阳性率。Barbara等人[1]提出了一种最著名的混合系统，称为审计数据分析和挖掘(ADAM)。ADAM有两个检测阶段：首先对基于异常的阶段使用关联规则挖掘，然后通过误用模块将可疑连接划分为正常、已知和未知的攻击。Kim等人[2]将用于误用模块的C4.5决策树与多个单类支持向量机(SVMs)相结合，对正常行为进行建模。同样，depren等人[3]对异常模块和C4.5决策使用自组织映射(SOM)。

基于异常的IDS(Anomaly-based IDS)根据其使用的方法可分为以下几类：统计、监督(分类)、无监督(聚类和孤立点检测)、软计算、基于知识的学习和组合。

在有监督IDS中，对模型进行训练，以便从样本(即标签数据)中学习。当引入新样本时，分类器尝试将其分配给一个预定义类。在基于网络的入侵检测任务中，采用了决策树(c4.5)、svm、k近邻、贝叶斯分类器、神经网络等几种分类算法。

然而，当数据集中存在未知攻击时，大多数分类模型都无法检测到它们，而且众多模型中svm得到了最好的结果。此外，基于分类的系统与基于签名的系统有类似的缺点，即它们需要定期接受训练以保持高的检测率。但实际情况中这是不可行的，因为获得标签数据极其困难，特别是这些规则。此外，即使标签数据确实存在，也不确定它们是否包括所有新的攻击。

无监督异常检测(也称为基于孤立点的检测)使用聚类技术来识别给定数据集中可能存在的恶意实例，而不需要任何先验知识。聚类的目的是将有限的未标记数据集分割成有限的离散的“自然”隐藏数据结构集。换句话说，聚类算法的目标是将给定的数据划分成组(簇)，这些组(集群)实现了内部相似和外部的高度不同，而不需要任何先验知识。所有聚类方法都基于以下假设。首先，数据集中的正常实例的数量远远超过异常的数量。第二，异常样本和正常样本是有一定区分度的。但是这种方法的误报率通常很高。特征提取或选择是无监督异常检测的重要步骤之一。

在结果度量方面，一般有以下度量方法：

1.混淆矩阵，是将实际结果与预测结果之间的关系可视化的一种方法。它主要用于监督学习来评价分类器的预测精度。

2.recall值和precision值

3.准确性考虑了正负两方面的因素，并被定义为正确分类的样本与实例总数的比率。

4.敏感性，也称为真阳性率(TPR)，是正确分类的阳性样本的比例。相反，特异性或真阴性率(TNR)衡量的是被正确归类为阴性的实例的比例。同样，假阳性率(FPR)表示错误识别为异常的样本比例

5.接收系统工作特性(Roc)是信号处理理论中用于对不同参数设置下的TPR进行可视化的一种技术。它描述了利益(真正)和成本(假阳性)之间的相对权衡。

在入侵检测中，攻击被分为以下几类：

1.DDoS， 2.探测（scanning）。3.提权（缓冲区溢出）4.远程攻击（ssh 爆破）

2.具体实施的方法描述

2.1 特征的选取

特征选择过程有两个主要组成部分：搜索策略和评价准则。选择的搜索策略负责选择特征作为最优子集的一部分。评价标准为每个特征分配一个分数。如果这个分数超过了一个阈值，那么它就被认为是关联的。

特征方法可分为两大类：filters和wrappers。filiters不考虑分类技术，而是利用统计和信息论方法，如互信息、信息增益、相关系数和信息熵等，对提出的特征进行评分。因此，滤波器是一种快速、简单的方法。相反，wrappers(使用基于正在使用的检测算法的预测模型对候选子集进行评估。在每次迭代中，训练集上的分类器使用一个特征子集，根据结果，特征要么被接受，要么被拒绝。虽然Wrappers将检测算法考虑在内，并因此产生了一个与特定算法和id相适应的子集，但它们会导致过度拟合，而且计算量会很大，特别是对于高维的网络数据而言。

在无监督的学习场景中选择特征被认为是一个困难得多的问题，因为缺乏指导搜索相关信息的类标签。在这种情况下，最常见的标准是集群的质量、簇内和簇间距离。

常用的FS技术，有以下6种：信息增益（IG）， 增益率（GR），主成分分析（PCA）, 基于相关性特征选择（CBF），卡方和基于一致性搜索（CBC）

2.2 无监督和混合IDS技术实现

监督学习试图在不事先知情的情况下区分恶意通信量和正常流量。本章节介绍和比较了过去5年公布的无监督和混合(监督和无监督)方法。得到如下结论：

1）网络流量数据是高维的。具体而言，大多数数据集对每个数据点具有19至50个特征。此外，随着技术的发展，这一数字可能会进一步增加。因此，利用整个特征空间进行聚类不仅耗时，而且检测率低。因此，如上一节所述，特征选择至关重要。

2）具有检测任意形状簇的能力的聚类算法，如dbscan，比只能检测圆形集群的算法性能更好。

3）检测方法的并行化可以减少计算时间，实现实时检测。因此，可以利用高性能的云计算技术来达到更好的时间性能和分配处理工作量。

4）使用主机数据可以提高检测率，但必须考虑从大规模网络中实时提取这些数据是否可行。

5）初始化和调整系统参数的需要是阻止这些技术在工业中应用的主要缺点之一。因此，基于密度的方法，如dbcan，需要较少的参数化，似乎最适合在实际网络中部署。

6）无监督和监督技术的结合可能会导致较高的检测率和较低的假阳性率。

7）u2r和r2L攻击是最难检测的，因为它们类似于正常的通信量。子聚类、特征选择(和使用不同的网络分辨率似乎可以提高这些攻击类的检测率。此外，神经网络和无监督方法在这两种低频攻击中的结合效果很好。

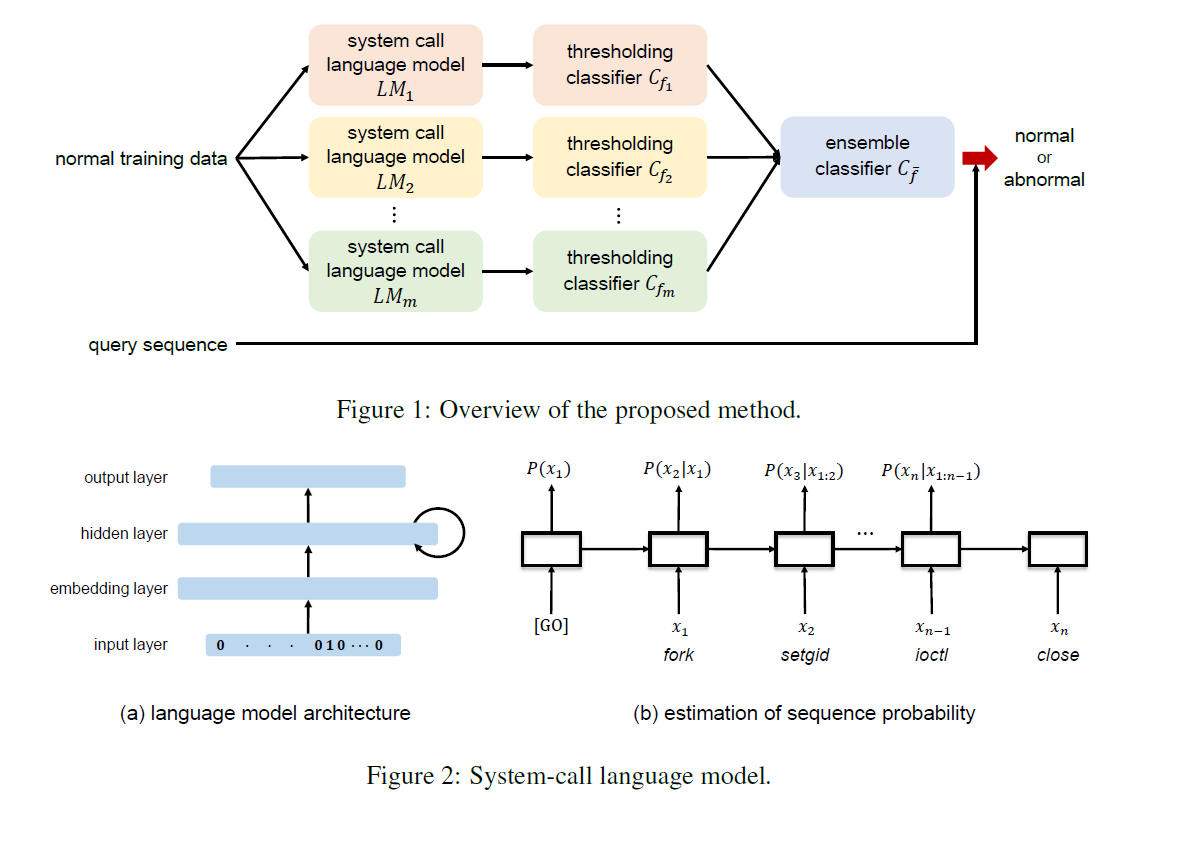
我们可以借鉴IDS中的一些先验知识，比如特征提取过程，以及深度学习模型在IDS中的应用。恶意软件检测技术的主要难点在特征选择，选择何种特征使检测具有较高准确性的同时，误报率比较低。传统的特征提取方法主要有

1. 指令序列  
   使用简单的指令序列作为特征进行匹配，同一恶意代码的变种，需要用不同的指令序列来表示，特征数据库大，容易被混淆技术绕过
2. 提取API调用序列  
   使用恶意软件与操作系统交互的API调用序列作为特征，记录指令序列后使用序列匹配，或有限状态机的方法，判断恶意行为，恶意代码采用充排顺序无关的系统调用，或插入无关API调用的方法可以躲避此类型的检测
3. 提取系统消息和事件  
   使用系统状态变化作为恶意软件行为特征，监控软件和系统的交互， 记录系统消息和事件序列作为特征，由于此方法仍然依赖于恶意软件调用序列的顺序，容易被绕过
4. 使用调用序列或操作码的频率分布，恶意软件通过垃圾代码插入，使得此类特征匹配无效
5. 使用差异子图  
   提取恶意软件与正常软件的行为，构成CFG控制流图，将CFG进行对比，获取差异子图作为特征由于有些行为正常代码和恶意代码均会产生，只是在特定上下文中 或次数不同而界定为恶意，此时只提取差异子图不足以描述恶意代码特征；且如何选用作 比较的正常代码会对准确性及特征大小产生影响

目前存在的问题：

分析及跟踪恶意代码已提取特征的系统与恶意软件具有相同的权限，复杂的隐藏技术可以绕过分析使特征无法提取；未能全面提取数据依赖和控制依赖信息表示特征，特征准确性不高；特征提取大多针对单一的恶意样本，对利用混淆技术产生的变种无法检测，特征适应性不强，轻微混淆技术产生的变种即产生新的特征，特征库庞大。

在最近有关恶意软件检测的论文中，使用机器学习模型根据系统调用序列来进行恶意软件分类的应用占多数。系统调用序列在传统的检测方法中，具有被混淆代码绕过的风险，但是神经网络和机器学习算法或许可以规避这一缺陷（使用处理时间序列的方式处理调用序列）。

在16年的一篇论文中，G Kim, H Yi, J Lee, Y Paek, S Yoon等人提出了一种将调用序列当作一种应用程序和操作系统交互的语言，使用自然语言处理的技术来处理恶意软件检测的方法。其系统结构如下图所示  


Figure表示整个模型的结构，使用多个语言模型，每个语言模型表示某种正常操作模式，当输入的序列是正常操作，该模型输出会以较高概率输出一个正常的分类。再使用一个权重矩阵和激活函数（ReLU）来将这若干个分类器组合为一个分类器，最终获得准确率上的提升。论文中表示模型可以获得较好的性能。因为语言模型中使用LSTM（改进的RNN，可以记忆更长久的信息，并且没有梯度爆炸）作为组成隐藏层的神经元，所以可以一定程度上，解决混淆代码导致检测失败的问题。