〈一般研究課題〉　機械学習ベースNIDS分散処理フレームワークの
実環境における実用性評価
助 成 研 究 者　豊橋技術科学大学　　中村　純哉

機械学習ベースNIDS分散処理フレームワークの実環境における実用性評価

中村　純哉
（豊橋技術科学大学）

Evaluation of a distributed processing framework for machine learning-based NIDS on real environments

Junya Nakamura
(Toyohashi University of Technology)

Abstract：

The Internet is widely spread today as an information infrastructure indispensable for our daily life and business. However, the threat of cyber-attacks increases year by year, and the technology that can make the Internet more secure is required. A Network-based Intrusion Detection System (NIDS) is a system to detect intrusion attacks from network traffic and is widely used as a defense method of cyber-attacks. However, its detection accuracy and throughput for unknown attacks are limited. We have proposed a distributed processing framework for building NIDS based on machine learning called MLNIDS to solve these problems. This framework is expected to have high practicality and processing performance because all processes are built on distributed processing frameworks capable of scaling out. However, since this framework was evaluated only in the form of omitting the classification process by machine learning, the practical performance of the whole framework, including the classification process, is unknown. In this paper, we implement the classification process by machine learning on the framework and evaluate the performance using real network traffic.

1. はじめに

インターネットは，我々の日常生活・ビジネスになくてはならない情報インフラストラクチャと

して，今日広く普及している．しかしながら，サイバー攻撃の脅威も年々増加しており，安全にインターネットを利用するための技術が求められている．ネットワーク型侵入検知システム(NIDS)はネットワークトラフィックに含まれる侵入攻撃を検知するシステムであり，サイバー攻撃の防御法として現在広く利用されているが，未知の攻撃に対する検知精度や処理能力に制約がある．我々の研究グループはこれらの問題を解決するため，機械学習に基づいたNIDSを構築するための分散処理フレームワークMLNIDS(Machine Learning-based Network Intrusion Detection System)を提案した［1］．このフレームワークは，通信トラフィックからの特徴量抽出や悪性トラフィックの検知などすべての処理がスケールアウト可能な分散処理フレームワーク上に構築されていることから高い実用性と処理性能を持つと期待される．しかしながら，これまで同フレームワークの評価は機械学習による分類処理部分を省略した形でしか行われておらず，分類処理も含んだフレームワーク全体の実用性評価は未実施である．

本研究では，機械学習による分類処理を同フレームワーク上に実装し，実環境のトラフィックを用いた性能評価を行うことにより，提案フレームワークの実用性とスケーラビリティを明らかにする．それにより，我々の生活環境を安全で便利なものにするために不可欠な情報基盤技術の実現を目指す．

2. 提案フレームワークMLNIDSの概要

2.1. 処理の流れ

ここでは，提案フレームワークMLNIDSにおける処理の流れ(図 1)の概要を説明する．処理や特徴量の詳細については，［1］を参照されたい．まずMLNIDSは，ネットワークトラフィックを元にセッション情報を構築し，セッションに関する基本特徴量を取得する．次に，基本特徴量と通信元および通信先に関する時系列情報から，ホストベース特徴量を計算し，MLNIDSで利用できるすべての特徴量を含んだ完全な特徴量を生成する．続いて，完全な特徴量を元に，機械学習によってセッションを分類し，通信の種類が良性を悪性かであるかを判定する．完全な特徴量と分類結果は，永続データストアに保存する．この情報は，本フレームワークによって提供される要約ビューで利用されるほか，ネットワーク管理者が自組織の通信状況を詳細に分析する目的でも利用できる．

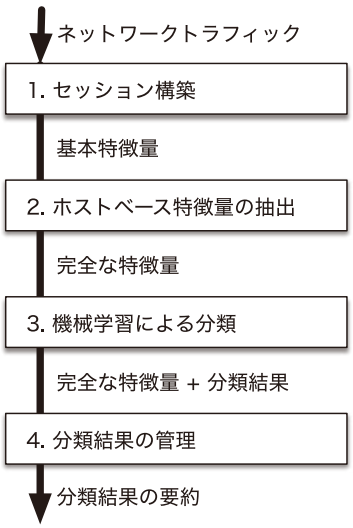

図 1　MLNIDSの処理の流れ

2.2. システム構成

図 2に，MLNIDSのシステム構成を示す．MLNIDSは2.1節の処理を実現するため，複数のサブシステムから構成される．それぞれのサブシステムはスケールアウト型のアーキテクチャを採用しており，処理性能が不足する場合は，各サブシステムを構成するノードを増やすことで，処理性能を向上できるという特徴を持つ．

本フレームワークでは，主となるデータ処理基盤としてApache Kafka［2］を用いる．Apache

Kafkaは分散イベントストリーミングプラットフォームであり，様々なシステムと連携した高速なデータ処理が可能である．図 2に示すように，MLNIDSにおける様々なデータ処理は，Kafka Brokerを中心に各サブシステムが連携することで実現される．

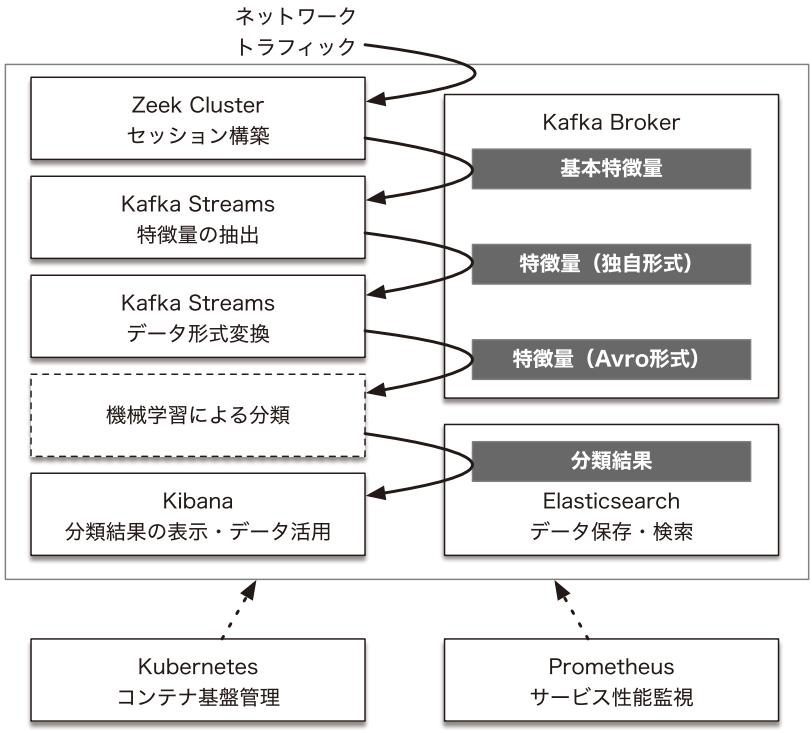

図 2　MLNIDSフレームワークのシステム構成

MLNIDSは，セッションを構築するため，Zeek Network Security Monitor［3］を利用する．Zeekは，20年を超える長い歴史を持つネットワークセキュリティ監視ツールであり，オープンソースで公開されている．ネットワークトラフィックをセッション単位で管理する．Zeek Clusterと呼ばれる負荷分散の仕組みを備える．Zeek Clusterでは，複数のプロセスでZeek IDSを並列実行し，トラフィックをそれらのプロセスに分散する．これにより，マルチコアを有効活用でき，容易に処理性能を高めることができる．本フレームワークのZeekは，ネットワークインターフェイスからトラフィックをキャプチャし，得られたセッション情報を基本特徴量としてKafka Brokerに書き込む．

Zeek IDSがKafka Brokerに基本特徴量を書き込むと，Kafka Streams［4］がその情報から完全な特徴量を計算し，データ形式を変換する．Kafka StreamsはApache Kafkaの一部として提供されるクライアントライブラリであり，Kafka Brokerをデータ入力元・出力先とするアプリケーションを容易に実現するために必要となる様々な機能を備える．

本フレームワークの利用者は，Kafka Brokerをデータ入力元，Elasticsearch［5］をデータ出力先とした機械学習による分類器を実装し，ネットワークトラフィックの悪性判定を行う．分類器の実現方法は，フレームワークの利用目的に応じて無数の選択肢があることから，提案フレームワークには含まれていない．

完全な特徴量および分類器による分類結果はElasticsearchに保存され，利用者による様々な分析に利用される．Elasticsearchは分散型の検索・分析エンジンであり，高い利便性と柔軟性を備える．また，データ可視化ツールKibana［6］と連携することで，本フレームワークによる分類結果をグラフィカルに表示できる．

本フレームワークのサブシステムはすべてコンテナとして構築されており，代表的なコンテナ管理基盤Kubernetes［7］上で動く．そのため，各サブシステムは高い可搬性を持ち，オンプレミス環境やパブリッククラウドなど様々な環境に容易に配置できる．加えて，ノード数などは

Kubernetesの機能を使ってパラメータ化されており，コマンドで瞬時に増減できる．各サブシステムやコンテナが動くサーバの性能はサービス監視システムPrometheus［8］によって計測されており，フレームワーク全体の処理能力を容易に把握できる．

3. 評価実験

ここでは，提案フレームワークMLNIDSに，機械学習による分類器を組み込み，実トラフィックにおける性能を評価する．

3.1. 機械学習による分類器の実装

実装する機械学習による分類器の構成を，図 3に示す．Kafka Brokerからのデータ取得およびElasticsearchへのデータ出力には，Logstash［9］を用いる．Logstashは，Elastic Stackの一部として開発されているデータ処理ソフトウェアで，Elasticsearchと高い親和性を持つ．プラグインの導入によって様々なシステムと連携でき，Apache Kafkaにも対応することから，選定した．分類処理は，Pythonの機械学習ライブラリscikit-learn［10］によって実現する．LogstashからPythonで作られた分類処理を直接利用するのは困難なため，Flask［11］を用いて分類処理をWebアプリケーションとして実装し，LogstashはWeb APIによって分類処理を実行する．学習モデルには，Random Forestを使用し，決定木の数は260，最大深さは60とした．

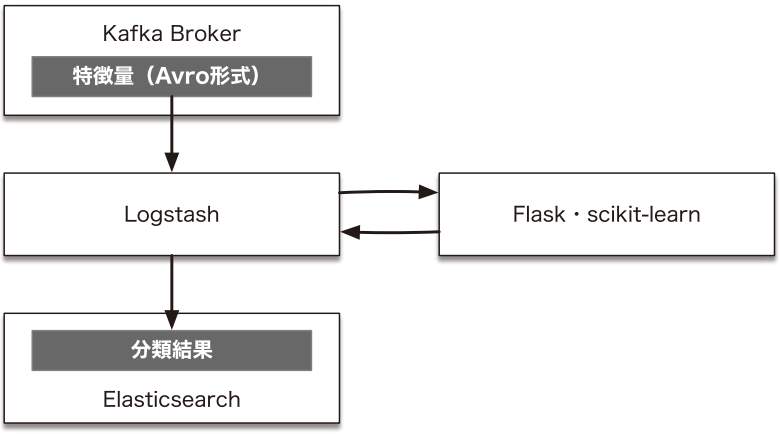

図 3　分類器の構成

3.2. 実験条件

評価実験で用いた各サーバの仕様を表 1に示す．すべてのホストは，NETGEAR社のネットワークスイッチXS716Tによって10Gbpsで接続されている．評価実験で用いた各サブシステムのバージョンを表 2に示す．各サブシステムは，表 2の「配置サーバ」列に示すように配置する．

分類器の学習と評価には，2020年1月22日にtcpdump［12］によってキャプチャした，所属組織とSINET間のネットワークトラフィックを用いる．キャプチャデータからセッション情報を取得した後，Snort［13］によって良性と悪性を分類し，それぞれのサンプル数がほぼ同数となるように調整した．合計サンプル数は，1,309,156である．

表 1　各サーバの仕様

サーバ番号	CPU	コア数	クロック	メモリ	SSD	NIC	通信速度
1〜8	Xeon E3-1220v5	4	3.0 GHz	16 GB	1 TB	Intel X550-T1	10 Gbps
9〜10	Xeon E-2136	6	3.3 GHz	16 GB	1 TB	Intel X540-T2	10 Gbps
11	Xeon Silver 4208	8	2.1 GHz	32 GB	1 TB	Intel X722	10 Gbps
12	Xeon Silver 4214R	12	2.4 GHz	16 GB	1 TB	Intel X722	10 Gbps

表 2　サブシステムのバージョンと配置サーバ

サブシステム名	バージョン	配置サーバ
Zeek	4.0.4	12
Kafka Broker	2.8.1	1, 2
Kafka Streams	2.0.0	4, 5
Logstash	7.15.2	6〜8
scikit-learn	1.0.1	
Elasticsearch	7.15.2	9, 10
Kibana	7.15.2	9
Prometheus	0.52.0	11
Grafana	8.2.3	11
Kubernetes	1.22.2	全て

3.3. 分類精度

ここでは，実装した分類器の性能を評価する．データセットの75％を学習用としてランダムに抽出し，そのデータセットを用いて学習した．残る25％を評価用データセットとして用いる．

学習した分類器の精度，適合率，再現率，F値を表 3に，混同行列を表 4に示す．比較的高いF値となっており，良好な分類結果が得られている．

表 3　分類器の評価結果

	スコア
正解率 Accuracy	0.962
適合率 Precision	0.942
再現率 Recall	0.984
F 値 F-measure	0.963

表 4　混同行列

	良性と予測	悪性と予測
良性クラス	153,777	9,867
悪性クラス	2,462	161,183

3.4. 分類時間

次に，サンプルの分類にかかる時間を評価する．図 4に，scikit-learnによる分類処理にかかる時間，および，Logstashによるデータ入出力も含めた全体の処理時間を示す．処理時間の差は1.672 msと小さく，Web APIを経由することによるオーバヘッドはわずかである．一方で，絶対的な処理時間は長く，大量のセッションを処理できない可能性がある．この点については，用いる学習モデルの種類やパラメータとの関連性を，さらに詳しく調べる必要がある．

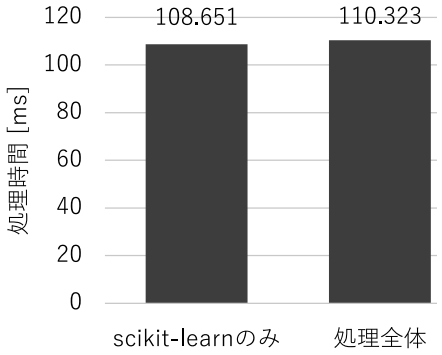

図 4　分類時間

参考文献

1. 多田竜之介，中村純哉，大村廉，小林良太郎．機械学習ベース NIDS 構築のための分散処理フレームワーク．情報処理学会論文誌．2019 Sep 15;60(9):1448-65.
2. Apache Software Foundation. Apache Kafka. https://kafka.apache.org/. 閲覧日: 2021年11月28日.
3. The Zeek Project. The Zeek Network Security Monitor. https://zeek.org/. 閲覧日: 2021年11月28日.
4. Apache Software Foundation. Kafka Streams. https://kafka.apache.org/documentation/streams/. 閲覧日: 2021年11月28日.
5. Elasticsearch B.V. Elasticsearch. https://www.elastic.co/elasticsearch/. 閲覧日: 2021年11月28日.
6. Elasticsearch B.V. Kibana. https://www.elastic.co/kibana/. 閲覧日: 2021年11月28日.
7. The Linux Foundation. Kubernetes. https://kubernetes.io. 閲覧日: 2021年11月28日.
8. The Linux Foundation. Prometheus. https://prometheus.io/. 閲覧日: 2021年11月28日.
9. Elasticsearch B.V. Logstash. https://www.elastic.co/jp/logstash/. 閲覧日: 2021年11月28日.
10. scikit-learn developers. scikit-learn: machine learning in Python. https://scikit-learn.org/stable/. 閲覧日: 2021年11月28日.
11. Pallets. Flask | The Pallets Projects. https://palletsprojects.com/p/flask/. 2021年11月28日
12. The Tcpdump Group. TCPDUMP/LIBPCAP public repository. https://www.tcpdump.org/. 閲覧日: 2021年11月28日.
13. Cisco Systems. Snort - Network Intrusion Detection & Prevention System. https://www.snort.org. 閲覧日: 2021年11月28日.