Vol.2021-DPS-186 No.59
Vol.2021-CSEC-92 No.59
2021/3/16

MQTT ブローカーのための免疫的攻撃検知の試作

岡本 剛 [1]

概要：スマートシティの実現に向けて，社会インフラや公共サービスのスマート化が進められているが，スマートシティに対するサイバー攻撃が懸念されている．その脅威の 1 つが未知の脆弱性に対するゼロデイ攻撃である．ゼロデイ攻撃によって発生したサービスを回復させるには，セキュリティアナリストなど専門家による対応が必要であり，数時間から数日のダウンタイムが発生する．特に防災サービスの妨害は人の生命を脅かすリスクがあるため，ダウンタイムの最小化などレジリエンスの強化が課題である．そこで本研究は著者らが開発してきた免疫的攻撃検知を防災サービスに応用することによって課題の解決を目指す．免疫的攻撃検知とは，からだの免疫のように，たとえ最初の攻撃を防止できなくても，その攻撃を学習することによって，二度目以降の類似の攻撃を防止する技術である．本研究では温度や水位などの観測データを MQTT プロトコルで収集する防災サービスの一部を想定して，免疫的攻撃検知を MQTT ブローカーに組み込み, MQTT ブローカーのレジリエンスを強化する. 本稿では, 免疫的攻撃検知を MQTT ブローカーに組む込む方法を提案し，免疫的攻撃検知のパラメータ値を示す．性能評価では，免疫的攻撃検知を組み込んだ MQTT ブローカーが脆弱性への攻撃に対して 99.72%の検出精度で攻撃メッセージを検知・防止できたことを示す．さらに，プロトタイプの実装により，メッセージサイズが 5,000 バイト以下なら，免疫的攻撃検知のオーバーヘッドはほとんどなかったことを示す．

キーワード：侵入検知，DoS 攻撃，MQTT，脆弱性，機械学習，免疫系 [**]

Prototype of Immune-Like Detection of Cyberattacks for an MQTT Broker

TAKESHI OKAMOTO[†1]

1. はじめに

スマートシティの実現に向けて，社会インフラや公共サービスのスマート化が進められている．スマートシティは従来の社会インフラや公共サービスを効率化し，環境にも配慮しながら，人々の生活の質を高められることが期待されている．その一方で，スマートシティに対するサイバー攻撃が懸念されている．その脅威の 1 つが未知の脆弱性に対するゼロデイ攻撃である．

制御フロー強制技術[1]など最新の技術が適用できればゼロデイ攻撃を防止できることがあるが，これらの技術にはサービスが停止するなどの副作用があり，サービスの妨害そのものを防止できない．ゼロデイ攻撃によって発生したサービス不能からサービスを回復させるには，セキュリティアナリストなど専門家による対応が必要であり，数時間から数日のダウンタイムが発生する．機械学習による検知技術[2]が世界中で研究されているが，検出精度が十分ではないため，専門家によるサポートが必要である．特に防災サービスの妨害は人の生命を脅かすリスクがあるため，サービスのダウンタイムを最小化するなどレジリエンスの強化が課題である．

著者らは，サービスの不能をトリガーにして攻撃を特定し, その攻撃を機械学習により学習する「免疫的攻撃検知」という技術を開発してきた[3]．免疫的攻撃検知は，たとえ最初の攻撃を防止できなくても，二度目以降の類似の攻撃に対してサービスを止めることなく未然に防止する技術である．従来の機械学習では多種多様な教師データを必要とするため，教師データの準備が実用上の障壁となることがある．それに対して，免疫的攻撃検知はサービス不能の状態から攻撃を特定して学習するため，事前に教師データで攻撃を学習する必要がない．著者が知る限り，このような免疫的攻撃検知に類似するアプローチは存在しない．

そこで本研究は免疫的攻撃検知によりスマート化された公共サービスのレジリエンスを強化することを目的とする．本研究ではセンサーノードから温度や水位などを MQTT プロトコルで収集する防災サービスの一部を想定する．本研究の目的に対して，本稿では，MQTT ブローカーに免疫的攻撃検知を実装する方法とその有効性と課題について明らかにする．

本稿の貢献は次の通りである．

- MQTT ブローカーの実装の 1 つである Mosquitto ブローカーに免疫的攻撃検知を組み込む方法を提案した．
- シミュレーション評価により，Mosquitto ブローカーのための免疫的攻撃検知のパラメータ値を示した．
- MQTT プロトコルバージョン 3 において，免疫的攻撃検知はサービス停止を引き起こす脆弱性への攻撃を

1 神奈川工科大学
Kanagawa Institute of Technology

Vol.2021-DPS-186 No.59
Vol.2021-CSEC-92 No.59
2021/3/16

99.72%の検出精度で検知・防止できること示した．

- プロトタイプを実装することによって，免疫的攻撃検知のオーバーヘッドは，MQTT メッセージサイズが 5,000 バイト以下なら，ほとんどないことを示した．

2. 関連研究

ファイアウォールなどのセキュリティアプライアンスでネットワークトラフィックを集中的に監視して，そのトラフィックに含まれる攻撃を機械学習によって検出する方法がある．最新の機械学習は，高精度かつ高速な分類と学習が可能になったが[4][5]，100%の検知は不可能であり，ゼロデイ攻撃がサービスを停止する可能性がある．サービスが停止したとき，サービスを再起動しても再び攻撃されるとサービスが停止するため，機械学習だけではサービスの停止を防げない．さらに，これらの機械学習は多種多様な教師データを必要とするが，攻撃に関する教師データは十分に収集することが困難な場合がある．特にゼロデイ攻撃の教師データを収集することは困難である．一方，正常なデータに関する教師データも収集することが難しい場合がある．例えば，正常なデータを多数のユーザや端末から大量に収集した場合には，大量のデータがすべて正常であることを保証することが難しいと考えられる．

機械学習によって検出した攻撃を学習することによって新しい攻撃に適応する手法も提案されている[6]．しかし，上述の通り，機械学習には誤検出が発生するため，これらの手法は誤検出したデータを誤学習する可能性があり，誤学習はさらに誤検出を増やす要因になる．誤学習を防ぐため，検出した攻撃が間違いなく攻撃であることを確認できる仕組みが必要である．

ネットワークトラフィックを監視するのではなく，アプリケーションプロセスの振る舞い（システムコールシーケンス[7]や制御フローの完全性[8]など）を監視して，ヒューリスティックにゼロデイ攻撃を検出して防止する技術がある．しかし，この技術は攻撃を検出したとき，プロセスを強制終了させることによって攻撃を防止するため，サービスの停止が問題となる．

多様性に基づくフォールトトレランス設計によって，ゼロデイ攻撃の耐性を強化する侵入耐性システム（Intrusion Tolerant System）[9]がある．侵入耐性システムは，実装が異なる複数のサーバアプリケーションを並行稼働させる．実装上の脆弱性が原因で 1 つのサーバアプリケーションがサービス不能になったとしても，他のサーバアプリケーションがサービスを代行する．仕様上の脆弱性のように実装に共通する脆弱性でなければ，ゼロデイ攻撃によりサービスが停止することはないと考えられる[10]．しかし，侵入耐性システムは，実装が異なる多数のサーバアプリケーションを実行するために多くのリソースを必要とする上に，各サーバアプリケーションのセットアップやメンテナンスなどのコストが大きい．

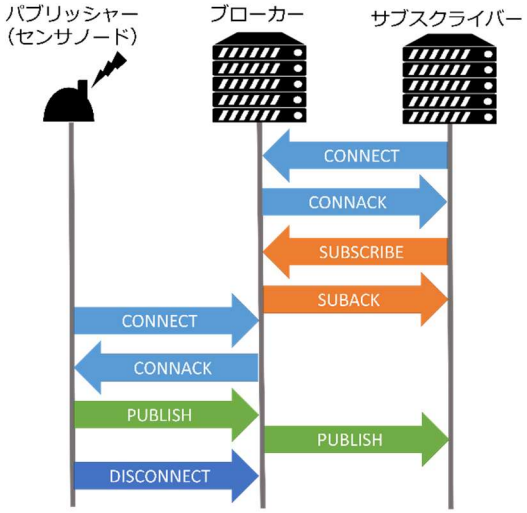

図 1　MQTT の通信フロー

3. MQTT と Mosquitto の脆弱性

MQTT は軽量な Publish/Subscribe 型データ配信プロトコルであり，IoT サービスで利用されるプロトコルの 1 つである．MQTT ブローカーの実装は本稿執筆時点で 17 の実装が存在する[11]．これらの実装の中から，著者らの研究成果を活用できる実装として，オープンソースの中から，C 言語で実装された Mosquitto を選んだ．また Mosquitto は CVE-2018-12543 や CVE-2019-11779 などサービスの停止を引き起こす脆弱性が発見されていることも理由の 1 つである．

本章は Mosquitto の動作を理解するために，MQTT の通信フローを述べ，過去に報告された Mosquitto の脆弱性と本稿で取り扱う脆弱性を示す．

3.1 MQTT の通信フロー

MQTT にはバージョン 3 と 2019 年に公開されたバージョン 5 があるが，本稿では広く普及しているバージョン 3 の MQTT を取り扱う．

MQTT の一般的な通信フローを図 1 に示す．MQTT のクライアントにはデータを送信するパブリッシャーとデータを受信するサブスクライバーがある．パブリッシャーはブローカーに CONNECT メッセージを送ってから PUBLISH メッセージで計測データなどを送信し，DISCONNECT メッセージでブローカーとの通信を終了する．サブスクライバーはブローカーに CONNECT メッセージを送ってから SUBSCRIBE メッセージを送って，指定したトピックのデータ受信を待ち受け続ける．

この他に Will メッセージの遺言機能，伝達保証や認証などの機能があるが，本稿ではこれらの機能を取り扱わない．

3.2 Mosquitto の脆弱性

NIST や Mosquitto のウェブサイト[12]によれば，セキュリティ侵害を引き起こす可能性がある脆弱性は本稿執筆時点で 14 個ある（クライアントの脆弱性は除く）．その中で，

Vol.2021-DPS-186 No.59
Vol.2021-CSEC-92 No.59
2021/3/16

リモートから攻撃可能であり，可用性が侵害される脆弱性を表 1 に示す．

CVE-2017-7651 と CVE-2017-7654 はメモリが枯渇してサービスが停止する攻撃であり，これらは OS やブローカーの設定により対策できるため，本研究の対象としない．さらに，CVE-2019-11778 と ISSUE 番号＃1244[13]はバージョン 5 の MQTT を対象としているため，これらも対象としない．ただし，Mosquitto は二つのバージョンを同時にサポートするため，ファイアウォールでバージョン 5 のメッセージを拒否することとする．以上から，本稿が取り扱う脆弱性は CVE-2018-12543 と CVE-2019-11779 である．

表 1　遠隔から可用性を侵害可能な Mosquitto の脆弱性

CVE or ISSUE	CVSSv2	機密性	完全性	可用性
CVE-2019-11779	4.0			✓
CVE-2019-11778	5.5		✓	✓
#1244	n/a			✓
CVE-2018-12543	5.0			✓
CVE-2017-7654	5.0			✓
CVE-2017-7651	3.5			✓

4.　Mosquitto ブローカーのための免疫的攻撃検知

免疫的攻撃検知は，自然免疫機能と獲得免疫機能を備える（図 2）．自然免疫機能は免疫系の自然免疫に対応する仕組みであり，未知の攻撃も検知できるが，サービスの停止を手がかりにして攻撃を検知するため，サービス停止そのものを防げない．一方，獲得免疫機能は免疫系の獲得免疫に対応する仕組みであり，最初の攻撃は検知できないが，自然免疫機能が検知した攻撃を学習して，類似の攻撃であれば 2 回目以降の攻撃を未然に検知することによってサービスの停止を防ぐことができる．これらの機能は Mosquitto のソースコードを改造することによって組み込まれる．Mosquitto のソースコードに関する記述は，CVE-2019-11779 が原因のサービス停止を再現するため，Mosquitto のバージョン 1.6.1 に基づく．OS は Linux とする．

4.1　自然免疫機能

自然免疫機能は，サイバー攻撃の検知に加えて，獲得免疫機能に入力する教師データ（受信した MQTT メッセージのデータ群）の保存とプロセスの再起動を行う．これらの仕組みや動作について述べる．

4.1.1 攻撃検知

自然免疫機能が対象とする攻撃はサービス妨害（DoS）やリモートコード実行（RCE）を引き起こす脆弱性に対する攻撃である．ネットワーク帯域を消費させる攻撃や OS 内部のリソースを消費させる攻撃は対象としない．

DoS の脆弱性にはメモリアクセス違反によるものとアサーションの失敗によるものがある．前者の脆弱性が攻撃されたら，プロセスはセグメンテーション違反のシグナル

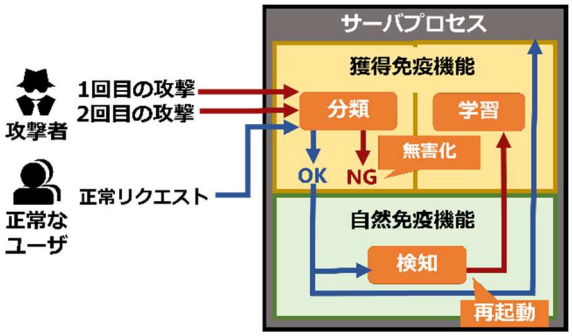

図 2　免疫的攻撃検知の構成と仕組み

（SIGSEGV）を受け取る．後者の脆弱性が攻撃されたら，プロセスは異常終了のシグナル（SIGABRT）を受け取る．つまり，これらのシグナルの受信が攻撃検知となる．

一方，RCE の脆弱性に対する攻撃を検知する機能は Mosquitto に備わっていないため，コンパイル時とリンク時に組み込める対策技術を使って，攻撃を難しくするか，攻撃を検知できるようにする．本研究では GNU C コンパイラとリンカを使うこととし，次の対策機能を Mosquitto に組み込む．

- スタックスマッシュ攻撃対策（-fstack-protector-strong）
- スタッククラッシュ攻撃対策（-fstack-clash-protection)
- 制御フローの保護（-fcf-protection=full）
- 符号付き整数のオーバーフロー検知（-ftrapv）
- 安全な関数への置換（-D_FORTIFY_SOURCE=2)
- 位置独立実行ファイルの生成（-fPIE, -pie）
- メモリセグメントの読み取り専用化（-z relro)
- コード実行可能なスタックの禁止（-z noexecstack）

これらの機能を有効にすると，プロセスは攻撃されたとき SIGABRT などのシグナルを受け取れるようになる．これらのシグナル受信によって多くの攻撃を検知できる．

Mosquitto にはこれらのシグナルを処理するシグナルハンドラーが存在しないため，攻撃を検知できたとしても，シグナル受信時にプロセスは終了する．そこで，自然免疫機能はプロセス起動時にこれらのシグナルハンドラーを作成して，シグナルハンドラーが攻撃検知時の処理を行えるようにする．

シグナルハンドラーは，シグナルの原因となった攻撃メッセージを特定する．シグナルハンドラーはシグナルが発生する直前に処理したメッセージを攻撃メッセージと判断する．このメッセージを攻撃と判断する根拠は，Mosquitto ブローカーがシングルスレッドであり，Mosquitto ブローカーにはメッセージを処理中に別のメッセージを処理する仕組みが存在しないためである．メッセージが正常なメッセージであると判断するタイミングは 4.2.2 項で述べる．

4.1.2 教師データの保存とプロセスの再起動

教師データはキューで管理される．正常メッセージは正常メッセージ用キュー（正常キュー）に，攻撃メッセージ

は攻撃メッセージ用キュー（攻撃キュー）に保存される．各キューが上限サイズに達したら，古いメッセージから順番に破棄される．キューのサイズ L は免疫的攻撃検知のパラメータであり，これは検出精度に関係するため，5.3 節で述べるシミュレーション評価により決定する．

最後に，シグナルハンドラーは 2 つのキューのメッセージをファイルに保存してプロセスを終了する．プロセス終了後に Mosquitto のプロセスが直ちに再起動されるように Systemd や System V の init システムに再起動の設定を加えておく．

4.2 獲得免疫機能

獲得免疫機能は，教師あり機械学習に加えて，機械学習による攻撃検知と攻撃の無害化を行う．本節では学習モデルに入力する特徴量と，獲得免疫機能の各動作について述べる．

4.2.1 学習モデルに入力する特徴量

学習モデルに入力する特徴量はクライアントから受信した MQTT メッセージの先頭から N バイト目までのバイト列を N 次元の特徴量にマッピングしたデータである．MQTT メッセージが N バイト未満なら，残りのデータには値 256 を割り当てる．

MQTT メッセージの最大サイズは 256MB であり，この最大サイズのメッセージを機械学習で処理するには N を最大サイズに設定する必要がある．機械学習の分類速度は次元数が大きくなるにつれて遅くなる．N を最大サイズに設定した場合，1 メッセージあたり 10 数秒も要する．

特徴量の次元 N は免疫的攻撃検知の重要なパラメータであり，検出精度と分類速度に影響を及ぼすため，5 章で述べる性能評価により決定する．攻撃データを N バイト目以降に仕込んだ攻撃については 6 章で考察する．

4.2.2 教師あり機械学習による学習モデルの生成

獲得免疫機能は Mosquitto ブローカーのプロセスが起動した直後に教師データで学習を行って学習モデルを生成する．教師データは自然免疫機能がプロセス終了間際に保存した正常キューと攻撃キューのメッセージである．学習モデルを生成したら，MQTT ブローカーのサービスが始まる．

機械学習アルゴリズムの種類は免疫的攻撃検知のパラメータであり，検出精度と分類速度に影響を及ぼすため，5 章で述べる性能評価により決定する．

4.2.3 機械学習による攻撃検知と攻撃の無害化

獲得免疫機能はクライアントからメッセージを受信するたびに学習モデル（以下，分類器）でメッセージを正常または攻撃に分類する．分類器による分類処理は Mosquitto のソースファイルの 1 つである read_handle.c で定義された関数 handle__packet に追加する．この関数はメッセージの制御タイプごとにメッセージ処理関数を呼び出して，各メッセージを処理する．この関数の 3 行目からメッセージの処理が始まるので，2 行目に分類処理を追加する．

分類器がメッセージを攻撃クラスに分類したとき，メッセージの先頭から 2 バイト目の「残りデータサイズ」の後に続くデータを無害化する．無害化ではデータをゼロで上書きする．無害化したら，関数 handle__packet の処理を続行する．

分類器がメッセージを正常クラスに分類したとき，関数 handle_packet の処理を続行する．メッセージを正常キューに追加するタイミングは，分類器がメッセージを正常クラスに分類したタイミングではない．分類器が誤って攻撃メッセージを正常クラスに分類する可能性があるからである．正常であると判断するタイミングは関数 handle__packet が処理を正常に完了するタイミングである．つまり，この関数が戻り値を呼び出し元に返すまでに SIGSEGV と SIGABRT のシグナルが発生しなかったとき，獲得免疫機能はこの関数が処理したメッセージを正常なメッセージであると判断して，正常キューにこのメッセージを追加する．

5. 性能評価

免疫的攻撃検知を組み込んだ Mosquitto ブローカーの性能を評価する．まず，シミュレーション評価により妥当な免疫的攻撃検知の各種パラメータを明らかにする．次に実在した 2 つの脆弱性に対して攻撃を行って，攻撃の検出精度を評価する．また，前章で述べた免疫的攻撃検知のプロトタイプを実装して，免疫的攻撃検知のオーバーヘッドを評価する．

5.1 性能評価のシナリオ

様々なパブリッシャーとサブスクライバーがメッセージを繰り返しブローカーに送ることを想定する．各クライアントは CONNECT→PUBLISH / SUBSCRIBE→DISCONNECT の順番にメッセージを送信する．正常なパブリッシャーとサブスクライバーが送受信するデータは以下の通りとする．

- クライアント ID
  10 文字から 100 文字のランダムな文字列とする．
- メッセージ ID
  0 から 65535 の疑似乱数とする．
- トピック
  トピックレベルは 1 から 4 の疑似乱数で決定し，各レベルのトピック名は 1 文字から 8 文字のランダムな文字列とする．最下層レベルのトピック名は温度，湿度，水位を想定して，それぞれ temperature, humidity, waterlevel とする．サブスクライバーの場合に限り，トピック名にワイルドカード（「＃」または「＋」）を 10%の確率で割り当てることとする．

Vol.2021-DPS-186 No.59
Vol.2021-CSEC-92 No.59
2021/3/16

- メッセージ
  各トピックのメッセージは温度，湿度，水位を想定するため，−100 から 100 の疑似乱数とする．

ブローカーには CVE-2018-12543 と CVE-2019-11779 の脆弱性が同時に存在すると仮定する．1 つのサーバアプリケーションに二つ以上の脆弱性が同時に存在して，同時期に両方が攻撃されるようなケースは少ないと考えられるが，防御側が不利な状況を想定するため，2 つの脆弱性が同時に存在して両方が攻撃されることとする．

CVE-2018-12543 は，トピックの先頭文字がドル記号のとき（$ SYS を除く），アサーションが失敗してサービスが停止する脆弱性である[14]．この脆弱性を攻撃するクライアントは上記のトピックの先頭文字を「$」に置換した PUBLISH メッセージまたは SUBSCRIBE メッセージを送信する．ただし，最初のトピックが $ SYS の場合はサービス停止を引き起こさないので，$ SYS で始まるトピックを除く．

CVE-2019-11779 は，およそ 65,400 個以上のトピックセパレータ（スラッシュ記号）がトピックに含まれるとき，スタックオーバーフローを起こしてサービスが停止する脆弱性である[15]．この脆弱性を攻撃するサブスクライバーは 65,400 個から 65,535 個のセパレータを持つランダムな文字列をトピックに指定した SUBSCIBE メッセージを送信する．

性能評価のシナリオは，次の通りである．

1) 初めてブローカーがサービスを開始して，パブリッシャーまたはサブスクライバーが合計 X 個の正常なメッセージをブローカーに送信してブローカーはこれらを正常に処理する．X 個のメッセージには CONNECT メッセージ，PUBLISH メッセージ，DISCONNECT メッセージ，SUBSCRIBE メッセージを含む．個数 X は 5.4 節で評価する事前学習データの個数を表す．
2) ブローカーは CVE-2018-12543 または CVE-2019-11779 の脆弱性に対するサイバー攻撃を受ける．このとき，ブローカーは再起動してから X 個の正常メッセージと 1 個の攻撃メッセージを学習してサービスを再開する．
3) ブローカーは 1,500 個の正常メッセージと 1,500 個の攻撃メッセージをランダムな順番で受信する．

性能評価は上述の第 3 ステップから行う．検出精度はランダムに生成されたメッセージの内容や送信順により変化すると考えられるため，それぞれの性能評価では 50 回の試行を行い，試行ごとにメッセージの内容と順番をランダムに変更する．検出精度などの性能指標は 50 回の試行の平均値とする．

5.2 分類器の比較評価

Python の scikit-learn モジュールを用いて，様々な分類器の検出精度と分類時間を比較評価する．検出精度は混同行列から求める．分類時間は 1 つのメッセージを分類するのに要した平均時間である．比較する分類器は scikit-learn に含まれるすべての分類器である．ただし，これまでの研究により，勾配ブースティングの実装は scikit-learn のものより LightGBM[16]の方が優れているため，LightGBM を使用する．分類器のパラメータはすべてデフォルト値を使用する．LightGBM のパラメータのみ，scikit-learn の勾配ブースティングのデフォルト値を使用する．免疫的攻撃検知のパラメータには以下の値を設定する．

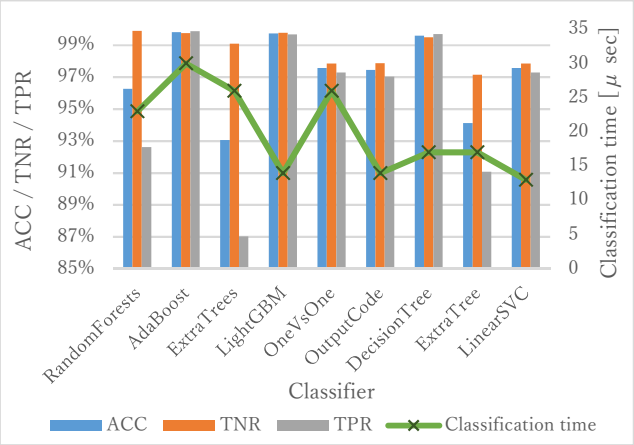

図 3　分類器の検出精度と分類速度

- 事前学習データ数 X：100
- キューのサイズ L：無制限
- 特徴量の次元数 N：300

図 3 に検出精度と分類時間の結果を示す．検出精度が 85%以下の分類器と分類時間が 35 マイクロ秒以上の分類器は除外した．左側の縦軸が検出精度（ACC），真陰性率（TNR），真陽性率（TPR）であり，右側の縦軸が分類速度である．分類速度は折れ線グラフで示している．

検出精度は AdaBoost が最上位であったが，AdaBoost の検出精度と同程度の LightGBM は AdaBoost と比べて 2.4 倍も速いことから，検出精度と分類速度のバランスを重視すれば LightGBM が適している．本節以降から分類器を LightGBM とする．

5.3 キューのサイズの評価

適切なキューのサイズを求めるために，各キューのサイズを 20 から 200 まで変更して各サイズの検出精度を調べた．免疫的攻撃検知のパラメータはキューのサイズを除いて 5.2 節のものと同じである．

図 4 にキューのサイズと検出精度の関係を示す．図 4 からキューのサイズが 70 を超えたとき，検出精度が 99.60%付近に収束した．この結果から検出精度を 99.60%以上にするには，キューのサイズを 70 以上に設定すればよい．

獲得免疫機能はブローカーの起動時に学習するため，学習が起動時間に大きな影響を与える．キューのサイズと学習時間を調べた結果，サイズが 200 以内なら学習時間は 1 秒以内であった．学習時間が 1 秒程度なら，ブローカーの

Vol.2021-DPS-186 No.59
Vol.2021-CSEC-92 No.59
2021/3/16

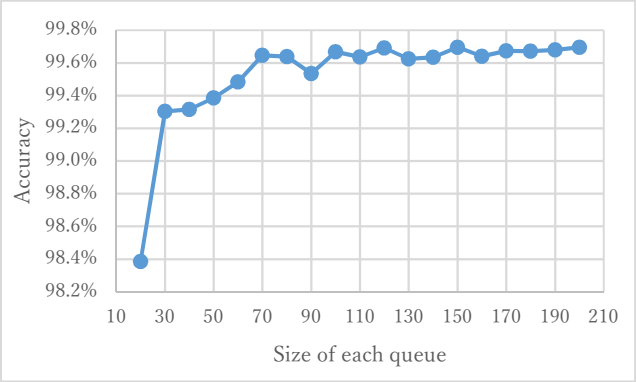

図 4　キューのサイズと検出精度

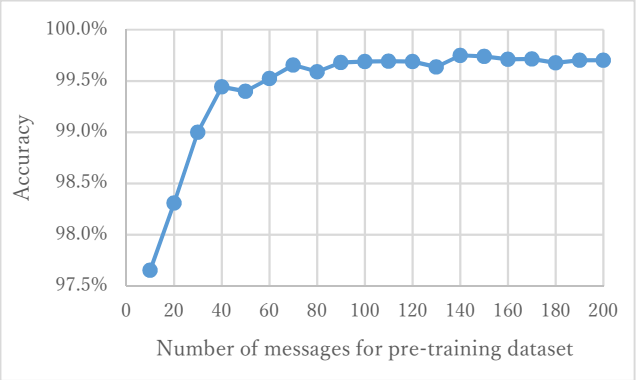

図 5　事前学習データ数と検出精度

起動にほとんど影響がないと考えられる．これらの結果から，キューのサイズ L を 100 に設定する．

5.4 事前学習データ数の評価

適切な事前学習データ数 X を求めるために，事前学習データ数を 10 から 200 まで変化させて各データ数の検出精度を調べた．免疫的攻撃検知のパラメータは事前学習データの個数を除いて 5.2 節のものと同じである．なお，事前学習データには正常なメッセージしか含まれない．

図 5 に事前学習データ数と検出精度の関係を示す．図 5 からデータ数が 70 を超えたとき，検出精度が 99.60%付近に収束した．この結果から事前学習データ数 X を 100 に設定する．

実際の IoT サービスでは，トピックはランダムな文字列ではなく，地名や建物名など場所を表す固有名詞が多いと考えられる．サービス開始前にすべてのトピックを把握できる場合，事前学習データ数を増やせば，誤検知（偽陽性）を減らせると考えられる．そこで，トピックは日本全国の市区名までの住所（1,893 件）[17]からランダムに選ばれると仮定して，事前学習データ数と検出精度の関係を調べた．その結果，予想通り，事前学習データ数を増やせば，TNR を改善できることがわかった（図 6）．

5.5 特徴量の次元数の評価

適切な特徴量の次元数 N を求めるために，次元数を 1,000 から 10,000 まで変化させて，各次元数における MQTT メ

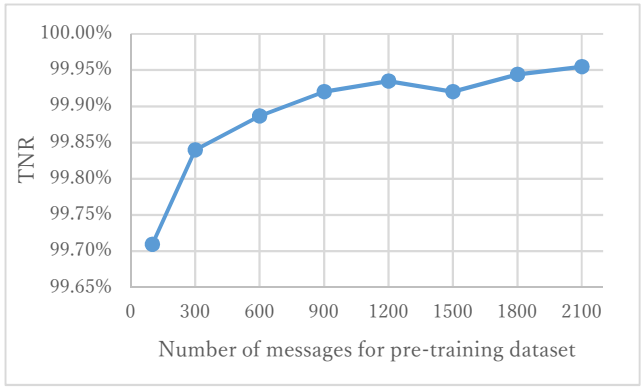

図 6　事前学習データ数と検出精度（住所トピック）

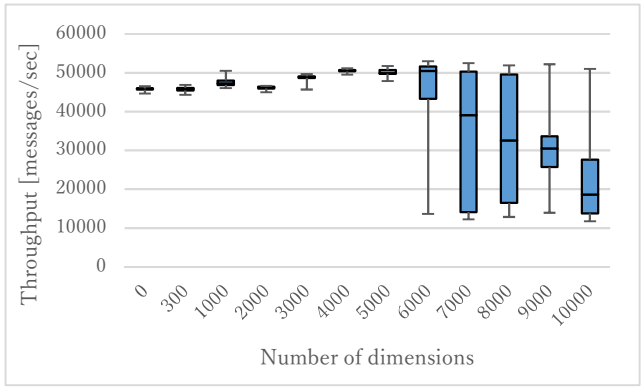

図 7　特徴量の次元とメッセージのスループット

ッセージのスループットを計測した．免疫的攻撃検知のパラメータは特徴量の次元数を除いて 5.2 節のものと同じである．スループットは，シミュレーションではなく，免疫的攻撃検知を組み込んだ Mosquitto の実機で計測した．計測はレイヤ 2 ギガビットイーサネットスイッチを介して 1 台のブローカーに接続された 1 台の クライアント上で行った．ブローカーとクライアントのマシンとソフトの仕様は次の通りである．

- Broker: Mosquitto 1.6.1
  - CPU: Ryzen Threadripper 2950X (30,510 score)
  - Memory: 128 GB
  - OS: Linux (Kernel 5.9.0)
- Client: MQTTLoader 0.7.3 [18]
  - Celeron G3900 @ 2.8GHz (2,220 score)
  - Memory: 8 GB
  - OS: Linux (Kernel 5.6.0)

脆弱性に対する攻撃を検証するため，Mosquitto のバージョンには CVE-2019-11779 の脆弱性を持つバージョン 1.6.1 を選んだ．ブローカーの CPU は Intel CET[1]に対応していないため，4.1.1 項で述べた制御フロー保護は機能しない．

MQTTLoader は MQTT のベンチマークソフトである．本研究の MQTTLoader は 1 つのパブリッシャーを起動して，250 万個のメッセージを発信することにした．計測結果の箱ひげ図を図 7 に示す．図 7 において 0 次元の結果は免疫的攻撃検知を組み込んでいない Mosquitto の結果である．

Vol.2021-DPS-186 No.59
Vol.2021-CSEC-92 No.59
2021/3/16

図 7 から次元数が 5,000 までなら，免疫的攻撃検知を組み込んだ Mosquitto と免疫的攻撃検知を組み込んでいない Mosquitto のスループットは同程度であることがわかった．この結果は MQTTLoader のパブリッシャーの個数を増やしてもほとんど変わらなかった．

本研究は温度や水位などの観測データの収集を想定しているので，メッセージサイズは最大でも数百バイトになると考えられる．また，本稿で取り扱った CVE-2018-12543 と CVE-2019-11779 の脆弱性は，その性質上，MQTT メッセージの先頭から 300 バイト以内に攻撃特有のデータが出現するため，特徴量の次元数を 300 バイトに設定すれば両方の攻撃を検出できる．次元数の増加はシミュレーション時間の増加につながるため，シミュレーション評価の都合により，次元数 N は 300 に設定する．ただし，MQTT メッセージの先頭から 301 バイト目以降に攻撃データを仕込める脆弱性が発見された場合は，免疫的攻撃検知は攻撃を検知できない．このような攻撃の対策は次章で考察する．

5.6 検出精度の変化の分析

1 回のシミュレーション評価における検出精度の変化を調べた．免疫的攻撃検知のパラメータは以下の通りである．

- 機械学習アルゴリズム：LightGBM
- 事前学習データ数 X：100
- キューのサイズ L：100
- 特徴量の次元数 N：300

図 8 に受信メッセージの 1 番目から 35 番目までの真陰性率（TNR），真陽性率（TPR），検出精度（ACC）の変化を示す．0%と 100%の水平線上のプロットはメッセージの正常または攻撃の種別を表している．正常なメッセージなら 0%の軸上に，攻撃であれば 100%の軸上にプロットしている．

図 8 から，最初の 3 つの攻撃メッセージを見逃しているが，4 番目の攻撃から検出できるようになり，それ以降，検知と見逃しを繰り返しながら，少しずつ TPR が改善していることがわかる．一方，正常なメッセージは一度も誤検知されることはなかった．最終的に 3,000 個のメッセージを受信したとき，検出精度は 99.87%，真陰性率は 100.0%，真陽性率は 99.73%となった．

表 2 に 50 回の試行の最終的な検出精度，真陰性率，真陽性率の統計を示す．偽陽性の回数（正常なメッセージを誤って攻撃として検知した回数）は平均で 3.38 回であった．つまり，3〜4 個の正常メッセージがブローカーで処理されないことを意味する．このような誤検知は，5.4 節で述べた通り，事前学習データ数を増やすことによって減らせると考えられる．偽陰性の回数（攻撃を検知できなかった回数）は平均で 5.02 回であった．つまり，5〜6 個の攻撃メッセージを学習すれば，獲得免疫機能が残りの類似の攻撃をすべて検知できたことを意味する．

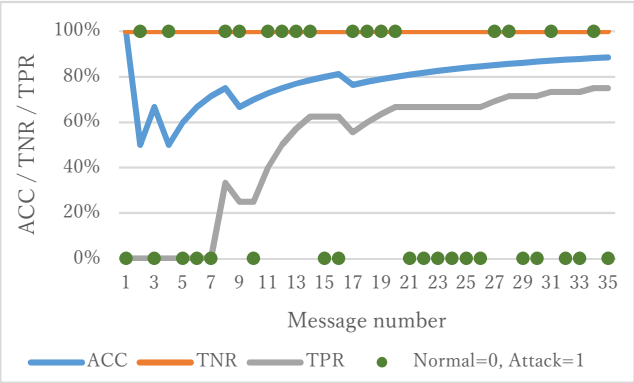

図 8　検出精度の変化

表 2　検出精度

Accuracy	99.7206±0.3216
TNR	99.7707±0.6147
TPR	99.6704±0.2626

5.7 脆弱性と検出精度の分析

各脆弱性の検出精度を調べるため，CVE-2018-12543 と CVE-2019-11779 に対する攻撃を混ぜないでそれぞれの攻撃について評価した．表 3 にその結果を示す．CVE-2019-11779 の結果は CVE-2018-12543 と比べて約 0.3 ポイント高かった．CVE-2019-11779 に対する攻撃には多数のスラッシュが含まれ，多数のスラッシュが LightGBM による分類を容易にしたと考えられる．

表 3　各脆弱性の検出精度

	CVE-2018-12543	CVE-2019-11779
Accuracy	99.6566±0.3205	99.9286±0.2334
TNR	99.6467±0.5887	99.9360±0.4480
TPR	99.6664±0.2532	99.9213±0.0393

6. 考察

6.1 特徴量の次元数を超えるメッセージによる攻撃

特徴量の次元数を N としたとき，獲得免疫機能は MQTT メッセージの N＋1 バイト目からのデータをチェックしない．そのため，N＋1 バイト目以降に脆弱性を悪用するデータが仕込まれたら攻撃を検知できない．

このような攻撃を防止するため，MQTT メッセージの最大サイズを制限する方法が考えられる．Mosquitto は message_size_limit オプションによりメッセージの最大サイズを指定できる．このサイズを N に設定すれば，N+1 バイト以上のメッセージは処理する前に破棄されるため，攻撃を防止できる．本研究が想定するような温度や水位などの小さなデータであれば，最大サイズの制限による副作用はないと考えられる．

最大サイズを制限できない場合には，マルウェア分類などで使用されるファジーハッシュの利用が考えられる．ファジーハッシュ値は 2 つのメッセージが似ているなら，ファジーハッシュ値も類似しているという性質を持つ．この

性質を利用すれば，攻撃メッセージに類似したメッセージを検出できると考えられる．ただし，ファジーハッシュはメッセージの要約の過程でメッセージの詳細部分の情報が失われるため，高い検出精度は期待できないと考えられる．そこで，メッセージの先頭 N バイトだけでも高い検出精度を確保するため，メッセージ全体のファジーハッシュ値に加えてメッセージの先頭Nバイトのデータを学習モデルに入力する方法が考えられる．例えば，SSDEEP のファジーハッシュ値は最大で 96 文字の文字列で表現されるので，分類器の入力データは N+96 次元のベクトルデータとなる．

6.2 正常キューのフラッシュ攻撃

攻撃者が攻撃の前に大量の正常メッセージを送ってから，攻撃メッセージを送ると，正常キューがフラッシュされ，正常キューは攻撃者が用意したメッセージで埋められる．このとき，獲得免疫機能は攻撃者が用意したメッセージを学習することになる．攻撃メッセージが正常なメッセージに類似していた場合，獲得免疫機能はこの攻撃メッセージと類似した正常メッセージを攻撃として誤検知するようになる．

このようなフラッシュ攻撃を緩和するために事前学習データとキューを独立して管理する方法が考えられる．事前学習データをキューに加えずに不変のデータとして保持すれば，事前学習データが攻撃によりフラッシュされることはない．5.4 節で述べたように事前に把握できるメッセージを事前学習データに加えておけば，TNR の向上にもつながる．

攻撃を緩和するのではなく，キューを使わないことによって攻撃そのものを無力化する方法も考えられる．ナイーブベイズなど一部の機械学習アルゴリズムは新しい学習データのみで学習モデルを更新できるものが存在する．このようなオンライン学習が可能な機械学習アルゴリズムを使えば，キューを利用しなくてよいため，フラッシュ攻撃を無力化できる．

7. おわりに

本稿では MQTT プロトコルのバージョン 3 においてサービス停止を引き起こす Mosquitto ブローカーの脆弱性を示し，Mosquitto ブローカーのための免疫的攻撃検知に関する仕組みとプロトタイプの実装を述べた．シミュレーション評価により妥当な免疫的攻撃検知のパラメータを決定して，2 つの脆弱性（CVE-2018-12543 と CVE-2019-11779）への攻撃に対して免疫的攻撃検知の性能を評価した．その結果，検出精度は，平均で 99.72%，真陰性率は 99.77%，真陽性率は 99.67%であり，極めて良好な結果が得られた．さらにプロトタイプを実装して，ブローカーにおけるメッセージのスループットを計測した．その結果，特徴量が 5,000 次元までなら，免疫的攻撃検知のオーバーヘッドはほとんどないことがわかった．最後に免疫的攻撃検知に対する攻撃と対処方法について考察した．

今後は CVE-2019-11778 など MQTT プロトコルのバージョン 5 に特有の脆弱性に対する攻撃をサポートする．

謝辞 本研究は JSPS 科研費 JP20K05012 の助成を受けたものである．

参考文献

[1] “Control-flow Enforcement Technology Specification”, Revision 3.0, Intel. 2019.
[2] A. L. Buczak and E. Guven A survey of data mining and machine learning methods for cyber security intrusion detection, IEEE Commun Surv Tutor. 2016, vol. 18, no. 2, p. 1153–1176.
[3] Okamoto, T. et al.. An Immunity-Enhancing Security Module for Cloud Servers. IJICIC. 2020, vol. 16, no. 1, p. 137-151.
[4] Ashfaq, R. A. et al.. Fuzziness Based Semi-Supervised Learning Approach for Intrusion Detection System. Information Sciences. 2017, vol. 378, p. 484-497.
[5] Lopez-Martin, M. et al.. Application of deep reinforcement learning to intrusion detection for supervised problems. Expert Systems with Applications. 2020, vol. 141, 112963.
[6] Danforth, M. WCIS: A Prototype for Detecting Zero-Day Attacks in Web Server Requests. In Proc. of USENIX LISA’11. 2011.
[7] Haider, W. et al.. Detecting Anomalous Behavior in Cloud Servers by Nested Arc Hidden SEMI-Markov Model with State Summarization. IEEE Transactions on Big Data. 2017, vol. 5, no. 3, p. 305-316.
[8] Rahman, M. et al.. Origin-Sensitive Control Flow Integrity. In Proc. of USENIX Security 19. 2019, p. 195-211.
[9] Zheng, J. et al.. Security Evaluation of a VM-Based Intrusion-Tolerant System with Pull-Type Patch Management. In Proc. of 2019 IEEE HASE. 2019, p. 156-163.
[10] Garcia, M. et al.. OS Diversity for Intrusion Tolerance: Myth or Reality?. In Proc. of DSN 2011. p. 383-394.
[11] “Comparison of MQTT implementations”. https://en.wikipedia.org/wiki/Comparison_of_MQTT_implementations, (参照 2021-01-24).
[12] “Reporting Security Vulnerabilities”. https://mosquitto.org/security/, (参照 2021-01-24).
[13] “mosquitto crash when using will with properties V5”. https://github.com/eclipse/mosquitto/issues/1244, (参照 2021-01-24).
[14] “CVE-2018-12543”. https://nvd.nist.gov/vuln/detail/CVE-2018-12543, (参照 2021-01-24).
[15] “CVE-2019-11779”. https://nvd.nist.gov/vuln/detail/CVE-2019-11779, (参照 2021-01-24).
[16] Ke, G. et al.. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Advances in Neural Information Processing Systems. 2017, p. 3146-3154.
[17] “Geolonia 住所データ”. https://github.com/geolonia/japanese-addresses, (参照 2020-08-26).
[18] Banno, R. et al.. Measuring Performance of MQTT v5.0 Brokers with MQTTLoader. In Proc. of 2021 IEEE CCNC. 2021, https://github.com/dist-sys/mqttloader, (参照 2021-01-24).