修士論文

MQTT over QUIC ブローカーへの Initial 攻撃の包括的な分析と防御手法の評価

人見 信
奈良先端科学技術大学院大学
先端科学技術研究科
情報理工学プログラム

主指導教員: 藤川 和利 教授
（情報科学領域）

令和 6 年 1 月 30 日提出

本論文は奈良先端科学技術大学院大学先端科学技術研究科に
修士 (工学) 授与の要件として提出した修士論文である。

人見 信

審査委員：

藤川 和利 教授	（主指導教員，情報科学領域）
笠原 正治 教授	（副指導教員，情報科学領域）
新井 イスマイル 准教授	（副指導教員，情報科学領域）

MQTT over QUIC ブローカーへの Initial 攻撃の包括的な分析と防御手法の評価*

人見 信

内容梗概

近年、IoT 分野において、QUIC プロトコルを利用する MQTT over QUIC プロトコルが注目を集めている。このプロトコルは、従来の MQTT プロトコルに比べ、より高速な通信を実現する。しかし、MQTT と QUIC の新しい組み合わせにより生じるセキュリティリスクは、多くが未検証のままである。特に、QUIC プロトコル固有の脆弱性を悪用する Initial 攻撃により、MQTT over QUIC ブローカーの可用性が損なわれる可能性がある。Initial 攻撃は、QUIC プロトコルのハンドシェイクプロセスを標的とし、サーバー側のリソースを不当に消費させるものである。本研究では、MQTT over QUIC ブローカーに対する Initial 攻撃の影響を検証し、その防御手法を提案する。検証では、Initial 攻撃により、MQTT over QUIC ブローカーの可用性が損なわれることを明らかにした。また、セッションの継続を行うことにより、Initial 攻撃によるブローカーの可用性の低下を防ぐことができることを確認した。

本研究の主要な目的は以下の二点である。

1. MQTT over QUIC ブローカーへの Initial 攻撃の影響を厳密に分析し、攻撃がブローカーの性能、セキュリティの脆弱性、そして Pub/Sub クライアントの可用性に与える損害を評価する。

2. Initial 攻撃に効果的に対抗する防御手法を開発し、リソース制限のある IoT 環境で MQTT over QUIC ブローカーの運用性とセキュリティを確保する。

*奈良先端科学技術大学院大学 先端科学技術研究科 修士論文, 令和 6 年 1 月 30 日.

キーワード

IoT, MQTT, QUIC, Initial attack, Denial of Service

Comprehensive analysis and evaluation of defense methods against Initial attacks on MQTT over QUIC brokers.*

Hitomi Shin

Abstract

In recent years, MQTT over QUIC, which uses the QUIC protocol, has attracted attention in the IoT field. This protocol realizes faster communication than the conventional MQTT protocol. However, the security risks arising from the combination of MQTT and QUIC have not been fully investigated. In particular, Initial attacks, which exploit the vulnerabilities of the QUIC protocol, may affect the availability of MQTT over QUIC brokers. Initial attacks target the handshake process of the QUIC protocol and consume server-side resources. This may degrade the performance of MQTT over QUIC brokers and compromise the stability of the system. In this study, we verify the impact of Initial attacks on MQTT over QUIC brokers and propose defense methods against them. We first analyze the impact of Initial attacks on MQTT over QUIC brokers and evaluate the damage they cause to broker performance, security vulnerabilities, and the availability of Pub/Sub clients. We then propose defense methods against Initial attacks, ensuring the operability and security of MQTT over QUIC brokers in resource-constrained IoT environments. The primary objectives of this study are as follows:

*Master's Thesis, Graduate School of Science and Technology, Nara Institute of Science and Technology, January 30, 2024.

1. To rigorously analyze the impact of Initial attacks on MQTT over QUIC brokers and assess the damage they cause to broker performance, security vulnerabilities, and the availability of Pub/Sub clients.
2. To develop effective defensive methods against Initial attacks, ensuring the operability and security of MQTT over QUIC brokers in resource-constrained IoT environments.

Keywords:

IoT, MQTT, QUIC, Initial attack, Denial of Service

目 次

図 目 次

表 目 次

1. はじめに

Internet of Things (IoT) の分野は、現代社会におけるデジタル技術の進化とともに、急速に重要性を増している。IoT デバイスの普及は、情報通信技術の発展とともに、加速され、これらのデバイスは日常生活から産業分野に至るまで幅広く組み込まれている。2024 年までには、約 400 億の IoT 機器の存在が予測されており、この数の増加は IoT の重要性を示している [1]。増加する IoT デバイスは、音や可視光、赤外線などの様々な物理世界のデータをデジタル世界に投影して、新たな価値の創造に貢献する。具体的には、センサーやアクチュエータを通じて収集されたリアルタイムデータは、日常生活の質の向上、産業プロセスの効率化、環境監視、健康管理など、さまざまな分野で今までになかった価値を提供する。例えば、スマートホーム技術はエネルギー消費の最適化を実現し、スマートファーミングは農業の持続可能性を高める。また、スマートシティの構築は都市のインフラ管理と市民の生活を革新する。これらは、IoT 技術の普及以前には、想像もできなかったことである。

IoT デバイスを活用するためには、デバイス間の通信が必要不可欠である。良い端末と良い通信プロトコルが揃って初めて、IoT デバイスはその潜在的な価値を発揮することができる。この文脈において、Message Queuing Telemetry Transport (MQTT) や Constrained Application Protocol (CoAP) などのプロトコルは、IoT コミュニケーションに広く採用されている。これらのプロトコルは、帯域幅が限られた環境でも効率的なデータ転送を可能にし、消費電力を抑制しつつ、安定した通信を提供することが特徴である [2]。また、IoT デバイス間の通信の最適化と信頼性は、IoT ネットワーク成功の鍵であり、全体的な IoT システムの性能と効率性に直接影響を与える。IoT 技術の普及は、私たちの生活を根本的に変え、より効率的で持続可能な未来を実現する可能性を秘めている。この文脈で、MQTT プロトコルなどの技術は、社会にとって、今後、さらに重要になると予測されている [2]。しかし、IoT の普及に伴い、IoT デバイス間の通信量が増加し、生活におけるセンシング規模が拡大していることから、IoT デバイスはサイバー攻撃により脆弱になる可能性が高まっている。IoT 端末の性能の限界により、攻撃の行いやすさと攻撃によるサービスへの影響が大きくなり、攻撃者にとって魅力的な

ターゲットになっている。MQTT や CoAP などのプロトコルは、IoT デバイス間の通信において重要な役割を果たしているが、これらのプロトコルは、より効率的かつ信頼性の高い通信を実現するために、さらなる改良が必要である。このような状況を考慮し、IoT デバイス間の通信の効率性と信頼性を維持するためには、新たな IoT 通信プロトコルの開発と採用が重要な選択肢である。

近年、前述の課題に対応するために、MQTT over QUIC プロトコルが提案されている [3]。MQTT over QUIC は、既存の MQTT プロトコルに Quick UDP Internet Connections (QUIC) を組み込んだプロトコルである。QUIC は、従来の Transmission Control Protocol (TCP) に代わる新しいプロトコルであり、UDP をベースにしている。その主要な特徴は、接続の確立時間の短縮、パケット損失時の回復の高速化、そして終端間暗号化の強制にある。QUIC はハンドシェイクのラウンド数を削減し、これにより接続の確立時間を大幅に短縮する。また、QUIC では、TCP のようにストリーム全体が一つのパケット損失によってブロックされないため、パケット損失時の回復が高速である。さらに、QUIC では、実装上、暗号化が強制されてるため、TCP を用いたシステムよりも安全が担保される。この暗号化は、TLS over TCP よりも電力の消費が少ない特徴も有する。これらの特徴が組み込まれているため、MQTT over QUIC は、IoT 環境に最適であると考えられる。MQTT は、軽量で帯域幅の制限された環境での使用を目的として設計されたプロトコルであり、これらの特徴をもつ QUIC と低消費電力の IoT デバイスという観点で相性がいい。

したがって、MQTT over QUIC の導入は、IoT 分野において通信プロトコルの性能向上に重要な役割を果たす [3]。このプロトコルによって、IoT デバイス間の通信は高速で信頼性が高く、セキュリティが強化され、不安定な通信環境下でも応答性が向上する。これらの新しい IoT 通信プロトコルの開発と採用により、IoT アプリケーションの利便性がさらに拡大すると考えられる。新しいプロトコルによって、デバイス間の通信がより効率的かつ安全に行われることで、IoT アプリケーションのパフォーマンスが向上し、さらに多様な用途への展開が可能になる。例えば、医療、産業オートメーションなど高度な安全性とリアルタイム性が必要な分野における IoT の応用が、より高度な機能とセキュリティを持って展

開されることが期待される。IoT の未来は、これらの革新によって、より安全で効率的なものへと進化していくと期待される。

従来の MQTT over TCP プロトコルに関しては、セキュリティ面での広範な研究が行われてきた [4]。これには、Denial of Service (DoS) 攻撃、中間者攻撃、総当たり攻撃といった様々なセキュリティ脅威への対策が含まれており、これらの攻撃に対する防御策が検証用ツールキットの開発を通じて詳細に検討と検証がなされてきた。これらの努力により、MQTT over TCP を運用する際のセキュリティ要件は明確化されている。

MQTT over QUIC は、その性能の面で MQTT over TCP よりも優れていると考えられている。QUIC プロトコルは、低遅延、高スループット、およびより効率的なネットワーク通信を実現するために設計されており、これにより IoT デバイス間のデータ転送が大幅に改善される可能性がある。一方で、MQTT over QUIC に関する知見はまだ不足しており、特に運用上の安全性に関しては十分な調査が行われていない。QUIC が新たに導入された MQTT over QUIC プロトコルは、MQTT over TCP では見られなかった新たなセキュリティリスクが生じている可能性があり、検証なしに実環境に投入した場合、攻撃を受けた際に実社会に重大な影響を与えてしまう可能性がある。したがって、MQTT over QUIC のセキュリティ面における課題を明確にし、運用に関するセキュリティ要件を明確にする必要になる。しかし、現時点では、MQTT over QUIC に固有の脅威は、多くの点が未検証である。MQTT over QUIC の普及が進むにつれ、このプロトコルに対する攻撃の試みが増加することが予想される。このため、MQTT over QUIC に特有の攻撃脅威を明らかにし、それに対する適切な対策を開発することは、このプロトコルのセキュリティを向上させる上で重要である。適切な対策を実現するためには、研究の焦点をどこに当てるかの仮説、専用のセキュリティツールキットの開発、そして詳細なセキュリティ評価の実施が求められる。先述したような個々の研究が積み重ねられることにより、攻撃の脅威を検証するための専用ツールやデータセットが充実し、MQTT over QUIC のセキュリティが向上するための多様な基盤が生まれる。しかし、現時点では、MQTT over QUIC に固有の脅威を研究するための基盤は整っていない。このため、MQTT over QUIC のセキュ

リティに関する研究は、IoT 分野における通信技術の進化に伴い生じる新たな課題に対処するために重要であると考えられる。本研究分野において新たな仮説と実験を積み上げることは、IoT 技術のさらなる発展において重要な役割を果たすと考えている。

MQTT over QUIC システムを構成する要素の中でも MQTT over QUIC ブローカーは、特に Initial 攻撃の対象になりやすいと考えられる。Initail 攻撃により、計算量やメモリなどのリソースを消費させることが MQTT over QUIC システムに対して怒ることが予想されるが、クライアント側は見えにくく攻撃対象になりにくい。その一方で、MQTT over QUIC ブローカーは、クライアントからの接続を受け付けるために常に待機しているため、攻撃対象になりやすいと考えられる。また、常にデーターセンター内に豊富なリソースが割り当てられるとは限らないため、リソースが限られた MQTT over QUIC ブローカーは、Initial 攻撃によりシステムの全体的な性能が低下する可能性がある。

本研究は、MQTT over QUIC プロトコルが直面する新たなセキュリティリスクとして、QUIC プロトコル固有の Initial 攻撃に焦点を当てる。この攻撃は、QUIC プロトコルの接続確立プロセスを悪用し、サーバーのリソースを不必要に消費させ、重大な脅威をもたらす。Initial パケットを受け取ったサーバーは、接続を確立するために Hnadshake パケットを返す。Handshake パケットは、共有鍵を生成するために大きなリソースを必要とする。大量の Initial パケットを送信することで、攻撃者はサーバーのリソースを消費させ、正常な通信プロセスを妨害する。ここで、Initial 攻撃は、攻撃対象のリソースを消費することで DoS 状態を作り出すことを目指すため、リソースが限られた IoT 環境では、この攻撃によるシステムの全体的な性能と Pub/Sub の可用性への影響が特に深刻であると考えられる。

本研究では、MQTT over QUIC のセキュリティリスクを明らかにし、それに対処するための新たな対策を開発することで、Initial 攻撃下での MQTT over QUIC システムの Pub/Sub の可用性を向上させることを目指す。具体的には、まず、MQTT over QUIC を使用した IoT 環境における Initial 攻撃の影響を詳細に分析する。そして、得られた結果を基に Pub/Sub システムのメッセージの配信とルーティングを担当する MQTT over QUIC ブローカーが Initial 攻撃の脅威に対抗す

るための防御手法を提案する。防御手法の有効性を検証し、MQTT over QUIC システムの Pub/Sub クライアントの可用性保護を図る。

本研究の構成を以下にまとめる。2 章では、研究対象とするシステムの概要について述べる。3 章では、関連研究と課題について述べる。4 章では、想定する攻撃手法と防御手法について述べる。5 章では、実験計画、実験結果について述べる。6 章では、実験結果の考察について述べる。7 章では、本研究についてまとめる。

本研究の貢献を以下にまとめる。

- Initial 攻撃が MQTT over QUIC ブローカーへ与える影響の分析を行った。具体的な実験を通じて、この攻撃が MQTT over QUIC ブローカーの性能に顕著な影響を与えることを明らかにした。

- 防御策に関する検証と分析を行った。レートリミットの設定、セッションの再利用、および Retry の閾値の調整の有効性について検証し、セッションの再利用が有効な防御手法であることを明らかにした。

2. MQTT プロトコルの概要と QUIC の重要な機能

本章では、MQTT over QUIC を含む MQTT の概要と、QUIC の重要な機能について説明する。

2.1 MQTT の概要

Message Queuing Telemetry Transport（MQTT）は、リソース制約のある環境での利用に適した、効率的かつ軽量なメッセージングプロトコルである。1999 年に IBM によって開発され、その後 OASIS によって標準化された。このプロトコルは、Publish/Subscribe モデルを採用し、クライアント間のメッセージ交換を最適化するように設計されている。

主要な特徴として、以下のようなものがある。

- 軽量性: TCP/IP プロトコルスタック上に構築され、オーバーヘッドが少なく、リソース制約のある環境での効率的な動作が可能である。
- 低帯域幅の要件: 低帯域幅環境での効果的な動作を可能とし、データ通信コストを最小限に抑える。
- 信頼性の高いメッセージ配信: Quality of Service（QoS）レベルの提供により、メッセージ配信の信頼性を保証する。
- データ保持: オフラインのクライアントのためにメッセージを保持し、再接続時に配信する機能を有する。
- セキュリティ: Transport Layer Security（TLS）/Secure Sockets Layer（SSL）を用いたデータ暗号化と認証をサポートする。

MQTT のアーキテクチャは、パブリッシャ、サブスクライバ、ブローカーという三つの主要コンポーネントに分かれる。パブリッシャはメッセージを生成し、特定のトピックに送信する。サブスクーはトピックを購読し、関連するメッセージを受信する。ブローカーはパブリッシャとサブスクライバ間のメッセージの仲

介を行い、適切なルーティングと配信を担う。このようなアーキテクチャにより、MQTT は、IoT デバイスの低帯域幅環境での効率的な動作を可能とし、データ通信コストを最小限に抑えることができる。

MQTT は、その軽量性と低帯域幅要件により、特に Internet of Things（IoT）分野で広く採用されている。これらの応用において、MQTT はデータのリアルタイム収集、効率的な処理、および信頼性の高い配信を提供し、現代のデジタル環境において重要な役割を果たしている。

実環境での応用について、説明する。IoT デバイスとして、スマートホーム、ウェアラブルデバイス、環境モニタリングシステムなどの IoT デバイスが MQTT を使用してデータを送受信する。遠隔監視を行うために工業プラントや農業施設でのセンサーデータの収集と監視に利用される。車両追跡システムにおいて、GPS デバイスからの位置情報をリアルタイムで送信するために使用される。スマートエネルギー分野において、電力消費を監視し、エネルギーの効率的な利用をサポートするスマートグリッド技術に利用される。ヘルスケア領域において、患者をモニタリングするシステムやウェアラブル医療機器での患者データの収集に使われる。その他の分野でも分散システム、モバイルアプリケーションなど、様々な分野で利用される。

2.1.1 Publish

MQTT プロトコルにおけるパブリッシュ機能は、メッセージを生成し、指定されたトピックに対してこれを送信するプロセスである。このプロセスは、情報の配信というメッセージングシステムの重要な役割を果たす。パブリッシュを行うクライアントをパブリッシャと呼ぶ。

パブリッシュを用いることによる主要な特徴、利点は以下の通りである。

- パブリッシャは、メッセージの受信者を気にせずにメッセージを送信できる。
- メッセージの送信者（パブリッシャ）と受信者（サブスクライバ）は、お互いに直接関連することなく独立して動作する。

- メッセージは、複数のサブスクライバに同時に配信することができ、大規模な配信システムに適している。

- パブリッシャは、異なるトピックやチャンネルにメッセージを送信でき、広範な応用が可能である。

2.1.2 Subscribe

Subscribe 機能は、メッセージングおよびパブリッシュ/サブスクライブモデルにおける重要なコンポーネントである。サブスクライブとは、クライアントが特定のトピックやチャンネルからメッセージを受信するためのリクエストを行うプロセスである。サブスクライブを行うクライアントをサブスクライバと呼ぶ。

サブスクライブを用いることによる主要な特徴、利点は以下の通りである。

- サブスクライバは、メッセージが利用可能になると自動的にそれを受信する。

- サブスクライバは、メッセージの発行元（パブリッシャ）から独立しており、直接の相互作用は必要ない。

- 多数のサブスクライバが、同じトピックに対してサブスクライブできる。

- サブスクライバは、必要に応じてトピックの購読を開始または終了することができる。

2.1.3 Broker

MQTT システムにおけるブローカーは、パブリッシャからのメッセージを受信し、これを適切なサブスクライバに配信する役割を担う。ブローカーはメッセージのルーティング、品質保証、セッション管理、セキュリティ管理など、複数の機能を提供し、効率的かつ信頼性の高いコミュニケーションの実現に不可欠である。

- メッセージのルーティング: Broker は、パブリッシャから受信したメッセージを適切なサブスクライバにルーティングする。

- 品質保証: 特定のメッセージングプロトコルで定義された品質保証（QoS）レベルに従ってメッセージを配信する。

- セッション管理: クライアントとのセッションを管理し、接続の維持、監視、および適宜切断を行う。

- セキュリティ管理: 認証、承認、およびメッセージの暗号化を通じて、通信のセキュリティを担保する。

ブローカーは、メッセージングシステムにおける重要なコンポーネントであり、メッセージのルーティング、品質保証、セッション管理、セキュリティ管理など、多岐にわたる機能を提供する。ブローカーは、現代のメッセージングアーキテクチャの基盤となっており、効率的で信頼性の高いコミュニケーションを実現するために不可欠である。

2.2 QUIC の機能

2.2.1 0-RTT, 1-RTT

1-Round-Trip Time (1-RTT), 0-Round-Trip Time (0-RTT) のそれぞれの機能の中身について説明する [5]。1-RTT と 0-RTT は、Transport Layer Security (TLS) 1.3 プロトコルにおける重要な概念であり、暗号化された通信の確立プロセスを特徴づける。これらの概念は、通信のセキュリティと効率性のバランスを最適化するために設計されている。この概念は、TLS1.3 の仕組みを利用してハンドシェイクを行う QUIC でも、同様の概念が採用されている。1-RTT は、クライアントとサーバー間の安全な接続確立に 1 回の往復時間を必要とする。このプロセスは、以下のステップを含む。

- クライアントハロー: クライアントはサーバーに対して接続要求を送信し、暗号化オプションを提供する。

- サーバーハロー: サーバーはクライアントの要求を受け入れ、選択された暗号化オプションとセッションキーを生成する。

- 暗号化された通信: クライアントとサーバーは共有されたキーを使用して、暗号化された通信チャネルを確立する。

- セッションチケット: サーバーは、クライアントのセッション情報を保存するために、セッションチケットを生成する。

1-RTT プロセスは、中間者攻撃に対して堅牢なセキュリティを提供するが、接続確立に必要な時間が長くなるという欠点がある。

一方、0-RTT は、TLS 1.3 で導入された新しい機能であり、クライアントが初回のハンドシェイクメッセージと共にデータを送信することを可能にする。このプロセスは以下のように進む。

- 初期データ送信: クライアントは、以前のセッションからの情報（セッションチケット）を利用して、サーバーに対して初期データと共に接続要求を送信する。

- レスポンス: サーバーは、クライアントからの初期データを受信し、即座に処理することができる。

0-RTT は、TLS1.3 プロトコルの進化した特徴の一つであり、クライアントが以前のセッションからの情報を利用してサーバーとの再接続を迅速化する機能である。この機能は、リアルタイム通信や頻繁な接続が必要なアプリケーションにとって、接続時間を大幅に短縮する重要な利点を提供する。

しかしながら、0-RTT の利用はセキュリティ上のリスクを伴う。特に、再生攻撃（リプレイ攻撃）に対する脆弱性が指摘されている。再生攻撃では、攻撃者はクライアントの通信を捉え、それを再送信することで、サーバーを騙して不正な操作を実行させる可能性がある。このリスクは、0-RTT を利用する際に特に慎重な検討を要するセキュリティと効率性の間のトレードオフを示している。

このため、0-RTT の利用は、セキュリティのリスクと利便性の利点を慎重に比較し、適用する環境や要件に基づいて決定されるべきである。セキュリティが最優先される環境では、0-RTT の利用を避け、通常の 1-RTT ベースの TLS ハンド

シェイクを選択することが推奨される。また、0-RTT を利用する場合は、再生攻撃に対する追加の防御機構を設けることが重要である。

以上をまとめると、1-RTT と 0-RTT は、セキュリティと接続時間の最適化を目的としており、その使用は異なるシナリオで異なるメリットとデメリットを持つ。1-RTT は、セキュリティが最も重要な要素である場合に適しており、0-RTT は接続速度を優先するシナリオに適している。セキュリティと効率性のバランスは、通信プロトコルの設計と実装における重要な課題であり、このトレードオフを適切に管理することが、安全かつ効率的なインターネット通信の未来において重要である。

2.2.2 Retry

QUIC は、Google によって開発された通信プロトコルであり、RFC 9000 シリーズによって標準化されている。このプロトコルは、特に Web 通信のパフォーマンスとセキュリティを向上させることを目的としており、TCP と TLS/SSL の一部の機能を置き換えることを目的としている。QUIC の主要機能について説明する。

- 接続の高速化: QUIC は、接続の確立に必要なラウンドトリップタイム（RTT）を減らすことで、特に TLS を使用する HTTPS 通信の高速化を図る。
- 暗号化: QUIC は TLS 1.3 をベースにした暗号化を組み込んでおり、通信のセキュリティを強化している。
- マルチプレックス化: 単一の接続上で複数のリクエスト/レスポンスストリームを同時に処理できるため、HTTP2 のマルチプレックス機能をより効率的に利用できる。
- ヘッドオブラインブロッキングの削減: QUIC はパケットの損失が他のパケットの配信を遅延させないため、TCP におけるヘッドオブラインブロッキング問題を軽減する。
- 接続の移行: IP アドレスが変更されても接続を維持できるため、モバイルデバイスのネットワーク変更がスムーズに行われる。

QUIC において、Retry 機能は特に重要な役割を果たす。この機能は、安全な接続確立を保証し、不正なトラフィックの拒否や Denial of Service 攻撃（DoS）への耐性を高めるために導入された。Retry 機能は、クライアントとサーバ間の初期のハンドシェイクプロセスにおいて、安全性を確保するための追加的な確認手段として機能する。Retry 機能は、以下のようなプロセスで動作する。

- **初期ハンドシェイク**
  - クライアントがサーバに接続要求を送信する。サーバは、この要求を受け入れる前に、クライアントに対して Retry 要求を送信ができる。このステップは、接続試行の正当性を確認し、セキュリティを高めるために重要である。
- **Retry 要求**
  - サーバがクライアントに対して Retry 要求を送信する。Retry 要求には、一時的なトークンが含まれる。このトークンは初期のクライアント検証の手段として機能し、通信プロセスの初期段階でのセキュリティリスクを軽減する。
- **トークンの生成と検証**
  - Retry 要求に含まれる一時的なトークンが、接続確立の鍵となる。クライアントは、このトークンを次の接続要求に含める必要がある。サーバはこのトークンを検証し、その結果に基づいてクライアントの要求を承認する。トークンの検証は、以前の Retry 要求と一致することを確認することで、トークンが有効であり、クライアントの要求が正当であることを確認することである。このトークンによる検証プロセスは、安全で認証された接続を保証し、不正アクセスや悪意のある活動のリスクを最小限に抑える。

Retry 機能の利点について説明する。Retry 機能により、QUIC は次のような利点を提供する。

- DoS 攻撃の緩和: 未確認の接続要求に対してリソースを割り当てることなく、サーバは正当なトラフィックのみを処理する。

- 正当な接続の確認: サーバはクライアントの接続要求が正当であることを迅速に確認できる。

- 効率的なネットワーク使用: 余分なハンドシェイク手順を排除することで、ネットワークのオーバーヘッドを減少させる。

QUIC プロトコルは、Retry 機能を通じて、DoS 攻撃の緩和、正当な接続の確認、効率的なネットワーク使用という利点を提供する。

3. 関連研究と課題

本章では、まず MQTT プロトコルへの様々な攻撃についての広範なサーベイについて述べる。次に、MQTT の通信効率とセキュリティを向上させるための、新しいプロトコル MQTT over QUIC の提案について述べる。ここで実環境で観測された Initial 攻撃の事例とその影響について述べる。続いて、MQTT に対する既知の攻撃に対する防御戦略とその有効性について述べる。最後に、関連研究の課題について述べる。

3.1 MQTT プロトコルへの様々な攻撃についてのサーベイ

MQTT プロトコルは、その軽量性と高いスループットの特性により、IoT デバイス間のメッセージングにおいて広く使用されている。特に、ネットワーク帯域幅が限られた環境や、処理能力が低いデバイスにおいて、広く使用されている [4]。しかし、このような特性は、セキュリティ面での脆弱性をもたらす可能性があり、MQTT プロトコルを利用したシステムは多様な攻撃手法の対象となっている。MQTT 運用環境に対する主要な攻撃タイプは、Denial of Service (DoS) 攻撃、中間者攻撃、総当たり攻撃の三つに分類される。DoS 攻撃は、サービスの正常な提供を妨害することを目的とした攻撃の総称である。この攻撃方法には、Flooding 攻撃、クライアントのなりすまし、MQTT Publish Flood、SlowITe 攻撃、有害なデータの挿入などが含まれる。

中間者攻撃は、通信の傍受や改ざんを行うことにより、MQTT 通信の機密性、完全性、可用性を侵害する [6]。この種の攻撃には、ARP ポイズニングや DNS ポイズニングなどの技術が用いられることが多い。

総当たり攻撃は、大量のユーザー名とパスワードの組み合わせを試し、適切な認証情報を見つけ出すことを目的とする [7]。MQTT ブローカーがユーザー名とパスワードによる認証機能を提供している場合、この攻撃によりセキュリティが脅かされる可能性がある。これらの攻撃手法について、次に具体的なメカニズムとそれがシステムに与える影響について詳しく解説する。

Flooding 攻撃は、意図的に大量の無駄なデータを MQTT ブローカーに送信す

る攻撃である。MQTT ブローカーを過負荷状態にし、正常なサービスを妨害するのが主な目的である。大量のメッセージや接続リクエストを MQTT ブローカーに送信し、その処理能力を超えさせることによってサービスの妨害を達成する。結果として、正常な通信が遅延し、最悪の場合はブローカーがクラッシュしてサービスが停止する。MQTT プロトコルは軽量であり、多くの場合、限られたリソースを持つデバイスで使用されることが多い。これにより、過剰なトラフィックに対して脆弱である可能性が高い。また、MQTT はブローカー中心のアーキテクチャを採用しており、この単一点が攻撃の標的になりやすい。

クライアントのなりすましは、パブリッシャーまたはサブスクライバに偽装し、大量のトピックを作成・削除し、購読・解除する攻撃である。MQTT システムでは、パブリッシャー（発行者）はブローカーにトピックを作成し、このトピックにメッセージを送信して配布する。その後、ブローカーはトピックの購読者に新しいメッセージについて通知する。そのため、ブローカーはトピックのパブリッシャーと購読者を追跡する必要が生じる。攻撃者はこの点を悪用し、パブリッシャーとしてトピックを作成・削除、購読者としてさまざまなトピックを購読・解除することでブローカーにリソースを消費させる。

MQTT Publish Flood は、MQTT ネットワーク上の悪意のあるデバイスから、利用可能なブローカーのリソース、つまり接続スロットや帯域幅をすべて消費するために、膨大な量のコンテンツをパブリッシュ（発行）する攻撃である [8]。Flooding 攻撃と同様の理由で、MQTT に対して、MQTT Publish Flood は攻撃としての効果が大きい。

SlowITe 攻撃は、ブローカーの接続スロットを占有し、新しい正常な接続を妨害する攻撃である [9]。長時間続くリクエストを同時に多数実行し、ブローカーの接続スロットを占める。結果、正規のデバイスやクライアントがブローカーに接続できなくなる。通常、ブローカーは接続を一定時間保持する。SlowITe 攻撃はこの特性を利用し、接続を意図的に長引かせてリソースを占有する。正常なリクエストと見分けが付かず、通常のセキュリティ対策では検出が困難な攻撃である。MQTT ブローカーは同時に処理できる接続数に制限がある。この攻撃は、これらのスロットを全て占めることで新しい接続の確立を妨害する。

ファジング (有害なデータの挿入) は、MQTT ブローカーに例外やエラーを引き起こし、サービスの停止または障害を引き起こす攻撃である [8]。プロトコルの標準に従わない、または不完全な形式の MQTT パケットをブローカーに送信する。プロトコルの実装ミスにより、特定の入力タイプやサービスコールの場合にサービス障害が発生する可能性がある。攻撃者は、特定の MQTT 実装に対して例外を発生させる目的で、形式が不正な MQTT パケットを MQTT ブローカーに送信することがある。

中間者攻撃は、通信データの傍受や改ざんを行い、MQTT 通信の機密性、完全性、可用性を侵害する攻撃である。Address Resolution Protocol (ARP) ポイズニングや Domain Name System (DNS) ポイズニングといった技術を使用して、通信経路に割り込む。通信を不正に傍受し、偽装したブローカーメッセージを使用して、クライアントを騙す。結果として、データが傍受され、機密性が侵害される、改ざんされたデータが送信され、通信の完全性が損なわれる、通信の乱れや遅延が発生し、可用性に影響を与える等々の影響が発生する。

総当たり攻撃は、大量のユーザー名とパスワードの組み合わせを試して、正しい認証情報を見つけ出す攻撃である。MQTT ブローカーは、デフォルトではないものの、ユーザー名とパスワードを使ってクライアント（パブリッシャーやサブスクライバー）を認証する機能を持っている。ユーザー名とパスワードを使ってクライアントを認証する機能を利用していた場合、攻撃者がこの方法でクライアントに成りすまし、機密情報にアクセスし、ブローカーの認証サービスを圧倒して正規ユーザーの利用を妨げる可能性がある [4]。次に、MQTT over QUIC の提案されたプロトコルの詳細と、それがもたらす利点について詳述する。

3.2 MQTT over QUIC プロトコル

QUIC は、通信の遅延を低減し、セキュリティ基準を向上させることを主な設計原則とするトランスポートプロトコルである。インターネット通信のセキュリティと効率性を最大限に高めるため、多くの機能が追加されている。

- **TLS1.3 とハンドシェイクの共用**:

  - 暗号化された接続を迅速に確立することが可能である。
  - TLS over TCP と比較して接続開始時間が短縮される。

- **QUIC 独自の組み込みの暗号化機能**:
  - データのプライバシーとセキュリティが強化される。

- **複数の独立したストリームのサポート**:
  - リソースをより効率的に活用できる。
  - パケット損失が発生しても独立したストリームは影響されず続行可能である。

- **高度な輻輳制御機構の使用**:
  - ネットワークが混雑している際でもパフォーマンスを維持する。

- **ヘッドオブラインブロッキングの削減**:
  - 一つのストリームの遅延が他のストリームに影響を与えず、データ伝送が効率的である。

- **IP アドレスの変更に対する強さ**:
  - モバイルデバイスがネットワークを切り替える際に同じ QUIC 接続を維持可能である。

- **柔軟な設計**:
  - 開発者が独自の輻輳制御アルゴリズムやその他の機能を実装しやすい。

- **0-RTT 再接続機能**:
  - 再接続時の遅延を大幅に削減する。
  - リプレイ攻撃の対象になる可能性があるため、採用は実装および運用に依存する。

特に、MQTT プロトコルは、QUIC のこれらの特性を活用することで通信効率とセキュリティを向上させる可能性がある。MQTT は TCP をトランスポートプロトコルとして使用しており、信頼性のあるエンドツーエンドの通信サービスを提供するが、TCP には多くのデメリットがある。これには、損失のあるネットワークへの対応の難しさ、非常に低い通信遅延の要件を適切に保証できない点などが含まれる。QUIC は、MQTT のこれらの問題を解決することが期待されている。

Kumar と Dezfouli は、IoT シナリオにおける Google QUIC の性能を調査した [3]。彼らは、Raspberri Pi 3B デバイスで構築されたさまざまなテストベッドで、MQTT の性能を QUIC と TCP で比較した。接続確立時のパケットオーバーヘッド、ランダムパケットの消失が存在する場合の遅延、エンドポイントの 1 つが接続を切断した場合のプロセッサとメモリ使用量、接続の移行中のスループットの低下などの観点で性能が評価された。これらの実験の結果、MQTT over QUIC はそれぞれの実験で MQTT over TCP を上回る性能を示した。しかし、パブリッシャの CPU 使用率は MQTT over QUIC の方が高い条件も存在した。

Fernández らは、遅延性能を分析した [10]。彼らは、さまざまな IoT 運用環境を考慮し、MQTT over QUIC と MQTT over TCP を比較した。1 つ目の実験では、パブリッシャからブローカーに 1000 の MQTT パケットを送信し、その後ブローカーが対応するメッセージをサブスクライバに転送する。各パケットは前のパケットからの応答を受信して確認した後に送信された。この際、確立した接続は実験が完了するまで維持された。そして、すべてのパケットが送信されるまでにかかる時間を測定し、それぞれ TTCP および TQUIC として計算された。この実験は、異なるパケット損失率を持つ 3 つのネットワーク構成によって行われた。各構成について、20 回の実験が実施された。結果、QUIC はすべてのシナリオで TCP が示した性能を常に上回り、低い RTT（Wi-Fi など）を持つ環境で、特に MQTT over TCP と比べて高い性能を示した。RTT が高く、遅延が支配的な要因となると、QUIC による改善の効果が目立たなくなり、特に衛星接続の場合には、改善の効果が目立たなかった。ただし、実験が行われたどのシナリオでも、QUIC が性能面で TCP を下回らなかった。

QUIC の接続確立と、TCP/TLS の接続確立の違いについて分析した研究がある [3]。接続が初期に確立されてからパブリッシュメッセージが送信されるまでの時間を測定し、2 回接続を確立することが 20 回行われた。QUIC の性能は TCP プロトコルに比べてわずかに優れていることがわかった。これは、QUIC が TCP よりも早く接続確立できることを示している。

以上のように、様々なシナリオで MQTT over QUIC は、既存の MQTT プロトコルよりも優れた結果が報告されている。しかし、MQTT over QUIC の導入に伴い、新たなセキュリティ上の課題が生じる。次に、MQTT over QUIC の導入に伴い生じる攻撃と既存の防御戦略について考察する。

3.3 実環境での Initial 攻撃の観測

MQTT の下部プロトコルが QUIC に変更されることにより、新たなセキュリティ上の検討が必要となる。近年、MQTT が IoT 環境で広く使用される中で、セキュリティ上の脅威が増加している [4]。特に、MQTT の下部プロトコルとして QUIC が採用されつつある現状では、この組み合わせ固有のセキュリティリスクが未解明である。この組み合わせにおいて独特のセキュリティリスクが存在する可能性があり、それがまだ明らかにされていないため、検討が必要である。ここで、MQTT over QUIC 環境に対する Initial 攻撃の影響を検討する。Initial 攻撃は、QUIC の初期接続フェーズでの暗号化処理を利用し、サーバーのリソースを消費する。IoT デバイスはリソースが限られているため、Initial 攻撃に対して特に脆弱であると予想される。よって、MQTT over QUIC 環境において、Initial 攻撃が試みられる可能性が高いと予想し、Initial 攻撃に焦点を当てる。

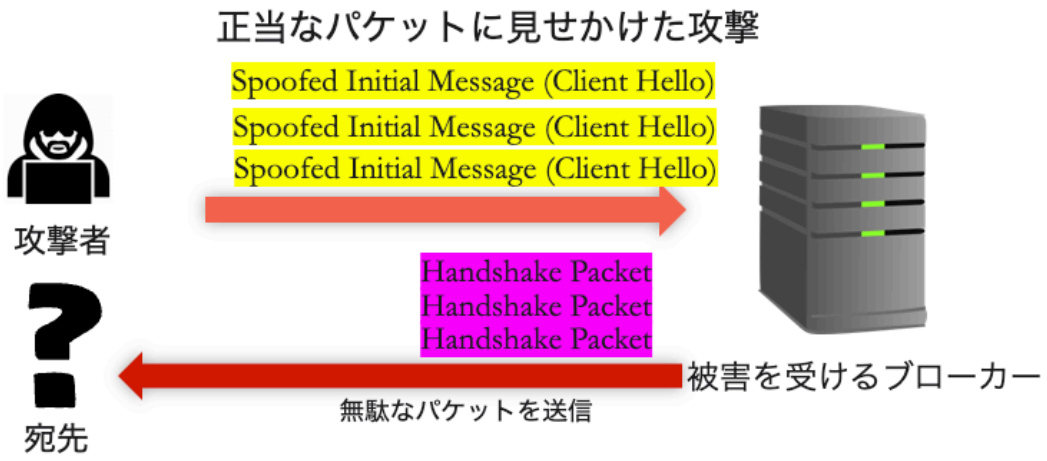

図 1: Initial 攻撃の概要

MQTT over QUIC の普及は未だ限られており、そのため攻撃に関するデータを収集することは困難である。この問題に対処するため、QUIC をトランスポートプロトコルとして使用する HTTP3 に焦点を当てることとした。実環境で、QUIC トラフィックを観測された研究が、Surname らによって行われていた [11]。

この研究によって、HTTP3 サーバを対象にしたような Initial 攻撃が facebook や Google に対して行われていることが確認された。さらに、Surname らは、Inital 攻撃が NGINX に対して有効な攻撃かどうかを確かめシステムの可用性を損なう攻撃であることを確認した。この事実は、Initial 攻撃が NGINX をはじめとした QUIC を利用するシステムの脅威となることを示している。Retry 機能を実装することで、Initial 攻撃を防御することができることは、実験で示唆されている。しかし、facebook や Google が運用する HTTP3 サーバでは、Retry 機能は攻撃が受けたサーバーでは実装されていなかった。これは、意図的に実装が省略されていると考えられる [11]。

3.4 Initial 攻撃に対する防御

Inital 攻撃の 1 つ 1 つのパケットは、通常のリクエストと区別することが困難である。なぜなら、QUIC のハンドシェイクは、あらゆる計算機が通信を開始し

ようと試みることを想定しており、最初のパケットの悪意を想定していない。攻撃者は、正規のデータパターンやプロトコルを模倣することで、セキュリティシステムを欺き、容易にネットワークに潜入する可能性がある。

現在の Initial 攻撃に対する防御策は、2 つある。1 つ目は、Retry 機能の実装である。QUIC プロトコルの Retry 機能は、サーバが初回のクライアント接続要求を受けた際に、安全性を確認するために一時的なトークンを含む Retry 要求をクライアントに送信することにより動作する。クライアントはこのトークンを含む新たな接続要求をサーバに送り返すことで、セキュリティを確保しながらハンドシェイクプロセスを完了する。しかし、Retry 機能は、実際にサーバーのリソースを消費する攻撃を受けたサーバーでは実装されていない [3]。

2 つ目は、暗号化処理を Network Acceleration Complex (NAC) に代替させることで、処理を高速化することである [12]。しかし、NAC を用意することは、コストがかかり、導入には手間がかかる。NAC の市場状況から、NAC の導入についての示唆が得られる [13]。NAC 市場全体は成長している。しかし、特に中小企業（SME）が NAC ソリューションを導入する際には、初期導入コストの高さや内部での技術対応能力の欠如が大きな障壁となっていることが示されている。これらの課題は、すべてのサーバーに NAC を導入するというシナリオがすべての組織にとって現実的でないことを示唆している。よって、すべてのサーバーに NAC が備えられるシナリオは非現実的と考えられる。MQTT over QUIC を利用するような生活や産業に関わるシナリオでは、ハードウェアに依存しない Initial 攻撃の影響を軽減するための運用上の工夫が求められる。

3.5 既存の研究の課題

MQTT over QUIC ブローカーへの Initial 攻撃の影響は未知である。そのため, 本節では、本研究で取り組む 2 つの課題について言及する.

1 つ目は、MQTT over QUIC ブローカーに対する Initial 攻撃の影響を検証することである。QUIC を利用するシステムに対しては、Initial 攻撃の脅威が予想され、IoT での運用に特化している MQTT over QUIC ブローカーは、特に脆弱

であると考えられる。しかし、その脅威が MQTT over QUIC ブローカーに与える影響については検証されていない。

2 つ目は、MQTT over QUIC ブローカーに対する Initial 攻撃を防御もしくは影響を回避する手法を提案することである。Initial 攻撃は、通常のリクエストと区別することが困難であり、暗号化処理を別に用意したハードウェアに代替させるなどの解決策しか存在していない。前述の対策は、IoT 環境用途では、システム運用に必要十分な構成を使用することが多く、Initail 攻撃を防御する時に特別なハードウェアを用意することは現実的ではない。MQTT over QUIC ブローカーの運用を工夫することで、MQTT over QUIC ブローカーに対する Initial 攻撃を防御することが必要である。

4. 本研究における攻撃手法と防御手法

本章では、MQTT over QUIC ブローカーに対する Initial 攻撃手法とその防御手法について説明する。最初に想定する攻撃手法について言及し、その後、検討する防御手法についても、利点と欠点を含めて言及する。この章の内容を仮説として、次章以降の実験において検証する。

4.1 想定する攻撃手法

4.1.1 MQTT over QUIC ブローカーに対する Initial 攻撃の概要

本節では、MQTT over QUIC ブローカーに対する Initial 攻撃の特性と影響について検討する。ブローカーに対する Initial 攻撃は、QUIC プロトコルの初期ハンドシェイクプロセスを対象とする攻撃手法であり、大量の Initial パケットを生成し、UDP ペイロードを通じてブローカーに送信することによって DoS 状態を作ることを狙った攻撃である。攻撃者はブローカーのリソースを消費させ、正常な通信プロセスを妨害する。

図 2 に Initial パケットのペイロードの暗号化について示す。Initial パケットのペイロードは、QUIC のハンドシェイクにおける暗号化処理によって暗号化される。図 3 に Initial パケットのヘッダの難読化について示す。QUIC パケットは、図 2 の過程にによって、ヘッダが難読化され送信される。

攻撃ツールを作成し、攻撃を行う際には、攻撃者は MQTT over QUIC ブローカーに対して、大量の Initial パケットを送信する。先述したような処理をリアルタイムに行った場合、攻撃のレートが不足するが、データセットを繰り返し再生することでより効果的な攻撃が可能となる。以後の実験では、この攻撃ツールを用いて攻撃を行った。本研究では、攻撃の主要な焦点をブローカー側に置いている。ブローカーの通信に使用されるポート番号は通常固定されるため、攻撃者にとって特定しやすい対象となる。これに対して、通常の MQTT Publish クライアントはメッセージ送信ごとに新しいポートを使用するため、攻撃対象になりにくい。Initial 攻撃が成功すると、CPU、メモリ、接続スロットなどのブローカーのリソースが不正に消費され、正常な通信プロセスが妨害されることが予想され

る。この結果、ブローカーの性能が低下し、正規のクライアントからのリクエスト処理が失敗、遅延する可能性がある。特にリソースが限られた環境では、この攻撃がシステム全体の性能と安定性に深刻な影響を及ぼす恐れがある。

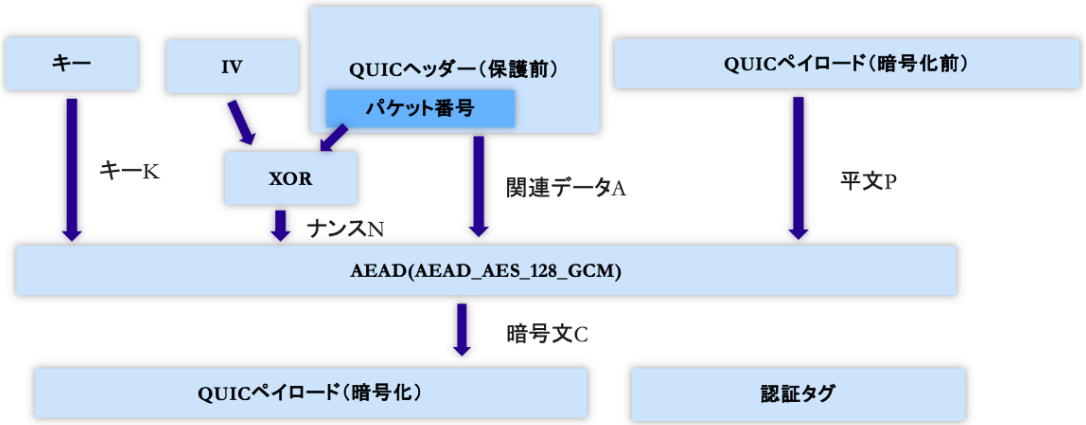

図 2: Initial パケットのペイロードの暗号化

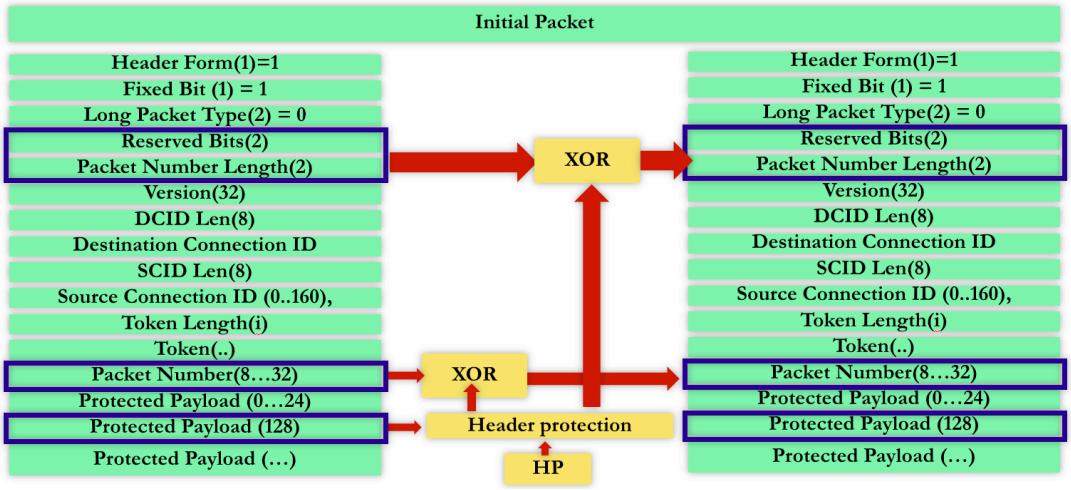

図 3: Initial パケットのヘッダとマスク

4.2 想定する防御手法

4.2.1 レートリミット (帯域制限) の設定

レートリミット (帯域制限) は、サーバーへのデータ転送速度を特定の帯域幅で制限することにより、サービスの品質を保ち、リソースの過剰な使用を防ぐ効果的な手段である。これは、トラフィックの管理、セキュリティの強化、リソースの効率的な利用に寄与する。具体的には、以下のように利点を示す。

- レートリミット (帯域制限) の利点
  - トラフィック管理とサービスの安定化: レートリミットは、予期しないトラフィックの急増によるサーバーの過負荷を防ぐものである。これにより、サービスが安定し、全てのユーザーに均等にサービスを提供することが可能である。
  - セキュリティ強化: 自動化された攻撃、例えば DoS 攻撃やブルートフォース攻撃を防ぐのに役立つ。レートリミットにより、攻撃者は短時間で多数のリクエストを送信することが困難になる。
  - リソースの効率的な使用: サーバーのリソースを効率的に使用することができ、過負荷によるコストの増大を防ぐ。
  - 公平性の確保: レートリミットは、一部のユーザーがサービスを過度に利用し、他のユーザーに不利益を与えることを防ぐ。

しかし、以下の欠点を伴うため、十分に運用環境の要件を検討する必要がある。

- レートリミット (帯域制限) の欠点
  - システム全体のパフォーマンスに影響を及ぼす可能性は高いこと: これは、特に高負荷時において、処理能力に制限を課すことが原因である。結果として、レスポンスタイムの低下が生じる恐れがある。
  - システム管理の複雑化: 調整やトラブルシューティングのプロセスを困難にする可能性がある。

– リソースの過剰な割り当て: リソースがレートリミットに過剰に割り当てられた結果、他のプロセスやアプリケーションのパフォーマンスに悪影響を及ぼすことがある。

以上のように、レートリミット (帯域制限) は、運用において重要な役割を果たす。これにより、サービスの安定性、セキュリティ、リソースの効率的な使用が促進され、全てのユーザーに公平なアクセスが提供される。しかし、利点と欠点の両方を考慮し、運用する必要がある。よって、本研究で 4 つのカーネルレベルでの帯域制限条件を設定し実験を行った。Initail パケットのサイズは、約 1220bytes であるため Initial パケットの 800 パケット分のカーネルレベルの帯域制限条件を設定し徐々に帯域制限を緩やかにしていく形で 4 つの条件を設定した。

- 0.976 Mbytes/秒
- 9.76 Mbytes/秒
- 48.8 Mbytes/秒
- 73.2 Mbytes/秒

4.2.2 Retry の閾値を調整

Retry の閾値を調整する手法で Initail 攻撃に対する防御を行うことを試みる。この閾値は、メモリ使用量に基づく Retry 機能の発動を制御するための閾値である。通常、Initail パケットに対して Handshake パケットが返されるが、Retry パケットが返される場合もある。Retry 機能は、任意の実装であるため、有効になっている場合と有効になっていない場合がある。Retry の閾値とは、メモリ使用量に基づく Retry 機能の発動を制御するための閾値である。MQTT over QUIC ブローカーは、クライアントからの接続要求を受け取ると、その要求を処理するためにメモリを割り当てる。本実験に用いたブローカーはメモリに余裕がない場合、ブローカーは Retry 機能を発動し、クライアントに対して接続要求を再送信するように要求する。ブローカーに割り当てられたメモリ利用量の一定割合を超えた

とき、メモリに余裕がないと判定される。単に機能の有効無効を切り替える場合と比べて、メモリ使用量に基づき Retry 機能の発動を制御することで、システムの性能と防御性を両立することが期待される。

閾値の設定にあたって、メモリ使用量に基づく Retry 発動の閾値は、システムの特性と要求に応じて適切に設定する必要がある。閾値が高すぎるとシステムの安定性が損なわれ、低すぎると再試行の機会が不足する可能性がある。適切な閾値により、システムのメッセージの配信性能を最適に保つことができる。本研究では、複数のパターンの Retry 閾値を設定し、攻撃に対する防御効果を試みた。実験では、デフォルトの 1/10 倍と 100 倍の閾値を用いて、これらがシステムの挙動に与える影響を観察した。閾値の大小がシステムの挙動にどのような影響を与えるかを観察する。

4.2.3 セッションを張り続ける防御手法

セッション継続により、MQTT クライアントとブローカー間の通信接続を持続させることが可能となり、これは接続確立に伴う遅延の削減とリソースの効率的利用を促進する重要な手段である。このアプローチは、接続確立の処理というオーバーヘッドを減少させ、結果としてシステム全体のパフォーマンスを向上させる。しかしながら、セッション継続の実装には、セキュリティ対策、セッション管理、そしてスケーラビリティの維持という課題が存在する。

セッション継続の主な利点は以下の通りである。

- 接続確立に伴う遅延の削減: 新しいセッションを頻繁に開始する必要がなくなるため、接続確立にかかる時間が削減される。
- 効率的なリソース利用: セッション再利用により、通信開始に伴うパケットのやり取りが減ることでサーバーとネットワークの負荷が軽減される。
- セキュリティの強化: アクセスパターンのモニタリングが容易になりセッションハイジャックなどのセキュリティリスクに対する防御策が強化される。

セッションの継続を効率的に利用できれば、システム全体に良い影響を与えるように思える。しかし、セッション継続を実装するには、以下のような実装上の考慮事項がある。

- セッションハイジャック: セッションが長期間にわたって継続されると、攻撃者によるセッションハイジャックのリスクが増加する。攻撃者が認証情報やセッション ID を入手すると、そのセッションを悪用することが可能になる。

- リプレイ攻撃: 長期にわたるセッションは、過去のメッセージを傍受し、後で不正に再送するリプレイ攻撃に対してより脆弱になる傾向がある。

- セッション管理: セッションの有効期限、更新、無効化などの管理が必要である。

- サーバーのリソース消費: 長期間継続されるセッションは、サーバーのメモリ、処理能力などを消費する。多数のクライアントが接続される環境では、よりリソース消費が大きくなる。

- ストレージの要求: セッション継続には、クライアントのセッション状態を記録し続ける必要があり、これには追加のストレージ容量が必要である。

まとめると、セッションを継続的に維持する手法は、レイテンシーの削減、リソースの効率的利用、ユーザーエクスペリエンスの向上、そしてセキュリティの強化に貢献する可能性を秘めている。しかし、これを実現するためには、セキュリティ、セッション管理、スケーラビリティといった要素への十分な配慮が不可欠である。この研究では、セッションを持続させる条件下での実験を行い、その効果を検証した。

5. 攻撃を想定した実験と防御実験

本章では、MQTT over QUIC 環境に対する Initial 攻撃の影響と防御策の有効性を検証するために行った実験について説明する。MQTT over QUIC ブローカーに対する Initial 攻撃の実際の影響を分析し、検証を行った。攻撃実験では、Initial 攻撃が MQTT over QUIC ブローカーに影響を与え、Pub/Sub 通信の成功率が低下することが期待される。具体的には、以下の事項を明らかにするために実験をおこなった。

1. 攻撃によるブローカーの性能への影響
2. セキュリティの脆弱性の特定
3. 攻撃が Pub/Sub クライアントへ与える影響の評価

そして、Initial 攻撃に対抗する効果的な防御手法を開発する為の実験を行った。防御手法の実験では、防御手法を適用することで、Pub/Sub 通信の成功率が向上することが期待される。

5.1 実験環境

本研究では、MQTT over QUIC ブローカーに対する Initial 攻撃の影響と防御策の有効性を検証するために、実験を行った。本節では、実験環境について説明する。

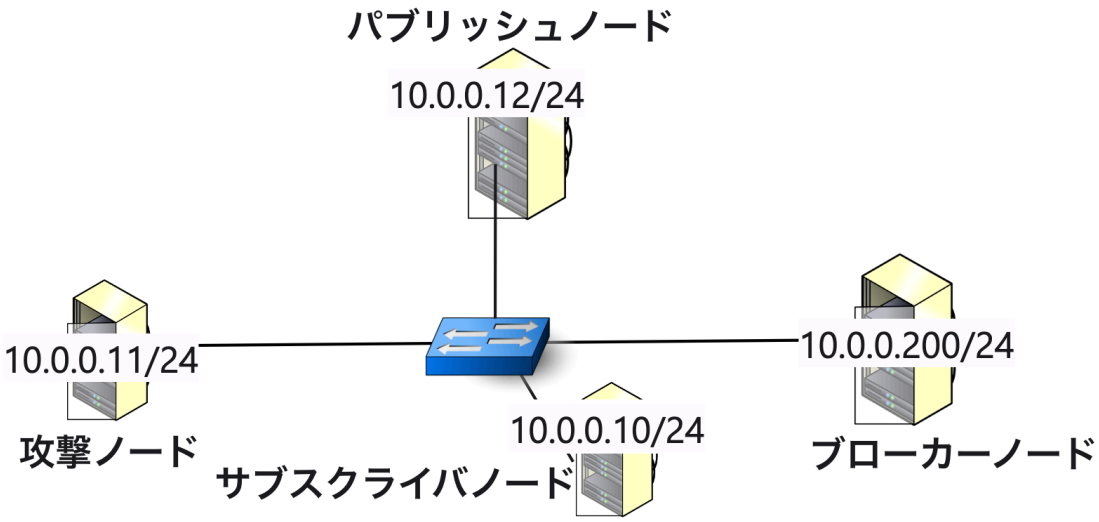

図 4: 実験環境

表 1: ハードウェア環境

OS	Ubuntu 20.04
CPU	Intel Xeon L5520 (2.27GHz, 8 コア)
RAM	10GB
NIC	1000BASE-T (1 Gbps)

表 2: ソフトウェア環境

ブローカー	emqx 5.1.3
クライアント	Erlang MQTT 5.0 Client, pynng-mqtt-v0.1 (実験 2-2 で使用した)

5.2 攻撃下でのクライアントの挙動についての実験

本節では、MQTT over QUIC ブローカーに対する Initial 攻撃下でのクライアントの挙動についての実験結果を報告する。実験は、攻撃状況下での MQTT over QUIC プロトコルのパフォーマンスと安定性を理解することを目的として行った。

5.2.1 実験 1-1:Initial 攻撃中に、Publish を行う

実験の目的は、Initial 攻撃が存在する状況下で Publish 処理を行った場合に、MQTT over QUIC ブローカーの可用性に与える影響の検証である。攻撃のレートとシステムの応答性を定量化し、攻撃下での性能低下の度合いを評価する。
以下の手順に従って実験を行った。実験前には時刻同期を行った。

1. ブローカーで、top、vmstat、netstat、tcpdump の記録を開始する (10 分間)。
2. ブローカーの記録開始後、10 秒後に Initial 攻撃を開始する Initial 攻撃は、6 分間継続される。
3. ブローカーの記録開始から、30 秒後、Publish を開始する (Publish を 0.1 秒に 1 回、合計 1000 回行う)。

5.2.2 実験 1-2:Publish 中に Initial 攻撃を行う

実験 1-2 では、Publish 処理中に Initial 攻撃を行うことで、ブローカーが攻撃にどの程度耐えうるかを検証する。メッセージの配信成功率を分析し、攻撃に対するブローカーの脆弱性を評価する。
以下の手順に従って実験を行った。実験前には時刻同期を行った。

1. ブローカーで、top、vmstat、netstat、tcpdump の記録を開始する (10 分間)。
2. Publish の試行を開始する (5 分間、0.1 秒に 1 回、実行する)。
3. ブローカーの記録開始後、1 分 40 秒経過後、Initial 攻撃を開始する。
4. Initial 攻撃は、1 分 40 秒継続される。

5.3 攻撃実験の結果

5.3.1 実験 1-1 の結果

実験 1-1 および 1-2 の結果は、Publish と Subscribe の成功率で評価した。成功率は、Subscribe の成功数を Publish の試行回数で割ったもので算出される。以降

の実験も同様の定義に基づいて結果が分析される。実験における攻撃レートは、図 5 のようになった。縦軸は、攻撃者から送信された 1 秒あたりの Initial パケット数を示している。横軸は、実験開始からの経過時間を示している。

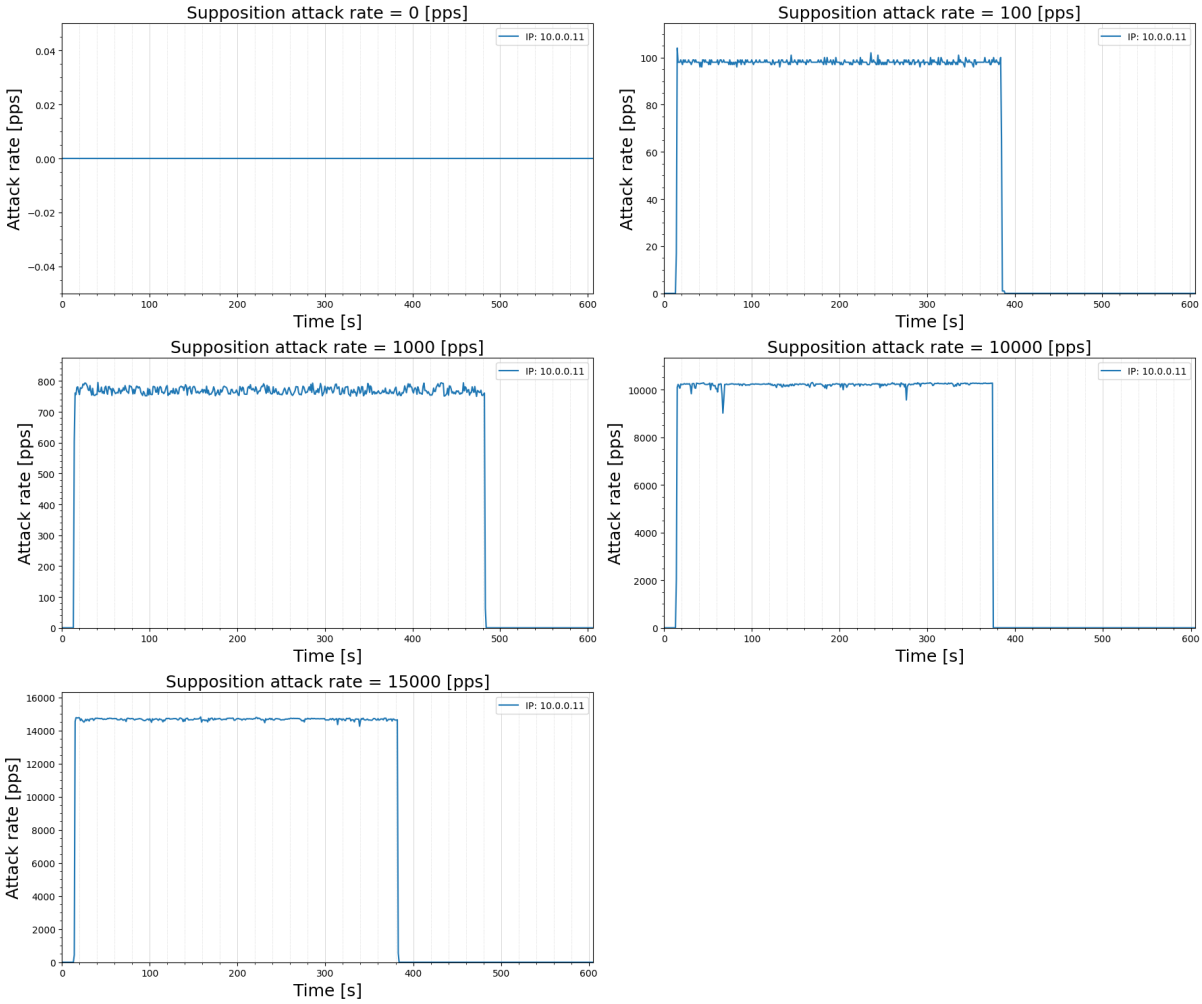

図 5: 実験 1-1: MQTT over QUIC ブローカーへの Initial 攻撃のレート

表 3: 実験 1-1 での Pub/Sub の成功率

送信レート [pps]	**成功率** [%]
0	100
100	99.9
1000	99.9
10000	44.4
15000	30.6

ブローカーが返しているトラフィック量、ブローカーの返すパケット数と CPU 使用率の結果、メモリの使用量の変化、攻撃元ポートの分散値について分析する。ブローカーの CPU 利用率の遷移を図 6 に示す。縦軸は、ブローカーの CPU 使用率を示している。横軸は、実験開始からの経過時間を示している。攻撃を受けている時は、CPU 利用率が高くなっていることがわかる。攻撃レートが高くなるほど、CPU 利用率が高くなっていることがわかる。

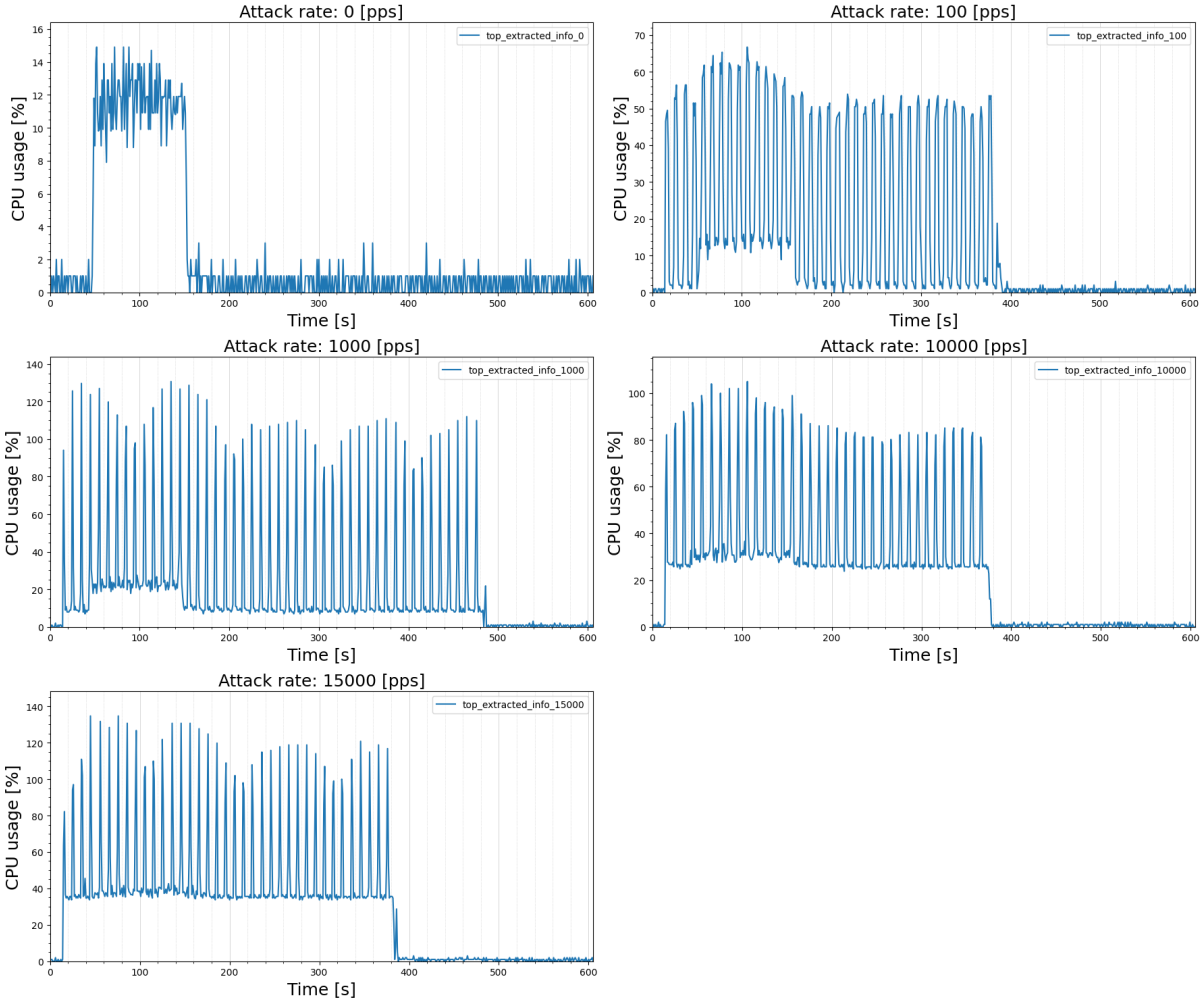

図 6: 実験 1-1 における CPU 利用率の遷移

実験中にブローカーが返しているトラフィック量について、図 7 に示す。縦軸は、実験中にブローカーが返しているトラフィック量を示している。横軸は、実験開始からの経過時間を示している。攻撃を受けている時は、ブローカーが返しているトラフィック量が高くなっていることがわかる。攻撃レートが高くなるほど、ブローカーが返しているトラフィック量が高くなっていることがわかる。

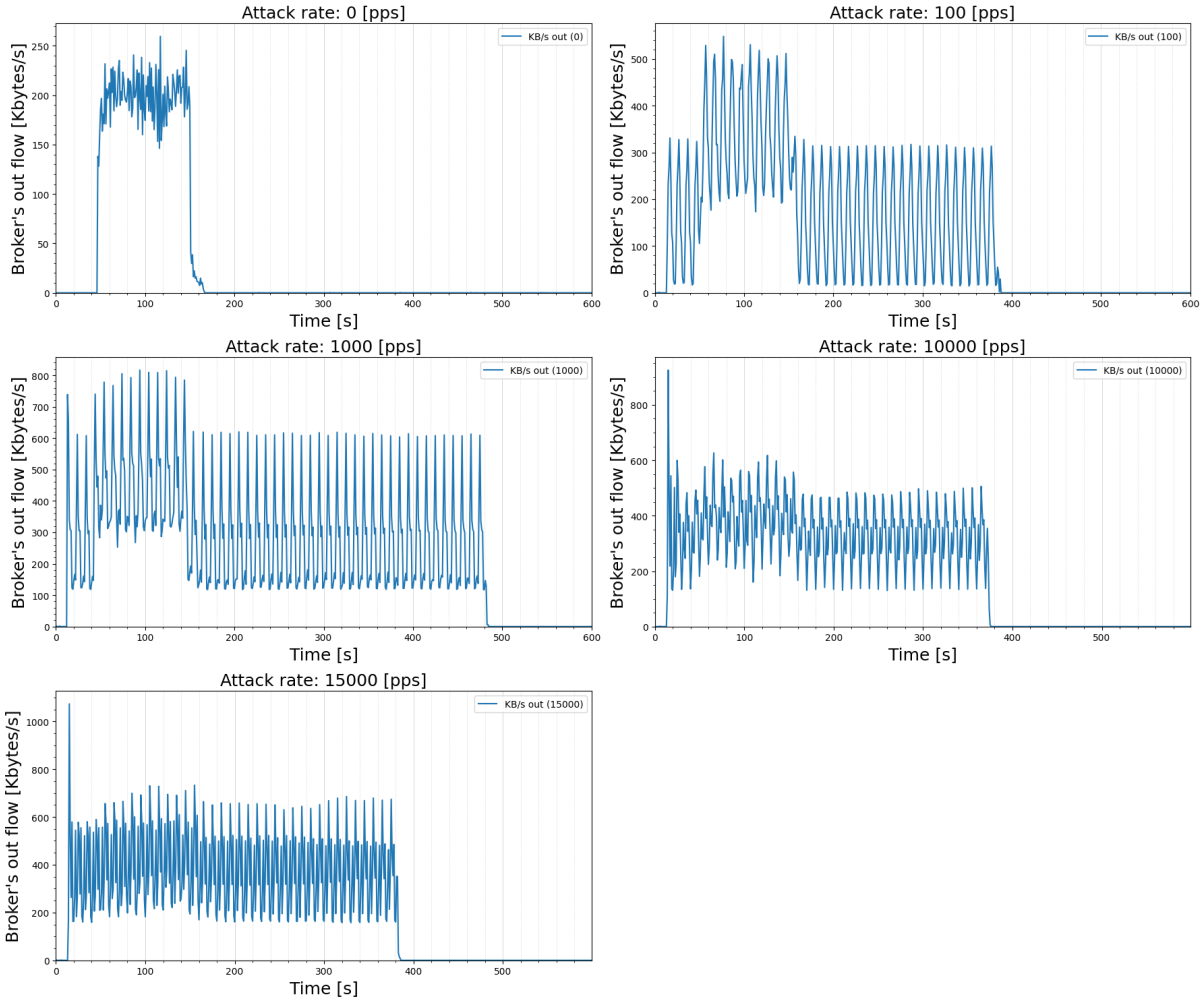

図 7: 実験 1-1 においてブローカーが返しているトラフィック量の遷移

攻撃者と正当なクライアントの送信元ポートの傾向を見るため、送信元ポートの分散をヒストグラムで示す。0 packets/秒の時の送信元ポートの分散を図 8 に示す。縦軸は、特定のポート範囲におけるパケットの数を示している。横軸は、送信元ポート番号を示している。IP アドレス、10.0.0.12 は正当なクライアントの IP アドレスである。IP アドレス、10.0.0.200 はブローカーの IP アドレスである。

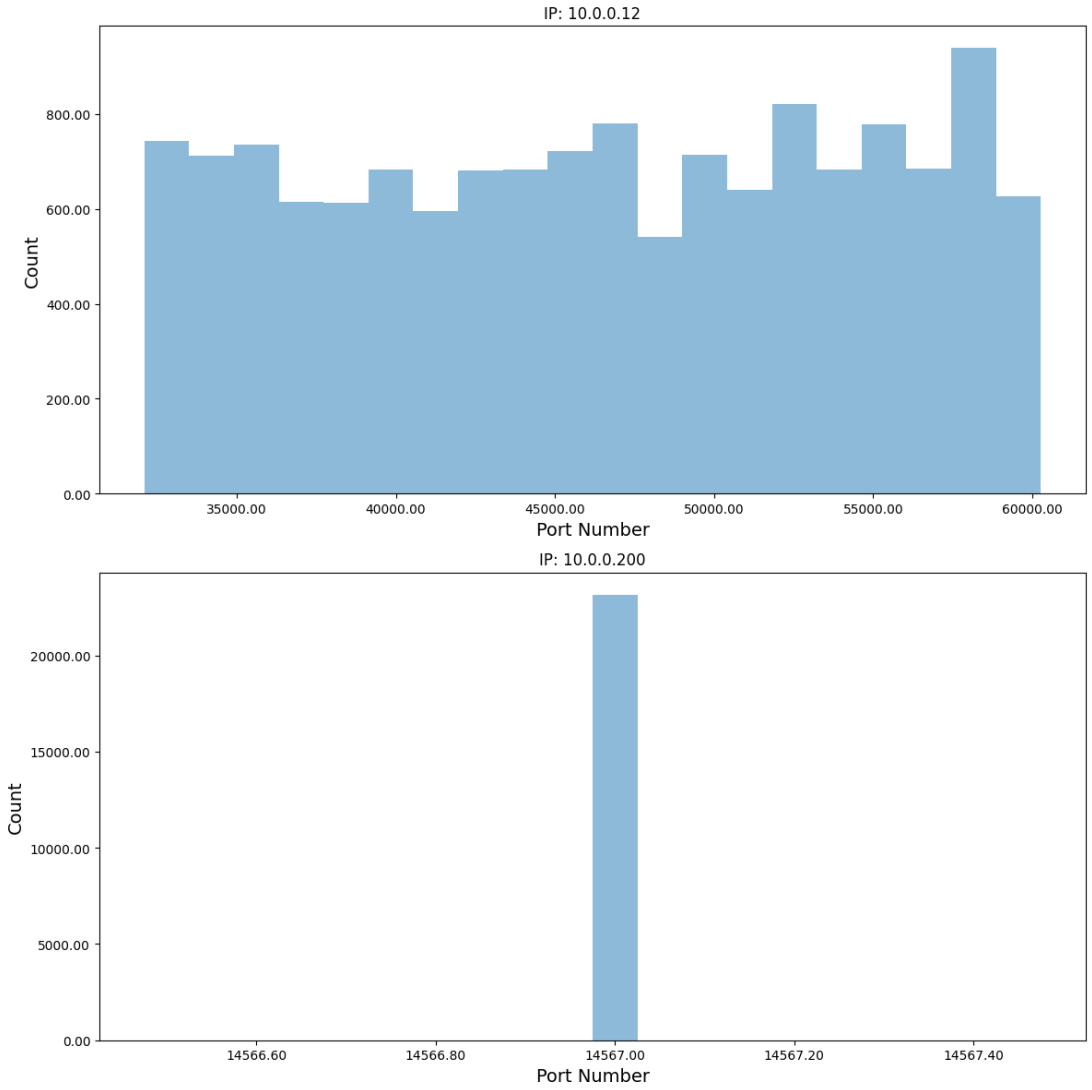

図 8: 実験 1-1: 0 packets/秒の時における送信元ポートの分散

100 packets/秒の時の送信元ポートの分散を図 8 に示す。縦軸は、特定のポー

ト範囲におけるパケットの数を示している。横軸は、送信元ポート番号を示している。IP アドレス、10.0.0.12 は正当なクライアントの IP アドレスである。IP アドレス、10.0.0.11 は攻撃者の IP アドレスである。IP アドレス、10.0.0.200 はブローカーの IP アドレスである。

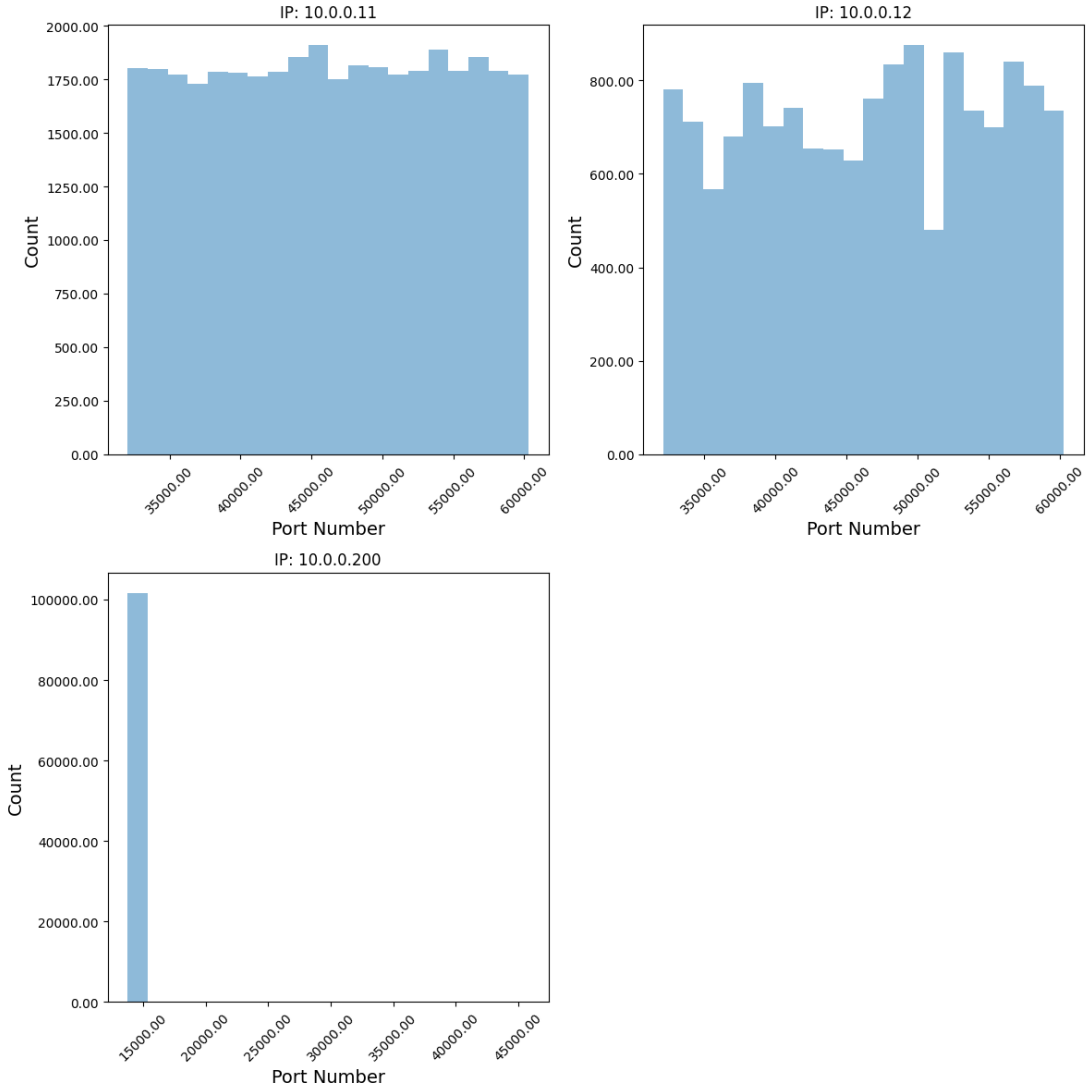

図 9: 実験 1-1: 100 packets/秒の時における送信元ポートの分散

1000 packets/秒の時の送信元ポートの分散を図 10 に示す。縦軸は、特定のポート範囲におけるパケットの数を示している。横軸は、送信元ポート番号を示して

いる。IP アドレス、10.0.0.12 は正当なクライアントの IP アドレスである。IP アドレス、10.0.0.11 は攻撃者の IP アドレスである。IP アドレス、10.0.0.200 はブローカーの IP アドレスである。

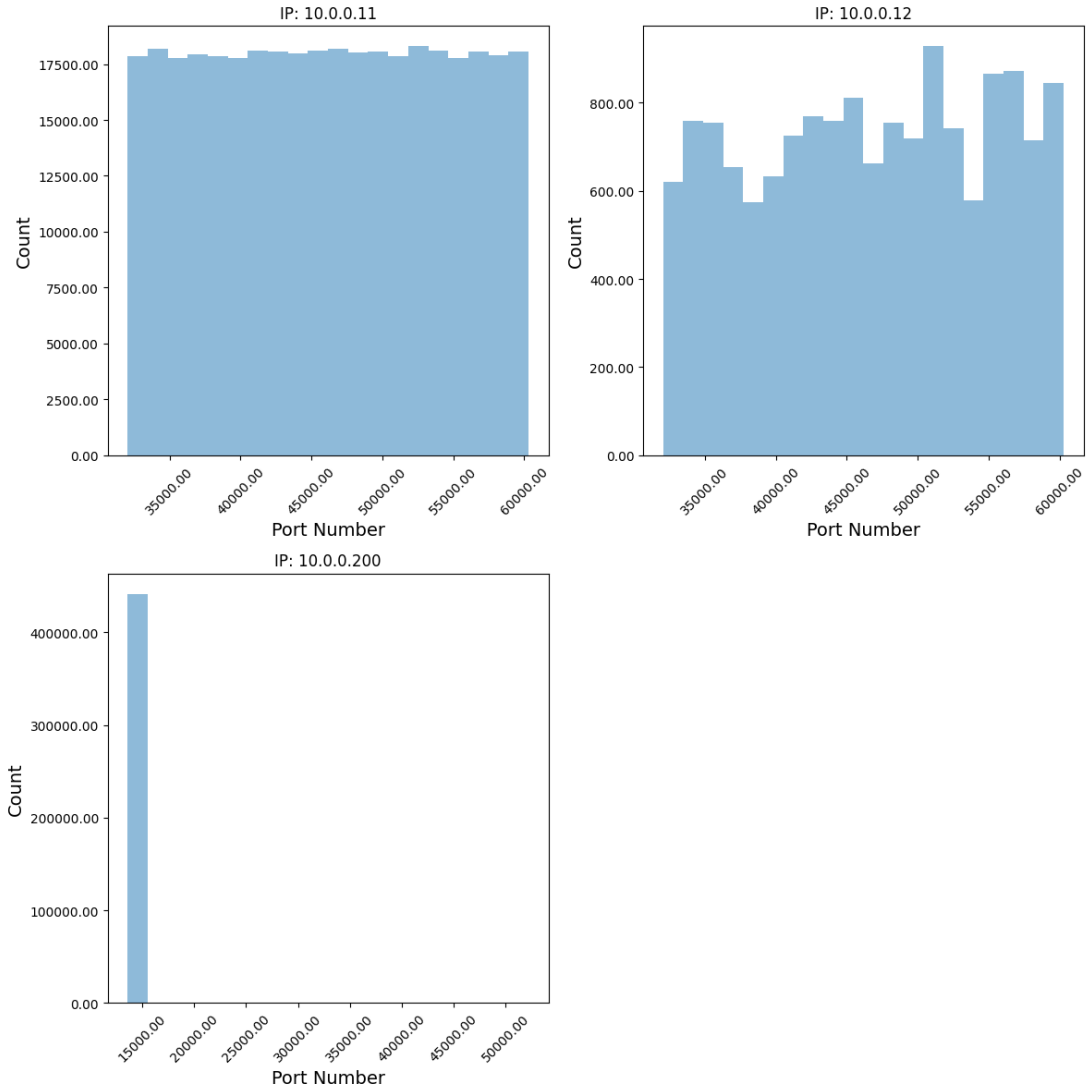

図 10: 実験 1-1: 1000 packets/秒の時における送信元ポートの分散

10000 packets/秒の時の送信元ポートの分散を図 11 に示す。縦軸は、特定のポート範囲におけるパケットの数を示している。横軸は、送信元ポート番号を示している。IP アドレス、10.0.0.12 は正当なクライアントの IP アドレスである。

IP アドレス、10.0.0.11 は攻撃者の IP アドレスである。IP アドレス、10.0.0.200 はブローカーの IP アドレスである。攻撃者は、送信元ポートをランダムに選択していることがわかる。正当なクライアントも、送信元ポートをランダムに選択していることがわかる。

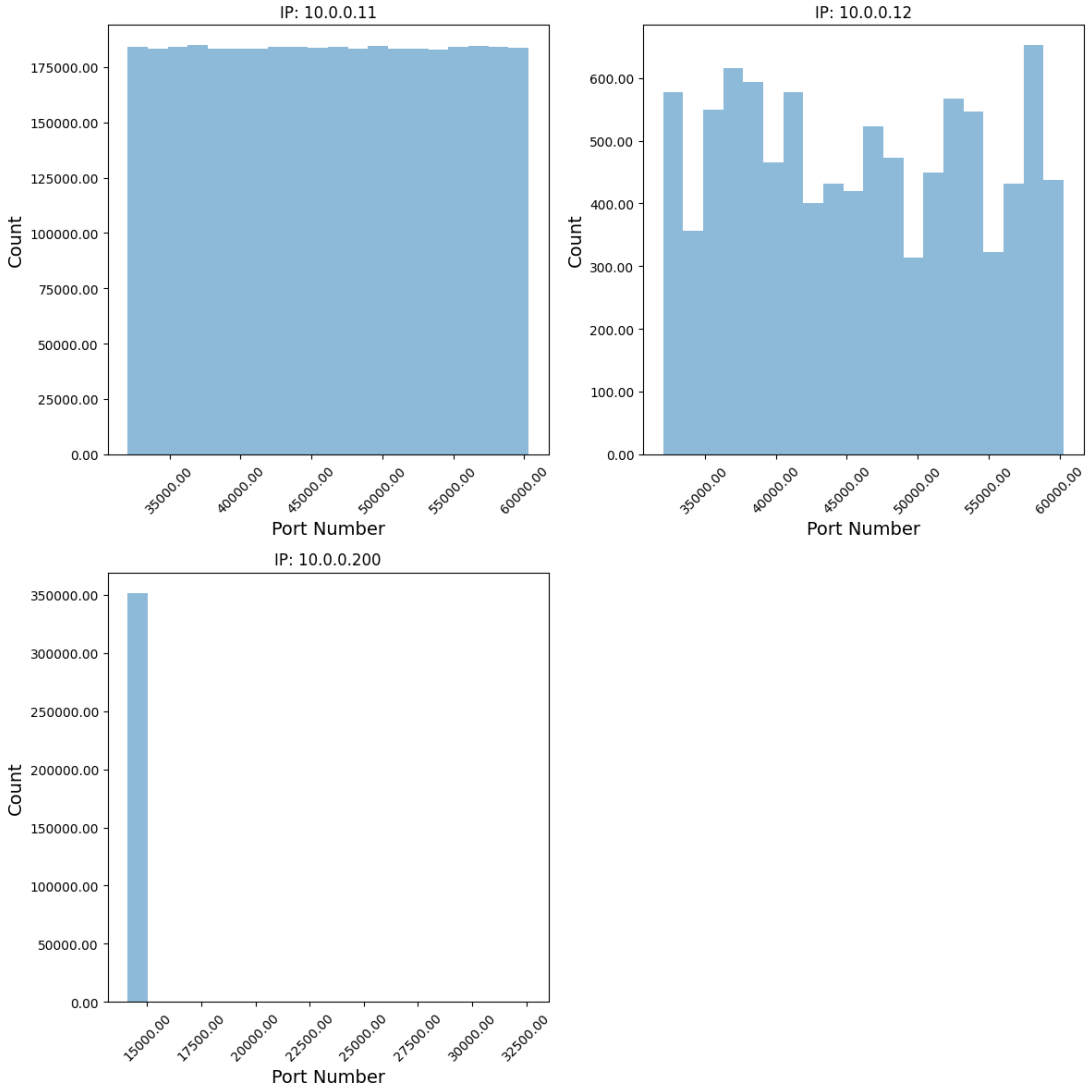

図 11: 実験 1-1: 10000 packets/秒の時における送信元ポートの分散

15000 packets/秒の時の送信元ポートの分散を図 12 に示す。縦軸は、特定のポート範囲におけるパケットの数を示している。横軸は、送信元ポート番号を示

している。IP アドレス、10.0.0.12 は正当なクライアントの IP アドレスである。IP アドレス、10.0.0.11 は攻撃者の IP アドレスである。IP アドレス、10.0.0.200 はブローカーの IP アドレスである。攻撃者は、送信元ポートをランダムに選択していることがわかる。正当なクライアントも、送信元ポートをランダムに選択していることがわかる。

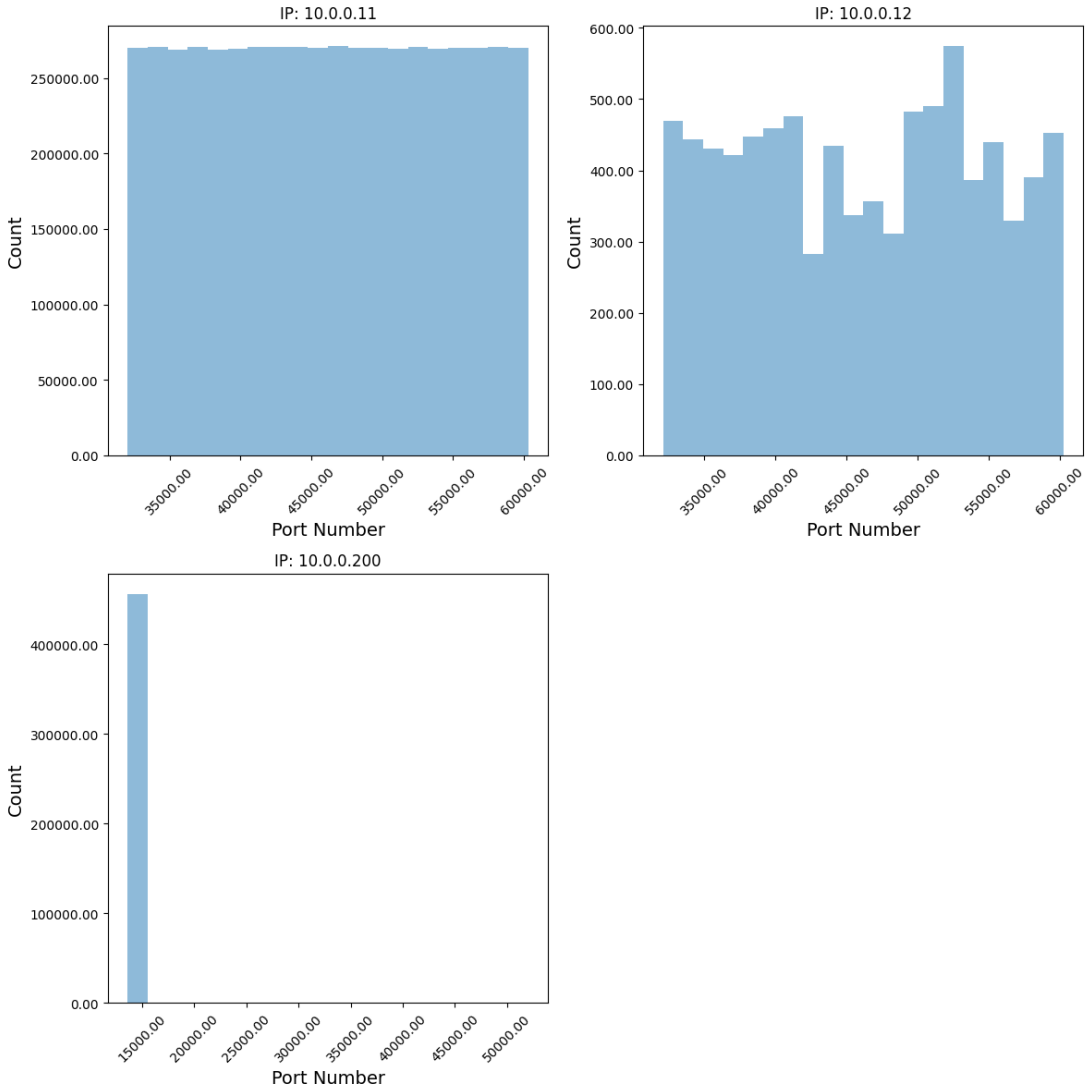

図 12: 実験 1-1: 15000 packets/秒の時における送信元ポートの分散

5.3.2 実験 1-2 の結果

成功率は、Subscribe の成功数/Publish の試行回数で算出した。実験における実際の攻撃レートは、図 13 のようになった。縦軸は、攻撃者から送信された 1 秒あたりの Initial パケット数を示している。横軸は、実験開始からの経過時間を示している。

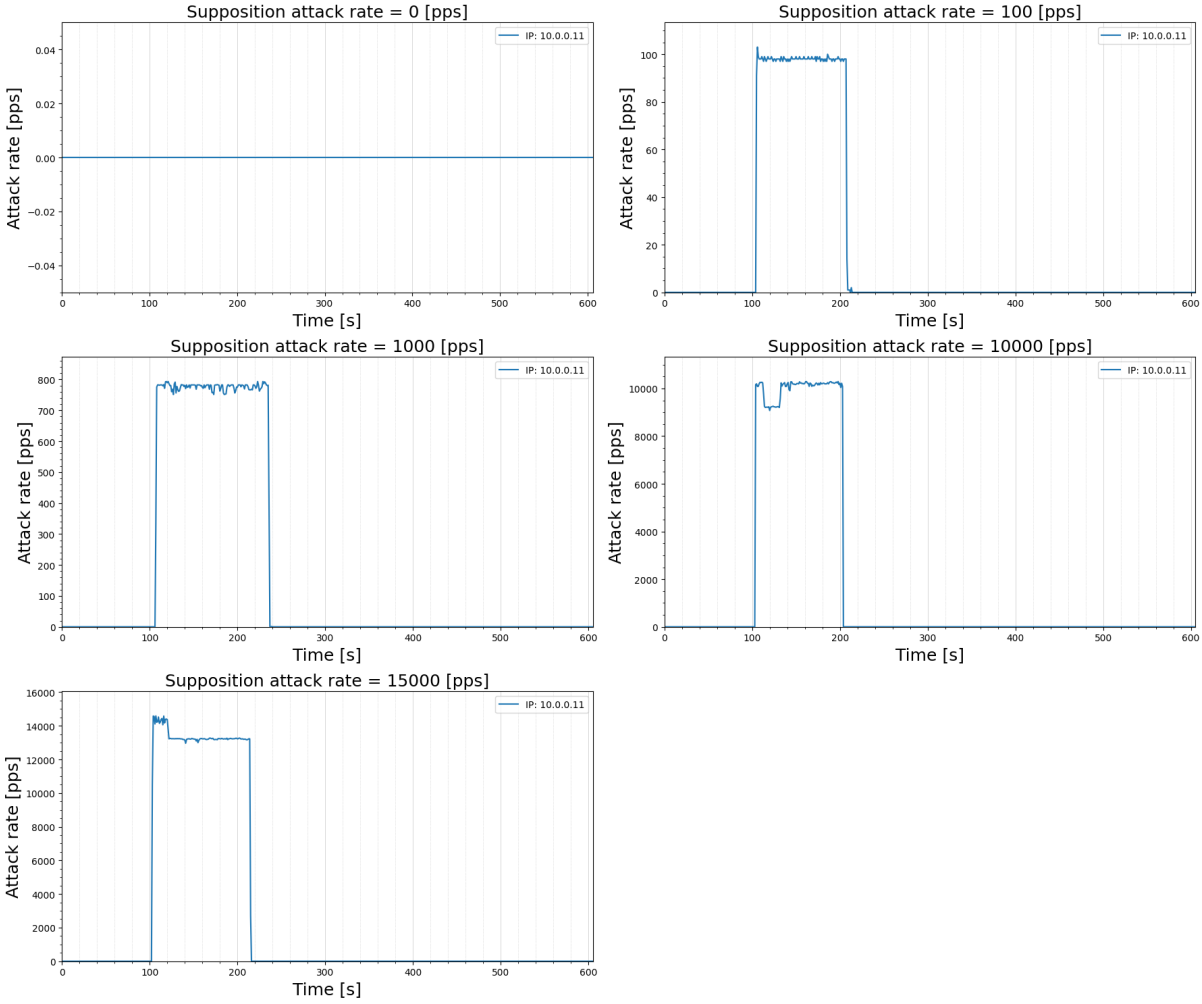

図 13: 実験 1-2: MQTT over QUIC ブローカーへの Initial 攻撃のレート

以下に示す表 4 は、実験 1-2 の攻撃レート別の成功率を示したものである。攻撃レートが増加するにつれて、成功率が低下していることがわかる。

表 4: 実験 1-2 での Pub/Sub の成功率

攻撃レート [**pps**]	**成功率** [%]
0	100
100	100
1000	99.96
10000	85.7
15000	80.0

5.4 攻撃に対する防御手法

実験 1-1、実験 1-2 の結果から、Initial 攻撃が MQTT システムの可用性に大きな影響を与えることが確認された。特に、Pub/Sub 操作の成功率が攻撃下で著しく低下することが観察された。防御手法を提案する先行研究 [12] では、QUIC の暗号処理部分を Network Acceleration Complex (NAC) などのハードウェアに特定の処理やタスクを移動させることでパフォーマンスが向上することを示している。これは Initial 攻撃の回避に有効であるが、ハードウェアに依存するこの手法は、特定のハードウェア構成が必要となり、実装が困難な場合もある。

5.4.1 実験 2-1: レートリミットの設定をした実験

ブローカー入力トラフィック全てについて、レートリミットを設定し、MQTT over QUIC ブローカーに対する Initial 攻撃への防御効果を検証する実験を行った。この実験は、攻撃に対するブローカーの耐性を評価し、効果的な防御手法を確立することを目的としている。

攻撃レートは 15000 packets/秒とし、レートリミットの効果を検証するために、異なる帯域幅の制限を試した。帯域制限は tc コマンドを用いて設定し、以下の 4 つの異なるレートリミットを適用した。

- 0.976 Mbytes/秒
- 9.76 Mbytes/秒
- 48.8 Mbytes/秒
- 73.2 Mbytes/秒

これらのシナリオは、ブローカーの処理能力に応じて、攻撃パケットの流入速度を調整し、ブローカーの性能への影響を最小限に抑えることを目的としている。実験では、これらの設定によるブローカーの耐性と性能の変化を詳細に観察し、どのレベルの帯域制限が最も効果的であるかを分析した。
以下の手順に従って実験を行った。実験前には時刻同期を行った。

1. ブローカーで、top、vmstat、netstat、tcpdump の記録を開始する (10 分間)。
2. 記録開始から 10 秒後に Initial 攻撃を開始する。Initial 攻撃は、6 分間継続させる。
3. 記録開始から 30 秒後、Publish を開始する (Publish を 0.1 秒に 1 回、合計 1000 回行う)。

5.4.2 実験 2-2:Retry の閾値を変化させた実験

本研究では、MQTT over QUIC ブローカーに対する Initial 攻撃への防御効果を評価するために、Retry の閾値を変えて実験した。この実験の目的は、異なる閾値設定がブローカーの耐性に与える影響を観察し、最適なパラメータ設定を探求することである。

実験は以下の 2 つのシナリオで構成されている。

- **シナリオ A**: ここでは、`retry_memory_limit` を 6 と設定し、この閾値がブローカーの耐性に与える影響を検証する。
- **シナリオ B**: このシナリオでは、`retry_memory_limit` を 6553 と設定し、より高い閾値がブローカーの性能に及ぼす影響を観察する。

これらのシナリオは、セッション情報を保持するために割り当てるメモリ量を最適化し、ブローカーの性能への影響を最小限に抑えることを目的としている。実験結果から、実験条件における MQTT over QUIC ブローカーが Initial 攻撃に対して持つ防御能力の程度が明らかになり、セキュリティ対策の最適化と攻撃への効果的な対応策の策定が可能となる。

以下の手順に従って実験を行った。実験前には時刻同期を行った。

1. ブローカーで、top、vmstat、netstat、tcpdump の記録を開始する (10 分間)。
2. Publish の試行を開始する (5 分間、0.1 秒に 1 回、実行する)。

3. 記録を開始し、Publish の試行が開始された後、Initial 攻撃を開始する。

4. Initial 攻撃は、1 分 40 秒継続される。

5.4.3 実験 2-3: セッションを再利用する実験

本研究では、MQTT over QUIC ブローカーに対する Initial 攻撃の脅威を軽減するための防御策として、セッションの継続による効果を検証した。セッション継続による防御メカニズムの可能性を探求し、その効果を実証することが目的である。pynng-mqtt-v0.1 を用いて、MQTT over QUIC ブローカーに対する Initial 攻撃の脅威を軽減するための防御策として、セッションの継続による効果を検証した。
予備実験として、セッションを張り続けた時の様子を攻撃なしで観察する実験を行った。以下の手順に従って実験を行った。実験前には時刻同期を行った。

1. ブローカーで、top、vmstat、netstat、tcpdump の記録を開始する (10 分間)。

2. Publish を開始する (Publish を 0.1 秒に 1 回、合計 1000 回行う)。

セッションを張り続けた時、Publish を行っている時に攻撃が行われる場合に攻撃を回避することはできるかを検証する実験を行った。以下の手順に従って実験を行った。実験前には時刻同期を行った。

1. ブローカーで、top、vmstat、netstat、tcpdump の記録を開始する (10 分間)。

2. Publish の試行を開始する (5 分間、0.1 秒に 1 回、実行する)。

3. ブローカーの記録開始後、1 分 40 秒経過後、Initial 攻撃を開始する。

4. Initial 攻撃は、1 分 40 秒継続される。

実験結果から、セッション継続により、MQTT over QUIC ブローカーが Initial 攻撃に対して持つ防御能力の程度が明らかになり、セキュリティ対策の最適化と攻撃への効果的な対応策の策定が可能となる。

5.5 提案手法の結果

成功率は、Subscribe の成功数を Publish の試行回数で割ることで算出した。各実験シナリオにおける成功率を比較し、提案した防御手法の有効性を評価する。

5.5.1 実験 2-1 の結果

レートリミットを設定した時の結果

実験における実際の攻撃レートは、図 14 のようになった。縦軸は、攻撃者から送信された 1 秒あたりの Initial パケット数を示している。横軸は、実験開始からの経過時間を示している。

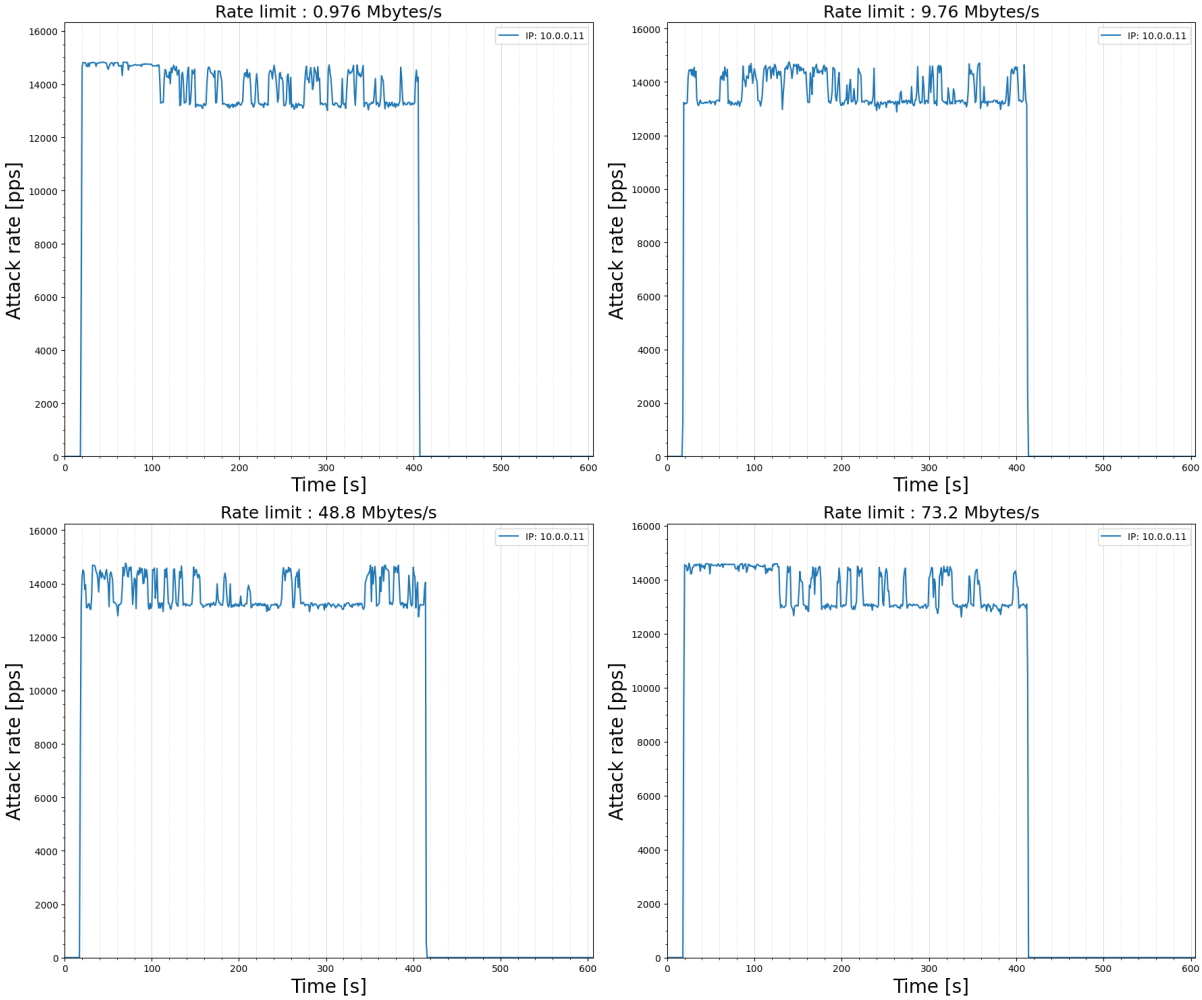

図 14: 実験 2-1: MQTT over QUIC ブローカーへの Initial 攻撃のレート

以下に示す表 5 は、実験 2-1 の帯域制限値別の成功率を示したしたものである。帯域の制限を緩和していったが、成功率に大きな差は見られなかった。

表 5: 実験 2-1 での Pub/Sub の成功率 (帯域制限と成功率の関係)

帯域制限 [**Mbytes/秒**]	**成功率** [%]
0.976	27.1
9.76	28.6
48.8	27.6
73.2	28.0

5.5.2 実験 2-2 の結果

Retry の閾値を変化させた実験の結果を示す。

シナリオ A(閾値=6) の実験における実際の攻撃レートは、図 15 のようになった。縦軸は、攻撃者から送信された 1 秒あたりの Initial パケット数を示している。横軸は、実験開始からの経過時間を示している。

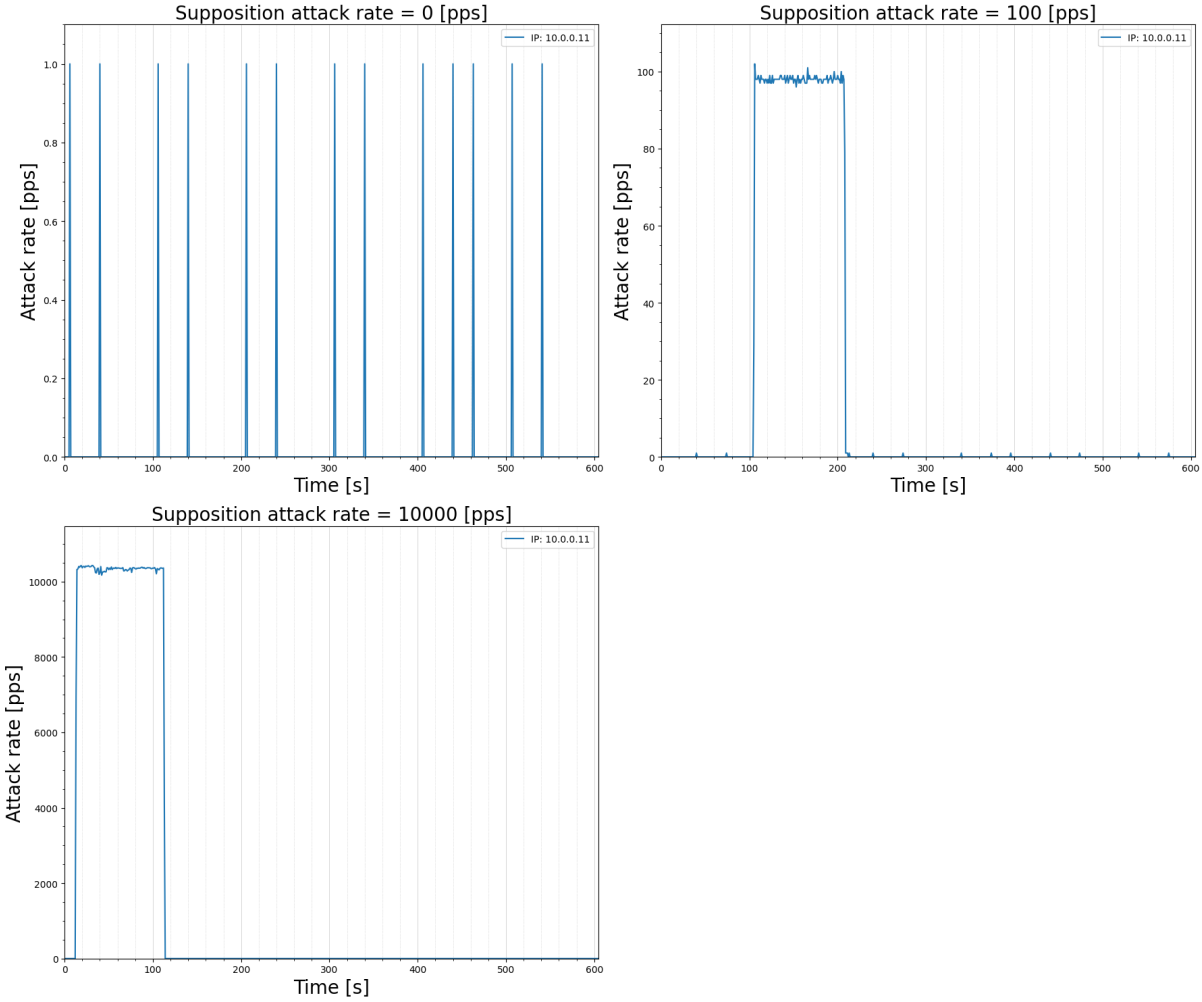

図 15: 実験 2-2-A: MQTT over QUIC ブローカーへの Initial 攻撃のレート

シナリオ B(閾値=65530) の実験における実際の攻撃レートは、図 16 のようになった。縦軸は、攻撃者から送信された 1 秒あたりの Initial パケット数を示して

いる。横軸は、実験開始からの経過時間を示している。

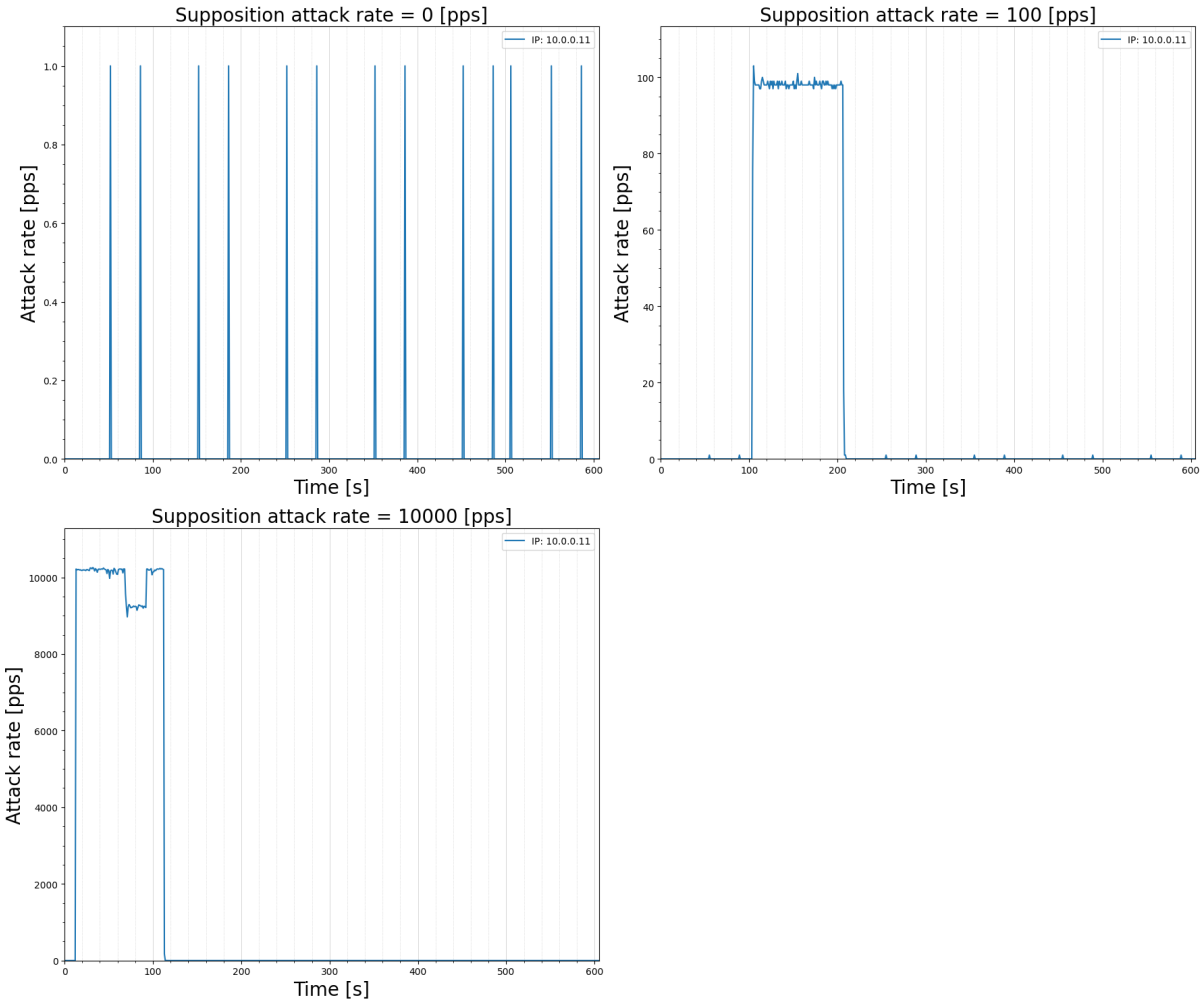

図 16: 実験 2-2-B: MQTT over QUIC ブローカーへの Initial 攻撃のレート

以下に示す表 6 と表 7 は、実験 2-2 の攻撃レート別の成功率を示したものである。閾値を変化させたことによる成功率の変化は見られなかった。

表 6: 閾値=6 のときの攻撃レートと Pub/Sub の成功率

攻撃レート [**pps**]	**成功率** [%]
0	100
100	100
10000	85.6

表 7: 閾値=65530 のときの攻撃レートと Pub/Sub の成功率

攻撃レート [**pps**]	**成功率** [%]
0	100
100	100
10000	86.0

5.5.3 実験 2-3 の結果

セッションを張り続ける実験の結果を示す。
実験における実際の攻撃レートは、図 17 のようになった。縦軸は、攻撃者から送信された 1 秒あたりの Initial パケット数を示している。横軸は、実験開始からの経過時間を示している。

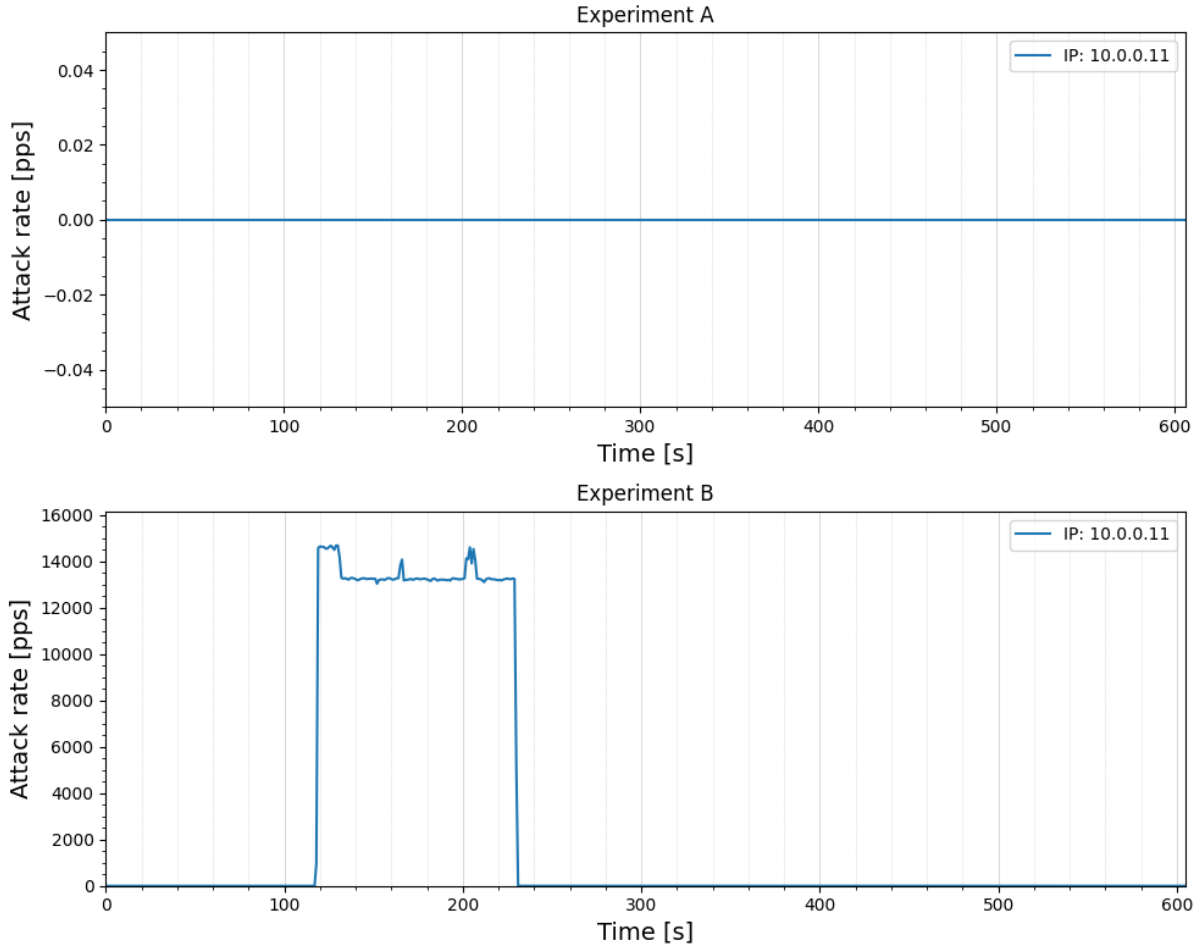

図 17: 実験 2-3: MQTT over QUIC ブローカーへの Initial 攻撃のレート

表 8 は、実験 2-3 の成功率を示したものである。攻撃下に関わらず、セッションを張り続けることで、成功率は保たれた。

表 8: 実験 2-3 での Pub/Sub の成功率

攻撃レート [pps]	**成功率** [%]
0	100
15000	100

ブローカーの CPU 利用率の遷移を図 18 に示す。縦軸は、ブローカーの CPU 利用率を示している。横軸は、実験開始からの経過時間を示している。

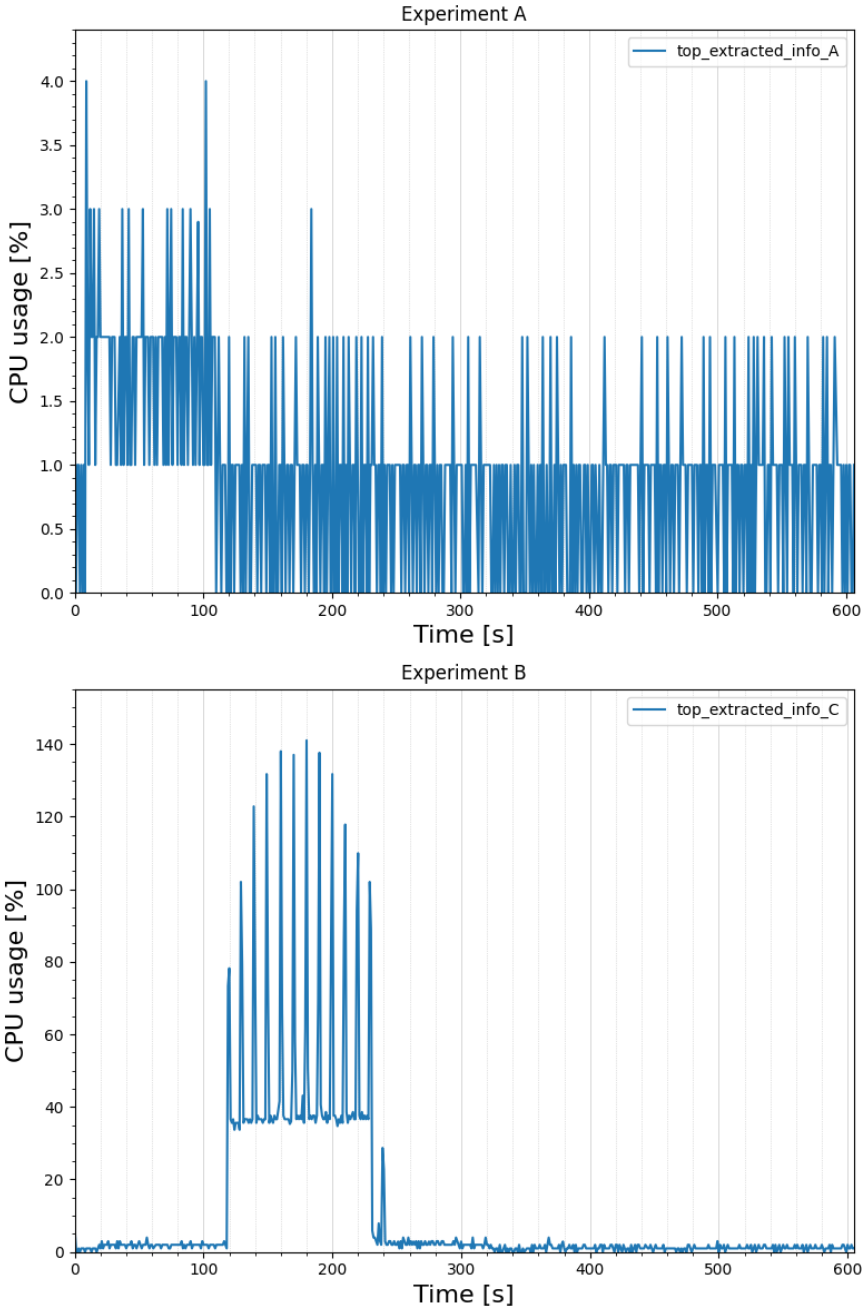

図 18: 実験 2-3 における CPU 利用率の遷移

実験中のブローカーが返しているトラフィック量の遷移について、図 19 に示す。縦軸は、実験中のブローカーが返しているトラフィック量を示している。横軸は、実験開始からの経過時間を示している。

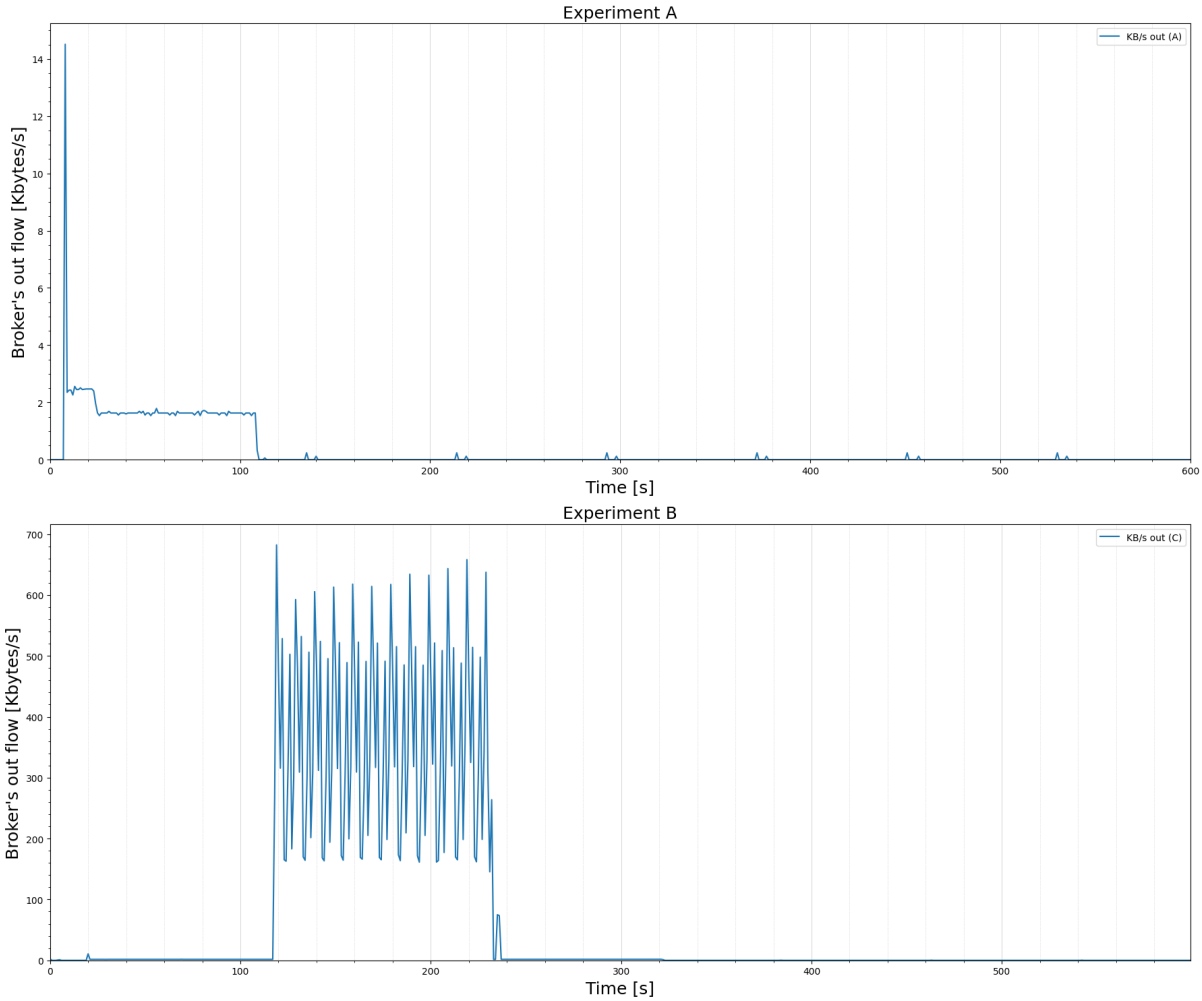

図 19: 実験 2-3 においてブローカーが返しているトラフィック量の遷移

シナリオ A の時の送信元ポートの分散を図 20 に示す。縦軸は、特定のポート範囲におけるパケットの数を示している。横軸は、送信元ポート番号を示している。IP アドレス、10.0.0.12 は正当なクライアントの IP アドレスである。IP アドレス、10.0.0.200 はブローカーの IP アドレスである。クライアントのポート番号は、固定されていることがわかる。

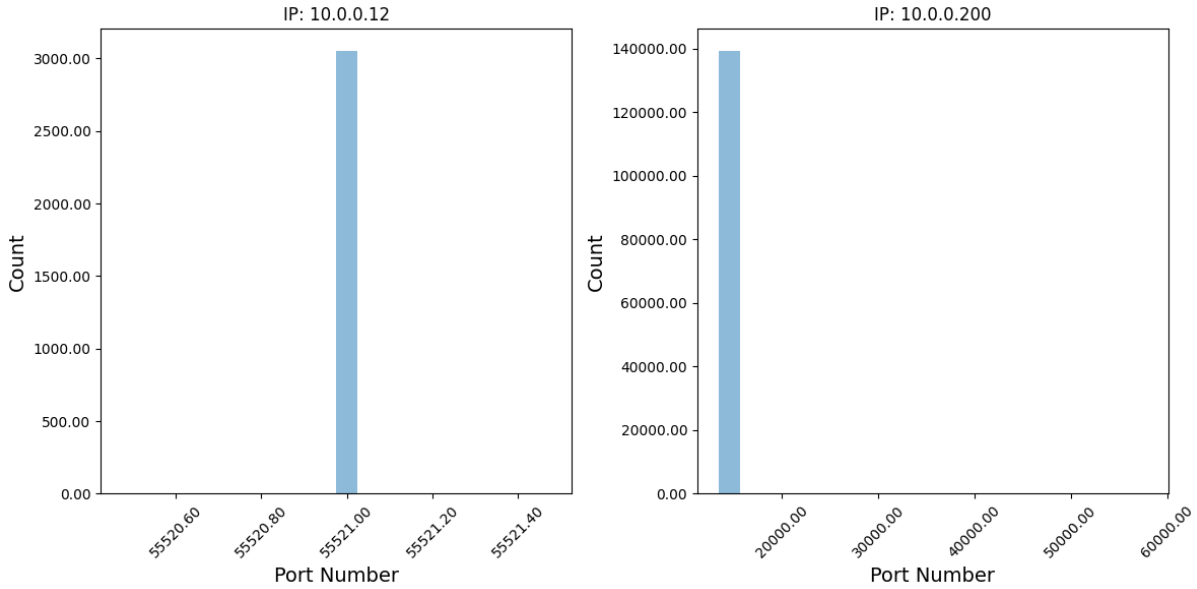

図 20: 実験 2-3-A における送信元ポートの分散

シナリオ B のときの送信元ポートの分散を図 21 に示す。縦軸は、特定のポート範囲におけるパケットの数を示している。横軸は、送信元ポート番号を示している。IP アドレス、10.0.0.12 は正当なクライアントの IP アドレスである。IP アドレス、10.0.0.11 は攻撃者の IP アドレスである。IP アドレス、10.0.0.200 はブローカーの IP アドレスである。クライアントのポート番号は、固定されていることがわかる。

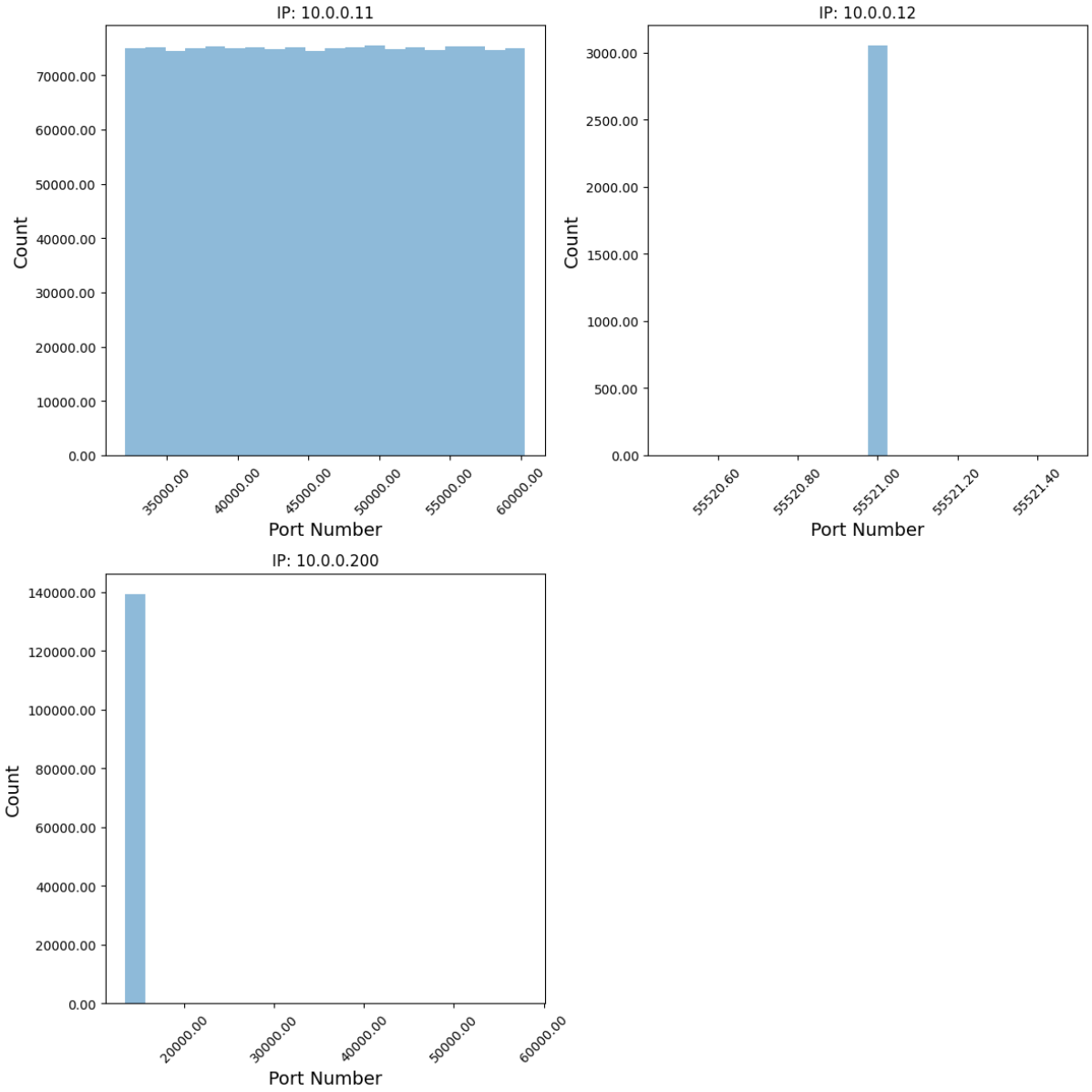

図 21: 実験 2-3-B における送信元ポートの分散

6. 考察

これら前章までの実験によって明らかになった主要な発見について論じる。また、研究過程で示された課題とその原因についての深い考察を行い、将来の研究の方向性についての示唆を提供する。これらの知見は、MQTT over QUIC 環境におけるセキュリティ強化と性能向上のための基盤となることが期待される。

6.1 研究の主要な発見

本研究は、MQTT over QUIC ブローカーへの Initial 攻撃がシステムの安定性に与える影響を明確に確認し、セッション継続による防御手法の有効性を実証した。Initial 攻撃が 10000pps 以上の攻撃レートで発生した場合、ブローカーの性能と応答性に影響を与えることが明らかになった。また、10000pps より小さい攻撃レートでは、ブローカーの性能と応答性に影響を与えなかったが、ブローカーの CPU 利用率を増加させた。

実験により、Initial 攻撃が MQTT over QUIC ブローカーの性能と安定性に影響を与えることが明らかになった。実験 1-1 と 1-2 では、Initial 攻撃がブローカーの性能と応答性に影響を与えることが明らかになった。実験 2-1 においてレートリミットを設定したが、この設定はブローカーの性能に影響を与えなかった。実験 2-2 において、Retry の閾値を 1/10 に設定した場合、100 倍に設定した場合のどちらも、ブローカーの性能に影響は見られなかった。実験 2-3 の結果から、セッション継続を用いた防御手法は、これらの攻撃に対して高い耐性を示した。しかし、セッション継続は、セッションハイジャック、リプレイ攻撃の可能性を高める可能性がある。セッション継続による防御メカニズムは、Initial 攻撃に対して有効であることが示された。

このことは MQTT over QUIC ブローカーの運用においてセッション継続を防御策の一つとして考慮することの重要性を示している。このアプローチにより、MQTT over QUIC ブローカーは Initial 攻撃による影響を効果的に軽減し、システムの安定性とパフォーマンスを維持することができた。これは、接続キューおよび Retry キューの飽和を防ぐことができたためであると考えられる。攻撃を受

けている際の MQTT over QUIC ブローカー CPU 利用率は、セッション継続を行なっている場合もそうでない場合も同程度であったからである。つまり、接続スロットに関する差分か生まれていると予想される。

リソースの消費の観点では、1000pps 程度での攻撃レートでリソースの消費が一定の値を示している。また、ブローカーが返すパケット数もこのあたりのレートで一定の値を示しているように見えた。一方で、Pub/Sub システムの可用性は 1000pps 程度の攻撃レートでは低下していない。また、実験 1-1 の条件において、1000pps の攻撃レートの時は Pub/Sub 成功時の遅延時間は、攻撃なしの時とほぼ変わりはなかった。しかし、10000pps、15000pps の攻撃レートの時は Pub/Sub 成功時の遅延時間が、攻撃なしの時と比べて、増加していた。

また、攻撃者の観点では、データセットの再生によってより効果的な攻撃実装が可能といった知見が得られた。この結果は、攻撃者が MQTT over QUIC ブローカーに対する Initial 攻撃を実装する際の実用的な手法を示唆している。また、このことから攻撃者が再生攻撃を行なっていると決め打つことでより効果的に防御側は対策を講じることができると考えられる。加えて、正常パブリッシャの送信元ポートの分散と攻撃者のパケット送信元の分散は同様の値を示した。これは、攻撃者が攻撃パケットを送信する際に正常なパブリッシャと同様のポートを使用できることを示唆している。この結果は、ポートのみに着目したやり方ではブローカーのフィルタリング機能で攻撃を回避することが難しいことを示唆している。

6.2 研究の限界と課題

本研究において、MQTT over QUIC ブローカーに対する Initial 攻撃とその防御策を探究したが、いくつかの限界と課題が明らかになった。セッション継続による防御手法は、効果的であるものの、リソース消費とセキュリティリスクという点で課題を抱えている。セッションハイジャックのリスクが増加する。これは、セッションが長期にわたって継続されると、攻撃者が認証情報やセッション ID を入手して、セッションを不正利用する可能性が高まるためである。また、リプレイ攻撃に対する脆弱性も増す傾向にある。長期間継続するセッションは、過去のメッセージを傍受し、それらを後で不正に再送するリプレイ攻撃の標的になりや

すいからである。

また、セッションを継続しない場合、メッセージの送信時以外は、クライアントのポートは解放されないため Initial 攻撃の対象となりにくい。しかし、セッションを継続することでクライアントのポートが固定されるため攻撃対象となる可能性が高くなる。その結果、ブローカーではなくクライアント側が Initial 攻撃の対象になり、クライアントのリソースを消費しバッテリーの急速な消耗を招き、デバイスの寿命が短期化するといった被害を受ける可能性がある。

また、本研究対象では QUIC プロトコルの特定の特性や新たな攻撃ベクトルに関する調査が求められる。例えば、ネットワーク遅延やパケット損失率の増加による影響は、実運用環境において重要であるため、これらの要因に関する影響の検証も要求される。

このように、MQTT over QUIC ブローカーに対する Initial 攻撃とその防御策は、特にリソースが制限された IoT 環境においてさらなる研究活動により個別の運用環境に最適な防御手法の採用が可能となることが期待される。将来の研究においては、これらのリスクと防御策に関する詳細な分析と評価が求められる。また、新たな攻撃ベクトルや環境要因についての研究も不可欠であり、MQTT over QUIC ブローカーの安全性と効率性をさらに高めるための基盤を築くことが重要である。

6.3 今後の展望

本研究を基に、MQTT over QUIC ブローカーに対する Initial 攻撃に耐えうる新たな防御策の開発が重要である。本研究では、セッションの再利用が有用なアプローチであることがわかった。MQTT over QUIC は、移動体通信網や IoT 環境など、リソースが制限された環境においても広く利用されることが期待されているが、運用要件に応じてセッション継続の実装を選択できるようにすることが重要である。そのために異なるネットワーク条件下での MQTT over QUIC ブローカーの性能評価や、新たな攻撃シナリオに対する耐性評価を行うことが必要であると考えられる。これらの研究は、MQTT over QUIC ブローカーの運用性と安全性をさらに高めるための基盤となる。

本研究で得られた知見は、IoT 分野における通信技術の発展に重要な影響を与える。IoT デバイス間の通信が増加し続ける現代において、MQTT over QUIC プロトコルの安定性とセキュリティを確保することは、社会全体のデジタルインフラの信頼性向上に直結する。したがって、本研究の結果は、IoT コミュニケーションの効率化と安全性の向上に対する提案を提供すると共に、IoT 技術の将来的な発展における重要な指針を示している。

7. おわりに

本研究の主な目的は、MQTT over QUIC プロトコルにおけるセキュリティ上の新たな課題を明らかにすることであった。これまで、MQTT over QUIC に関連するセキュリティ研究はほとんど行われておらず、特に Initial 攻撃に関する研究は不足していた。本研究では、MQTT over QUIC ブローカーに対する Initial 攻撃が与える影響とその防御策についての仮説を立て実験した。

研究の過程で、MQTT over QUIC ブローカーに対する Initial 攻撃が、CPU リソースを消費し、Pub/Sub 通信の失敗をもたらすことが明らかになった。よって、Initail 攻撃が、特にリソースが制限された IoT 環境において、ブローカーの運用性とセキュリティに大きな影響を与える可能性があることが示された。防御策としては、セッションの継続を通じてブローカーの耐性を高める方法が有効であることが示唆された。しかし、これはクライアント側の通信ポート番号を固定することになるので、セキュリティ上のリスクが高まる可能性がある。ただし、ブローカー側よりもクライアント側が公開されている範囲は狭いので、このリスクは低いと予想される。

本研究は、MQTT over QUIC のセキュリティ研究において新たな方向性を示し、IoT コミュニケーションの効率化と安全性の向上に寄与する。これらは、この分野における将来の研究と実践に実験結果を元にした価値ある貢献を提供する。本研究を通じて得られた MQTT over QUIC ブローカーへの Initial 攻撃と防御手法の提案の知見が、IoT 分野における通信技術の進化とセキュリティ確保に向けた重要な一歩となることを期待する。

謝辞

本研究を進めるにあたり、既存研究と比べた本研究の位置付けや、新規性についてご指導をいただいた、主指導教員である本学情報基盤システム学研究室の藤川和利教授に深く感謝いたします。副指導教員であり、研究の方向性について的確な意見をいただきました、本学大規模システム管理研究室の笠原正治教授に心より感謝いたします。副指導教員であり、本学情報基盤システム学研究室の新井イスマイル准教授には、実験方法の議論や攻撃の評価など細部にわたるご指導をいただきました。本研究を通じて、日々的確な指摘とサポートをいただいたことに、深く感謝いたします。特に、本学情報基盤システム学研究室の垣内正年助教には、実証実験における学内システムの利用や環境構築、攻撃実験の安全性に関する技術的なご指導を賜りました。また、研究過程で鋭く的確なアドバイスをいただきました。ここに、深く感謝申し上げます。さらに、遠藤新助教には、研究の核となる目的の明確化と最適な手法の検討において、深くご指導をいただきました。また、日頃から研究の進み具合を気にかけていただき、相談に乗っていただきました。ここに、深く感謝申し上げます。そして様々な面から研究活動を支援してくださいました、本学総合情報基盤センターの辻元理恵女史に心より感謝いたします。そして 2 年間の博士前期課程をともに過ごし、技術的・精神的にサポートをしてくれた本学情報基盤システム学研究室同期の眞田君、並びに本学情報基盤システム学研究室の学生の桂先輩、松永先輩、吉原君、合わせて既にご卒業された皆様に感謝申し上げます。最後に、経済面や生活面において多大な支援を頂いた家族に心より感謝いたします。本当にありがとうございました。

参考文献

[1] 経済産業省, “IoT 製品に対するセキュリティ適合性評価制度構築に向けた検討会中間とりまとめ,” 経済産業省, May 2023. [Online]. Available: https://www.meti.go.jp/shingikai/mono_info_service/sangyo_cyber/wg_cybersecurity/iot_security/pdf/20230515_1.pdf

[2] “MQTT Version 5.0,” OASIS Standard, OASIS, March 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[3] P. Kumar and B. Dezfouli, “Implementation and analysis of QUIC for MQTT,” *Computer Networks*, vol. 150, pp. 28–45, 2019. [Online]. Available: https://doi.org/10.1016/j.comnet.2018.12.012

[4] E. Atilgan, I. Ozcelik, and E. N. Yolacan, “MQTT security at a glance,” in *Proc. of the 2021 International Conference on Information Security and Cryptology (ISCTURKEY)*, 2021, pp. 138–142.

[5] J. Iyengar and M. Thomson, “QUIC: A UDP-based multiplexed and secure transport,” Request for Comments: 9000, Internet Engineering Task Force, May 2021. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9000.html

[6] H. Wong and T. Luo, “Man-in-the-middle attacks on MQTT-based iot using bert based adversarial message generation,” in *Proceedings of the 3rd International Workshop on Artificial Intelligence of Things (AIoT) in conjunction with KDD' 20 Workshops*, August 2020, pp. 1–7, [Online]. Available: https://github.com/HenryCWong/adversarialBERTMessages.

[7] D. Stiawan, M. Y. Idris, R. F. Malik, S. Nurmaini, N. Alsharif, and R. Budiarto, “Investigating brute force attack patterns in IoT network,” *Journal of Electrical and Computer Engineering*, vol. 2019, p. 4568368, 2019. [Online]. Available: https://www.hindawi.com/journals/jece/2019/4568368/

[8] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, "MQTTset, a new dataset for machine learning techniques on MQTT," *Sensors (Switzerland)*, vol. 20, no. 22, pp. 1–17, 2020.

[9] I. Vaccari, M. Aiello, and E. Cambiaso, "SlowITe, a novel denial of service attack affecting MQTT," *Sensors*, vol. 20, no. 10, p. 2932, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/10/2932

[10] F. Fernández, M. Zverev, P. Garrido, J. R. Juárez, J. Bilbao, and R. Agüero, "And QUIC meets IoT: Performance assessment of MQTT over QUIC," in *Proc. of the 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2020, pp. 1–6.

[11] M. Nawrocki, R. Hiesgen, T. C. Schmidt, and M. Wählisch, "Quicsand: Quantifying QUIC reconnaissance scans and DoS flooding events," in *Proceedings of the 21st ACM Internet Measurement Conference*, 2021, pp. 283–291.

[12] B. Cui, Z. Li, and F. Yu, "Manipulated client initial attack and defense of QUIC," in *Proc. of the 2022 IEEE 24th International Conference on High Performance Computing & Communications; 8th International Conference on Data Science & Systems; 20th International Conference on Smart City; 8th International Conference on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, 2022, pp. 611–618.

[13] "Network access control NAC market," Global Market Insights, 2022. [Online]. Available: https://www.gminsights.com/industry-analysis/network-access-control-nac-market