IoTデバイスの通信セキュリティ向上のための ホームネットワーク仮想化フレームワークの提案

塚崎 拓真[1]　滕 睿[2]　佐藤 健哉[1]

概要：近年，IoT(Internet of Things) が注目を集めるようになり，今後あらゆるモノがネットワークに接続され，利用されることが予想される．しかし，IoT の発展により利便性が高まる一方で，これまでネットワークに接続されていなかったモノが接続されることにより，セキュリティ上のリスクも高まっている．また，今後はホームネットワーク内で閉じたデバイス間の通信によって連携を行う形になることが想定される．デバイス間で直接通信を行う場合，各デバイスにおいてどのデバイスとの通信を受け入れるか，アクセス制御を行う必要がある．そこで本研究では，SDN(Software Defined Networks) の代表的プロトコルである OpenFlow を用いて，ホームネットワーク内の通信を監視するフレームワークの構築を検討した．また，提案システムでは，セキュリティ対策を適用可能な Proxy を仮想的に作成した．この Proxy にセキュリティ対策をオフロードし，IoT デバイス間の通信を中継することで，本来 IoT デバイスに適用したいセキュリティ対策を実現した．そして，IoT デバイス間で閉じた通信を行うシミュレーションの評価を行い，ホームネットワークにおいてセキュリティ要件を保つことを示した．

Proposal of Home Network Virtualization Framework to Improve Communication Security of IoT Devices

TAKUMA TSUKASAKI[1]　RUI TENG[2]　KENYA SATO[1]

1. はじめに

近年，IoT(Internet of Things) が注目を集めるようになり，今後あらゆるモノがネットワークに接続され，利用されることが予想される．あらゆるモノが IoT によりネットワークに接続されることで，情報流通を促進し，ビッグデータを収集，解析が行われることに加え，ローカルネットワーク内においても，今後様々な IoT デバイスの登場により，今まで相互接続されていなかった機器同士が接続されると考えられる．以上のような変化により，様々な課題の解決や新たな価値の創出が期待されている．

しかし，IoT の発展により利便性が高まる一方で，これまでネットワークに接続されていなかったモノが接続されることにより，セキュリティ上のリスクも高まっている [1]．IoT デバイスは十分なセキュリティを考慮せずに開発されたものが多いため，悪意のある攻撃者によるサイバー攻撃の標的になりやすい．脅威としては，ホームネットワークに侵入し，デバイスの遠隔操作による外部サーバへの攻撃やマルウェア感染によるプライバシーに関わる機密情報の収集などが挙げられる．また，攻撃によってホームネットワーク内に侵入されてしまうと，その内部でも端末間で自由にアクセス可能なため，マルウェア感染などのリスクがホームネットワーク全体の端末に広がる可能性がある．したがって，ホームネットワークのセキュリティ対策として，侵入を前提に攻撃を受けた時に被害を最小化できることが重要である．しかし，IoT デバイスは従来の PC 等の既存機器と比較した場合，CPU 等のリソースを十分に保持していないため，デバイスの計算能力の制限やソフトウェア自体の脆弱性によって，適用できる機能が限られるという問題がある．そのため，暗号化等のセキュリティ対策の適用は困難となり，どのデバイスも必ず利用するネットワークを利用したシステムを構築することや仮想的に作成した

[1] 同志社大学大学院 理工学研究科
Graduate School of Science and Engineering, Doshisha University
[2] 同志社大学モビリティ研究センター
Mobility Reserch Center, Doshisha University

システムを構築することが望ましい．

そこで本研究では，SDN(Software Defined Networks) の代表的プロトコルである OpenFlow[2] を用いて，ホームネットワーク内の通信を監視し，また，セキュリティ対策を施し仮想的に作成した Proxy を利用し，IoT デバイスに対してセキュリティ対策を施すフレームワークの構築を検討した．

2. ホームネットワークの問題点

IoT デバイスが普及し，スマートホーム等の考えが生まれると，アンチウイルスソフトやファイアウォール等のエンドポイントにおけるセキュリティ対策だけでは困難である．これは，カメラやスマート家電等の IoT デバイスでは，処理能力が低く，エンドポイントセキュリティ対策に求められる要件を満たさないためである [3]．さらに，独自に組み込み用 OS が使われているものもあり，それら全てに対応したシステムの構築や更新を続けるのは非常にコストが高い．そのような状況の中，脆弱なパスワードでの侵入やデータのプライバシー保護が不十分であることや，安全でないデータの転送等の IoT デバイスのセキュリティ対策不足が挙げられる [4]．上記の脆弱性から，IoT デバイスが他のデバイスへの感染や攻撃に悪用され，侵入や感染などの被害によってデータ流出や外部サービスへの攻撃などの恐れがあるため，侵入を前提に考えなければならない．このことから，ホームネットワークセキュリティ対策として必要なことは，侵入感染後の被害の最小化であると考えられる．

3. 関連研究

3.1 ネットワークレベルの攻撃検知・防止

Sivanathan らは，SDN と外部の解析エンジンを用いて，IoT デバイスのネットワークを常に監視し，フローレベルでのトラフィック検査を提案した [5]．パケットベースのネットワーク監視と比較し，処理コストの大幅な削減を実現した．

しかし，ホームネットワーク内のトラフィック情報検査を外部で行っていることが問題点として挙げられる．今後の IoT デバイスは、ホームネットワーク内で閉じたデバイス間の通信によって、デバイス間の連携を行う形になることが想定される [6]．デバイス間で直接通信を行う場合，各デバイスにおいてアクセス制御を行う必要がある．しかし，全てのデバイスがアクセス制御に対応しているとは限らず，デバイスの計算能力やソフトウェア自体の脆弱性によってアクセス制御が機能しない場合が考えられる [7]．

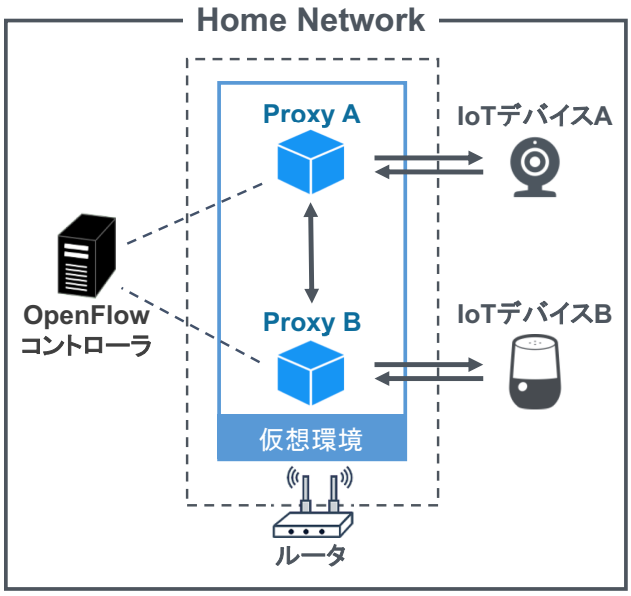

図 1　提案システムの構成

3.2 ホームネットワーク運用の外部依存を避けた自己完結型システム

Zhang らは，クラウド上の遠隔サーバから制御されている現状のホームネットワークの問題点を挙げ，ホームネットワークシステムを提案した [8]．提案システムでは、トラストアンカーをローカルコントローラに置くことで、デバイスやアプリケーションが全てローカルで管理され、データ検索等の承認や制御もホームネットワークシステム内で行われる。

しかし，各 IoT デバイスに対応したセキュリティ対策を施す柔軟性を持ち合わせていない．ホームネットワーク内には異なる規格のハードウェアや様々なアプリケーションが混在しているため，各デバイスに柔軟に対応できるシステムを構築することが望ましい．

4. 提案システム

4.1 概要

提案システムでは，セキュリティ対策を適用可能なデバイスを Proxy と定義し，ルータ上に仮想的に作成する．ここに，IoT デバイスがリソース量の制限により適用できないセキュリティ対策をオフロードし，この Proxy が IoT デバイス間の通信を中継することで，本来 IoT デバイスに適用したいセキュリティ対策を実現する．また，OpenFlow の機能も追加し，ネットワークの監視を行い，ホームネットワーク内通信のトラフィック情報は既知であることを考慮し，フローの検証を OpenFlow コントローラで行う．詳細なセキュリティ対策については後述する．

4.2 システム構成

提案システムの構成を図 1 に示す．本提案システムの構

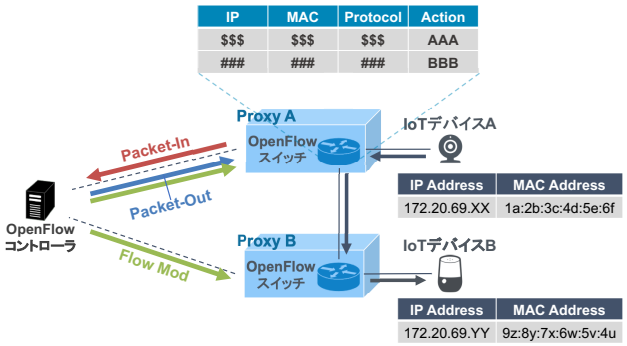

図 2　OpenFlow におけるフローチェック

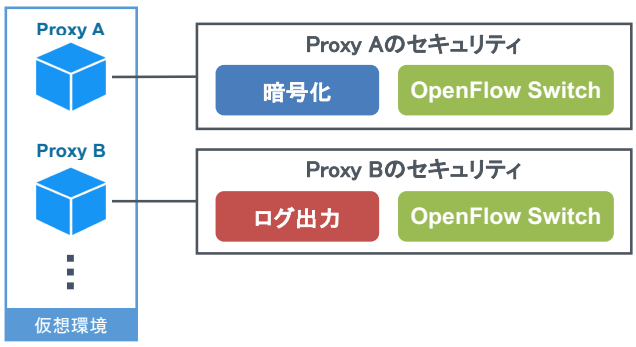

図 3　Proxy のセキュリティ対策

成要素は，IoT デバイス，Proxy，ルータ，仮想環境から構成される．

- IoT デバイス
  本研究で扱う IoT デバイスは，CPU 等のリソースを十分に保持しておらず，直接セキュリティ対策を適用できないデバイスと定義する．
- Proxy
  IoT デバイスに要求されるセキュリティ対策を，仮想的に実現したものである．IoT デバイスからの通信を中継し，セキュリティ対策を適用する．各 IoT デバイスに必要なセキュリティ対策をそれぞれ作成・適用することで、対象デバイスに応じた必要な対策を実現できる．
- ルータ
  IoT デバイス間通信の中継機器として用いる．ルータ上にコンテナを生成する．
- 仮想環境
  Proxy の実行環境である．Proxy が作成される際に要求されるリソースを十分に提供することが可能である．

4.3　OpenFlow によるフローチェック

本研究におけるネットワーク監視を OpenFlow を用いて行う．OpenFlow によるフローチェックを図 2 に示す．一つの IoT デバイスに対し，Docker イメージからコンテナ上に OpenFlow スイッチの機能を生成する．IoT デバイスはこの OpenFlow スイッチを中継し，デバイス間通信を行う．OpenFlow コントローラは事前に IoT デバイスの情報を保持しており，OpenFlow スイッチとの通信が確立次第、デバイス間通信のフローテーブルを作成する．ホームネットワークの特性である各 IoT デバイスのトラフィック情報は既知であり，変化が大きくないことを考慮し，IP アドレスや通信頻度の確認を行い，フローレベルにおける異常の検知を行う．

4.4　Proxy のセキュリティ対策

本研究におけるセキュリティ対策の特徴として，Proxy

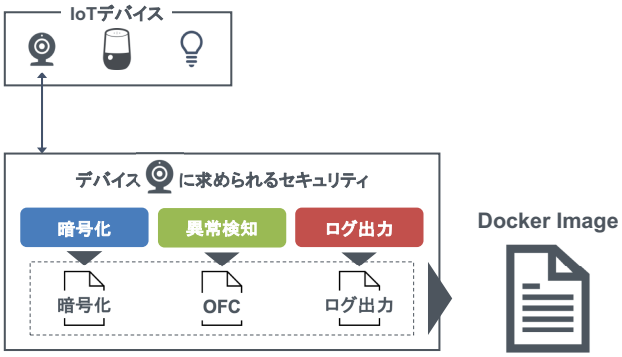

図 4　image ファイルの作成

ごとに異なるセキュリティ対策を適用可能なことが挙げられる．Proxy のセキュリティ対策の適用例を図 3 に示す．これによりリソースの都合上，IoT デバイスに直接適用できないセキュリティ対策を導入できることに加え，前述のホームネットワークの問題点で述べたような様々なハードウェアやアプリケーションへの適用や、様々なセキュリティ要件の変更に対しても柔軟に対応が可能となる．

また，コンテナ上で展開されるセキュリティ対策は、適応したいセキュリティ対策に対応したコンテナの image ファイルで定義される．image ファイルの作成図を図 4 に示す．この図のように IoT デバイスに対して適用したいセキュリティが複数ある場合においても，当該デバイスの規格に対応した対策をそれぞれ作成し，ソフトウェアモジュールのような形で組み合わせて定義することで，image ファイルを作成することが可能となる．

5.　実装

5.1　実装環境

本研究の実装環境，実装環境の構成をそれぞれ表 1 と図 5 に示す．Proxy の作成方法としては軽量なアプリケーション実行環境である Docker を利用した．Proxy を Docker で作成されるコンテナ上で稼働させることで、複数の Proxy をリソースやオーバーヘッドを抑えて作成できることに加え，Docker Hub より配布される Docker Image を用いることで容易に作成可能となる．また、OpenFlow コントロー

表 1　実装環境

種類	項目	説明
Proxy	使用ソフト	Docker
	OS	Ubuntu 20.04
	CPU	3.60GHz Intel Core i9
	メモリ	5GB
OpenFlow	使用ソフト	Ryu
	OS	Ubuntu 20.04
	CPU	3.60GHz Intel Core i9
	メモリ	5GB
	使用言語	Python

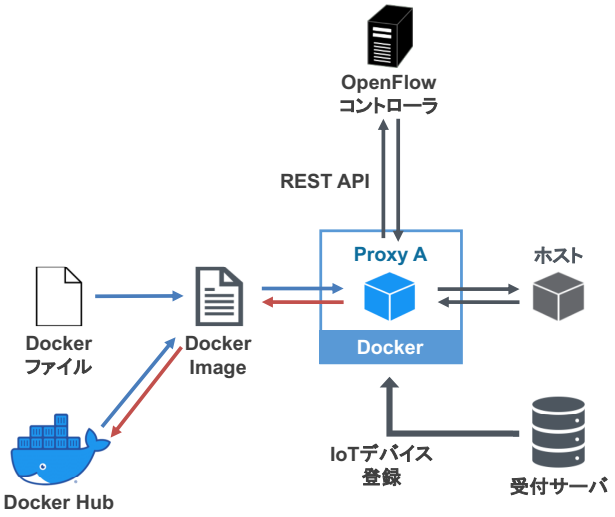

図 5　実装環境の構成

ラは Ryu を用いて作成した．

5.2　動作手順

IoT デバイスの所有者であるユーザが本提案システムを利用する際の動作手順を図 6 と以下に示す．

(1) ユーザはデバイスを LAN 内に接続した後，Proxy の受付サーバへアクセス．
(2) Proxy はホームネットワーク内に IoT デバイスが接続されたことを確認．
(3) Proxy は Docker Hub から Docker Image を取得．
(4) 取得した Docker Image を基に仮想環境内に Docker Image を作成．
(5) Proxy はデバイス情報を基に IoT デバイスに接続を行い，Proxy を経由して通信を行うように設定．
(6) Proxy は OpenFlow コントローラへデバイス情報を送信．
(7) 設定が完了し，IoT デバイス・Proxy 間の通信が確立された後，Proxy の作成・ネットワークの監視状況を報告．

5.3　想定ユースケース

本提案システムを用いた想定ユースケースを以下に示す．各 Docker イメージは Docker Hub というユーザが作成したコンテナをアップロードして公開・共有できるサービスを利用することを想定する．

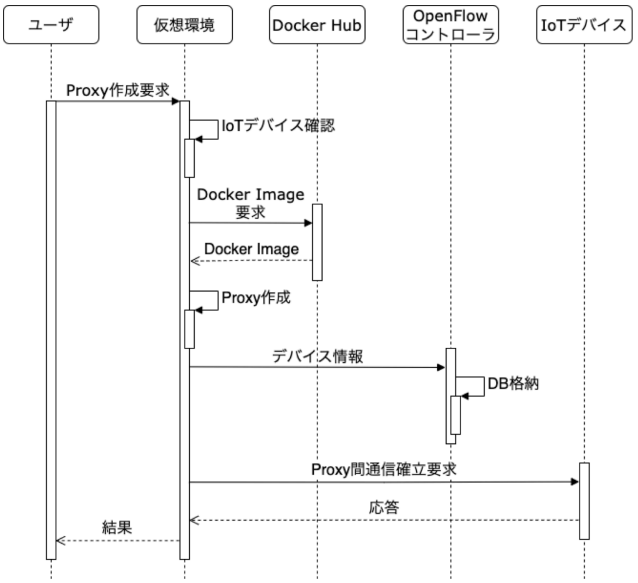

図 6　動作手順のシーケンス図

- セキュリティ対策を事前に提供する場合
  SSL による通信の暗号化など，IoT デバイスのリソースを多く利用するために適用できないセキュリティ対策を事前に Docker イメージとして提供し，Proxy 上で実現する．
- インシデント発生時などに対策を提供する場合
  事前に提供していたセキュリティ対策では想定していなかったインシデント等が発生した場合に，対象デバイスが保持するリソース量に依存せず，追加のセキュリティ対策を提供することが可能となる．

6.　評価

6.1　評価内容

本研究の評価として，まずセキュリティが適用されているかを検証した．今回は存在を把握していない IoT デバイスから通信があった場合と，あるデイバイスからの通信頻度が通常と異なる場合を想定した．それに対し，OpenFlow によるフローチェックが行われているかを検証した．

また，提案システムを適用した上で，IoT デバイス間で通信した際の EndtoEnd の通信時間の計測を行った．比較対象として，セキュリティ対策を適用せず，ルータを経由してデバイス間通信を行う場合についても計測を行った．

6.2　評価環境

今回適用するセキュリティ対策としては，鍵長が 1024bit の SSL による暗号化のイメージを Docker Hub より取得し，適用した．また，OpenFlow スイッチのイメージも取得し，OpenFlow によるフローチェックも行った．

```
*** s3 ----------------------------------------------------------------------
cookie=0x0, duration=41.578s, table=0, n_packets=11, n_bytes=1022, priority=1,in_port="s3-eth1",dl_dst=86:49:46:a5:a8:74 actions=output:"s3-eth2"
cookie=0x0, duration=12.852s, table=0, n_packets=10, n_bytes=924, priority=1,in_port="s3-eth2",dl_dst=4e:98:97:5f:fc:6e actions=output:"s3-eth1"
cookie=0x0, duration=89.081s, table=0, n_packets=73, n_bytes=9112, priority=0 actions=CONTROLLER:65535
```

```
*** s3 ----------------------------------------------------------------------
cookie=0x0, duration=6.029s, table=0, n_packets=0, n_bytes=0, priority=1,in_port="s3-eth1",dl_dst=86:49:46:a5:a8:74 actions=output:"s3-eth2"
cookie=0x0, duration=6.020s, table=0, n_packets=3, n_bytes=294, priority=10,in_port="s3-eth2",dl_dst=4e:98:97:5f:fc:6e actions=drop
cookie=0x0, duration=53.532s, table=0, n_packets=65, n_bytes=8258, priority=0 actions=CONTROLLER:65535
```

```
*** s3 ----------------------------------------------------------------------
cookie=0x0, duration=11.782s, table=0, n_packets=0, n_bytes=0, priority=1,in_port="s3-eth1",dl_dst=86:49:46:a5:a8:74 actions=output:"s3-eth2"
cookie=0x0, duration=59.285s, table=0, n_packets=65, n_bytes=8258, priority=0 actions=CONTROLLER:65535
```

図 7 登録済みのホストのフローテーブル (上)，異常時のパケットを Drop 処理するフローテーブル (中)，その後のアクションが削除されたフローテーブル (下)

7. 結果と考察

7.1 評価結果

登録済みの IoT デバイスから通信要求が来た場合，登録していない IoT デバイスや通常の通信頻度と異なる等の異常の通信がなされている場合のフローテーブルの結果を図 7 に示す．通常時は他のデバイスに対し，フローテーブルが作成されている一方で，異常時はパケットを Drop 処理するフローテーブルが作成されており，その後，そのフローテーブルが削除されていることがわかる．

また，セキュリティ対策を施した提案システムとセキュリティ対策を施してないシステムにおける通信の比較結果を図 8 に示す．僅かであるが、提案システムは、セキュリティ対策を施していないシステムにおける通信より、EndtoEnd の通信時間が劣っていることがわかる．

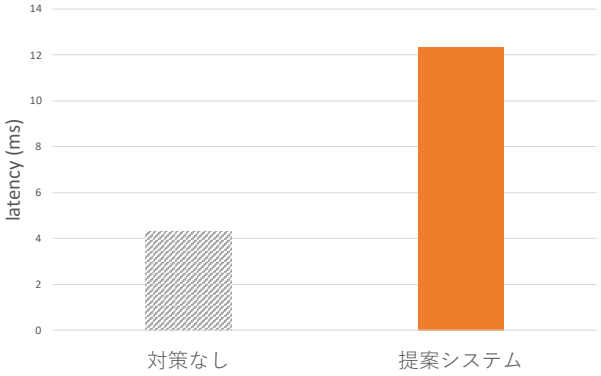

図 8 IoT デバイス間通信の EndtoEnd 遅延

7.2 信頼性に関する考察

IoT デバイスを用いたシステムの安心安全を確保するための機能として，IPA により IoT 高信頼化昨日が定義されており，IoT 高信頼化要件として，IPA により IoT 高信頼化要件として，開始，予防，検知，回復，終了の 5 つの局面に分けてそれぞれセキュリティ要件が定義されている [9]．今回は前述の 5 つより，システム稼働中の局面である予防，検知，回復の 3 つにおける高信頼化要件に対し，提案システムの有効性について考察する．

- 予防の局面における考察
  予防の局面での高信頼化要件は，稼働中の異常発生を未然に防止できることである．これに対応する IoT 高信頼化機能としては，ログ取集機能，暗号化機能等があり，以上の予兆の把握，資産の保護を実現する．提案システムを用いることで，リソース量の関係で通常の IoT デバイスに適用できない機能であっても適用可能となる．
- 検知の局面における考察
  検知の局面での高信頼化要件は，稼働中の異常発生を早期に検知できることである．これに対応する IoT 高信頼化機能としては，OpenFlow によるネットワーク監視機能，ログ収集機能があり，以上発生の検知や発生原因の特定を実現する．提案システムを用いることで，予防の局面同様，デバイスのリソース量に依存せず，求められる機能を実現できることに加え，Proxy は各 IoT デバイスごとに作成するため，個々のデバイスに応じた詳細な検知ルールを適用可能となる．
- 回復の局面における考察
  回復の局面での高信頼化要件は，異常が発生した場合に稼働の復旧ができることである．特に IoT では，さまざまなデバイスが相互通信を行うため，事前に予測していなかった異常が発生することが考えられる．今回の環境では各セキュリティ対策は Docker Hub を通して Docker イメージとして提供することで，事前に作成したセキュリティ対策だけでなく，追加のセキュリティ対策も配布・適用が容易である．

7.3 性能に関する考察

図より，本提案システムである Docker を用いた Proxy によるセキュリティ対策，OpenFlow によるネットワーク監視は，セキュリティ対策なしの場合と比較し遅延は発生しているが，許容範囲内であり，実用性があると考えられる．

今後 IoT デバイスが普及した際には，クラウドへの通信量が爆発的に増加するため，帯域の輻輳といった問題も発生すると考えられる．そのため，通信遅延はさらに増加す

ることが想定されることから，通信遅延が許容が難しいアプリケーションを利用する際にセキュリティ対策が必要な場合には本提案システムが有効となる．

8. まとめ

近年，IoT の発展により利便性が高まる一方で，セキュリティ上のリスクも高まっている．また，今後はクラウド上のシステムと連携する形態ではなく，ホームネットワーク内で閉じたデバイス間通信が多くなることが想定される．そのため，アクセス制御等の更なるセキュリティ対策を行う必要がある．そこで本研究では，OpenFlow を用いたホームネットワーク監視と，コンテナを用いた IoT デバイスへのセキュリティ対策の適用フレームワークを構築した．そして，IoT デバイス間で閉じた通信を行うシミュレーション評価の比較を行い，ホームネットワークにおいてセキュリティ要件を保つことと性能も許容範囲であることを示した．

今後は，オーケストレータ等を用いて，新しい IoT デバイスがホームネットワークに追加された際に，自動的にコンテナが Proxy として配備される仕組みを検討する．また，Raspberry Pi 等の実機を用いた実験を行う予定である．

参考文献

[1] 総務省，”IoT・5G セキュリティ総合対策 2020”, サイバーセキュリティタスクフォース, pp.1-56, 2020.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, ”OpenFlow: enabling innovation in campus networks”, SIGCOMM Computer Communication Review, Vol.38, No.2, pp.69-74, 2008.

[3] P. A. Abdalla and C. Varol, ”Testing IoT Security: The Case Study of an IP Camera”, 2020 8th International Symposium on Digital Forensics and Security (ISDFS), pp.1-5, 2020.

[4] P. Ferrara, A. K. Mandal, A. Cortesi and F. Spoto, ”Static analysis for discovering IoT vulnerabilities”, International Journal on Software Tools for Technology Transfer, Vol.23, No.1, pp.71-88, 2021.

[5] A. Sivanathan, D. Sherratt, H. H. Gharakheili, V. Sivaraman and A. Vishwanath, ”Low-cost flow-based security solutions for smart-home IoT devices,” 2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp.1-6, 2016.

[6] P. Pawar and A. Trivedi, ”Device-to-Device Communication Based IoT System: Benefits and Challenges”, IETE Technical Review, Vol.36, No.4, pp.362-374, 2019.

[7] M. Serror, M. Henze, S. Hack, M. Schuba, and K. Wehrle, ”Towards In-Network Security for Smart Homes”, Proceedings of the 13th International Conference on Availability, Reliability and Security (ARES 2018), No.18, pp.1-8, 2018.

[8] Z. Zhang, T. Yu, X. Ma, Y. Guan, P. Moll and L. Zhang, ”Sovereign: Self-contained Smart Home with Data-centric Network and Security”, IEEE Internet of Things Journal, 2022.

[9] IPA 技術本部 ソフトウェア高信頼化センター (SEC), ”「つながる世界の開発指針」の実践に向けた手引き [IoT 高信頼化機能編]”, pp.1-96, 2017.