Vol.2022-IOT-58 No.2
2022/7/12

OpenFlow のフロー統計情報を用いた TCP scan 検知手法の提案

福原 悠真[1]　池部 実[2]　吉崎 弘一[3]　吉田 和幸[3]

概要：我々は，これまでインターネットと学内ネットワークの境界でポートミラーリングから攻撃を検知・遮断する不正通信検知システムを開発・運用してきた．さらに，不正通信検知システムで検知した攻撃者を，OpenFlow スイッチにて遮断する手法を提案した．しかし，従来手法では複数の OpenFlow スイッチから構成されるネットワークの場合にはポートミラーリングしたパケットを不正通信検知システムに送るためのネットワークが必要であり，実現には至っていない．そこで，本研究では OpenFlow スイッチの統計情報を用いて OpenFlow コントローラにて攻撃者からの scan を検知する手法を提案する．scan は通常の TCP 通信に比べ，送信元からの SYN パケットが多くなることに着目した検知手法を検討した．OpenFlow スイッチから各フローエントリの受信パケット数を取得し SYN パケットとそれ以外のパケットの比率により scan を検知・遮断する．提案手法を仮想ネットワーク上で実装し，実験・評価した．複数の条件で実験した結果，scan の送信元 IP アドレスを攻撃者として検知・遮断できることを確認した．

Proposal for TCP scan detection method using flow statistics of OpenFlow

1. はじめに

NICTER の 2021 年観測レポートによると，ネットワークやホストの存在を探索する不正通信が増加している [1]．インターネットを安全に利用するためには，不正通信に対するセキュリティ対策が必要である．マルウェアによる調査目的で行われる scan とは，攻撃対象の情報を得る探索行為である．そのため，scan を迅速に検出し，遮断することで被害を最小限に抑えることにつながる．

我々は，これまでは学外から学内ネットワークへ送信されるパケットをミラーリングし，攻撃者を検知して，通信を遮断する不正通信検知システムを開発・運用してきた [2]．下川らの手法 [3] では，不正通信検知システムで検知した攻撃者を，OpenFlow スイッチで検知した送信元 IP アドレスをフローエントリのマッチフィールドに設定し，遮断する．下川らによる従来手法の構成を図 1 に示す．従来手法では，学内ホストに向けた通信に対して，ポートミラーリングし，不正通信検知システムがスイッチに流入してきたパケットを監視する．攻撃者を検知すると，OpenFlow コントローラへ攻撃者の送信元 IP アドレスを通知し，攻撃者を遮断するフローエントリを OpenFlow スイッチに挿入する．不正通信検知システムとによる通知を受け，動的にフローエントリを設定することにより，攻撃に迅速に対応できる．しかし，複数の OpenFlow スイッチから構成される学内ネットワークの場合には，ポートミラーリングしたパケットを不正通信検知システムに送るために専用ネットワークが新たに必要であり，システムの実現には至っていない．そこで，本論文では，OpenFlow ネットワークから取得可能な情報を用いて，不正通信を検知できるシステムを構築することを目的とする．本研究では，その足がかりとして OpenFlow スイッチが保持する統計情報に着目し，OpenFlow コントローラでインターネットからの scan を検知する手法を検討した．

本論文では，2 章で関連研究について述べ，3 章で，OpenFlow のフロー統計情報を用いた TCP scan 検知手法の詳細について述べる．4 章で，提案手法の動作検証と検証結果について述べ，5 章では，まとめと今後の課題について述べる．

[1] 大分大学大学院 工学研究科
[2] 大分大学 理工学部
[3] 大分大学 学術情報拠点情報基盤センター

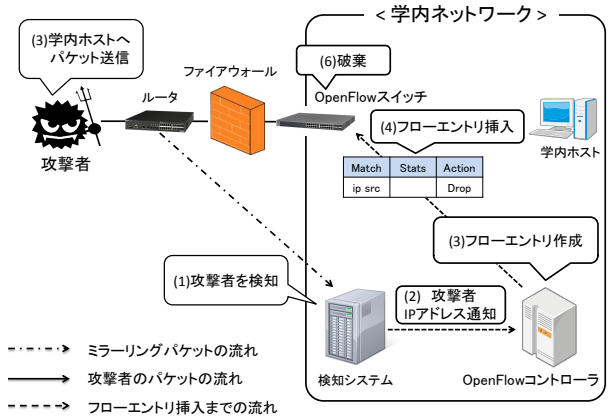

図 1 下川らの従来手法の構成図

2. 関連研究

OpenFlow を用いた scan 検知手法として Neu, C. V らの研究がある [4]．この研究では，各 OpenFlow スイッチの持つフローエントリの統計情報を数秒ごとに収集し，分析することで scan の発生を判定している．検知の条件として，scan のストリームが通常 3 パケット以内に収まることに着目している．TCP に関する各フローエントリについて，マッチしたパケット数が 5 以下の数をカウントし，頻繁に scan されるポート番号と scan されないポート番号に対して重みをつけてスコアを算出する．このスコアが閾値を超えた場合には scan が発生したと判定する．

小野らの scan 攻撃検知手法では，Packet_In メッセージに着目している [5]．Packet_In メッセージの異常増加を起点に，scan の検出処理に移行する手法を提案している．Packet_In メッセージの流量を単位時間当たりの Packet_In メッセージの数で表し，同一のホストから異なるポート番号に対する Packet_In メッセージを m 個受信するごとに $Flowrate$ を求める．

$$Flowrate : Rp = \frac{m}{\Delta Time} \tag{1}$$

求めた $Flowrate$ を基に k 近傍法を用いて異常増加を検出する．具体的な処理手順を以下に示す．

(1) 流量のデータ列に対してウインドウ W を設定する
(2) 新しいデータが得られた場合，新しいデータと W 内のすべてのデータとの差を求める
(3) 計算した差について，最も小さい k 個のデータを選び，その平均値を新しいデータの異常スコアとする
(4) 異常スコアが閾値を超えていた場合は異常増加があったと判定する

あるホストからの Packet_In メッセージの異常増加が検出された場合，Packet_In メッセージの異常増加を発生させているホストの MAC アドレスを元に，ホストが接続されているスイッチを特定する．その後に，コントローラからスイッチに対して統計情報をリクエストして，受信した統計情報を元に scan を検出する．scan の検出方法は，Neu, C. V らの手法に基づく．

関連研究 [4] では，数秒間隔で周期的に統計情報を取得してデータベースに保存し，ストリームに基づいて検出処理を実行している．また，統計情報の取得に用いるフローエントリのマッチフィールドには，送信元 IP アドレス，宛先 IP アドレス，送信元ポート番号，宛先ポート番号の 4 種類を設定している．そのため，scan が行われるとデータベース間における処理の増大やフローエントリの急増が問題点として考えられる．本手法では，OpenFlow の統計情報と TCP フラグに着目し，検知から遮断まですべて OpenFlow のみで行う．また，マッチフィールドに宛先 IP アドレスや宛先ポート番号を用いないことで，単一の攻撃者からの scan によるフローエントリの急増を防ぐ．

3. OpenFlow のフロー統計情報を用いた TCP scan 検知手法の提案

3.1 検知手法の概要

通常，ネットワーク中のシステムに対する攻撃や侵入の前に，攻撃対象の脆弱性を調査するために攻撃者による scan が行われる [6]．特にマルウェア感染活動の攻撃対象を選定するために scan は用いられる．scan を行っている不審なホストを迅速に検出して，ネットワークから遮断することは，さらなる攻撃やマルウェアの感染拡大の防止のために重要となる．

SYN scan では，短時間に多くの SYN パケット[*1]が送信されるため，SYN パケットの到達回数が増加することになる．TCP scan は TCP コネクションを確立することでサービスの稼働状況を調査する手法である．正常な通信においても TCP コネクションを確立するが，SYN パケットは TCP スリーウェイハンドシェイクでの TCP コネクション確立時のみに発生する．TCP コネクション確立後はデータ転送により，SYN パケット以外のパケットが多く到達する．この傾向に基づき，提案手法では，一定時間おきに OpenFlow コントローラで OpenFlow スイッチから統計情報として，フローエントリごとの受信パケット数を取得する．取得した受信パケット数をもとに，SYN パケットと SYN パケット以外のパケットの比率によって，送信元 IP アドレスをマッチフィールド，インストラクションを Drop に設定した優先度が高いフローエントリを挿入して検知する．提案手法の構成を図 2 に示す．本手法では，インターネットから学内ホストに対する scan を想定するため，インターネット側に攻撃ホスト，学内ネットワーク側に被攻撃ホストを設置する．提案システムでは，フローエントリのマッチフィールドにトランスポート層の情報

[*1] SYN パケットは TCP の制御フラグで SYN ビットのみが 1 に設定されたパケットを意味する．

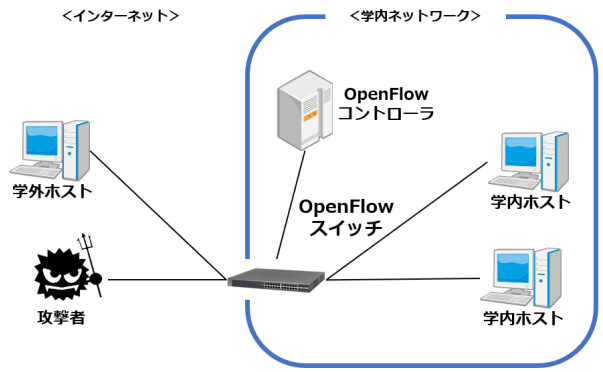

図 2 提案手法の構成図

である TCP フラグを指定するため，OpenFlow1.5 を使用し，学内の OpenFlow コントローラに接続した OpenFlow スイッチをインターネットと学内ネットワークの境界に設置した環境を前提とする．

3.2 検知のためのフローエントリ

提案手法では，SYN パケットの到達回数と，それ以外の TCP フラグパケットの到達回数を区別して取得する必要がある．そのため，1 つの送信元 IP アドレスに対して，2 種類のフローエントリを挿入する．2 種類のフローエントリにおけるパケット到達回数の比率を利用して，scan を検知し，遮断用のフローエントリを挿入する．提案手法で送信元 IP アドレス単位で設定するフローエントリを表 1 に示す．フローエントリ 1 では，マッチフィールドに送信元 IP アドレスと SYN フラグを設定し，SYN パケットが該当する．フローエントリ 2 では，マッチフィールドに送信元 IP アドレスのみを設定する．フローエントリ 1 の優先度をフローエントリ 2 よりも高く設定することで，フローエントリ 1 に SYN パケット，フローエントリ 2 にそれ以外のパケットが該当する．遮断用のフローエントリは，マッチフィールドに送信元 IP アドレスのみを設定し，インストラクションを Drop にした優先度の高いフローエントリである．このフローエントリにより，攻撃者からのパケットを破棄する．確認応答用のフローエントリはインターネットからのコネクション要求があった場合に学内ホストからの SYN/ACK パケットを送信するためのフローエントリである．また，Packet_In メッセージを発生させるために宛先を OpenFlow コントローラに設定したフローエントリをあらかじめ挿入している．

3.3 通信制御の流れ

提案手法による通信制御の流れを図 3 に示す．

(1) OpenFlow スイッチに SYN パケットが流入した場合，OpenFlow スイッチは送信元 IP アドレスに関するフローエントリの有無を確認する．
(2) 該当するフローエントリが無い場合，OpenFlow スイッチは OpenFlow コントローラに Packet_In メッセージで問い合わせる．
(3) OpenFlow コントローラは OpenFlow スイッチに表 1 のフローエントリ 1，2 を挿入する．
(4) OpenFlow スイッチはパケットが流入するごとに該当するフローエントリの受信パケット数を記録する．
(5) OpenFlow コントローラは OpenFlow スイッチから統計情報を一定時間毎に取得し，送信元 IP アドレス毎の SYN パケットと SYN パケット以外のパケットの受信パケット数の比率を計算する．
(6) 送信元 IP アドレス単位で，一定時間内に SYN パケットの到達回数がそれ以外のパケットの到達回数と同数以上である場合，遮断用のフローエントリを挿入し，攻撃者を検知する．

以上の手順により，攻撃者からの通信を制御する．

表 1 提案手法のフローエントリ

	Priority	Match fields	Instruction
フローエントリ 1	10	攻撃者候補の送信元 IP アドレス TCP フラグ = 0x002	Output (出力)
フローエントリ 2	1	攻撃者候補の送信元 IP アドレス	Output (出力)
遮断用フローエントリ	100	攻撃者の送信元 IP アドレス	Drop (破棄)
確認応答用フローエントリ	80	学内ホストの送信元 IP アドレス TCP フラグ = 0x012	Output (出力)
Packet_In 用のフローエントリ		ANY	Controller

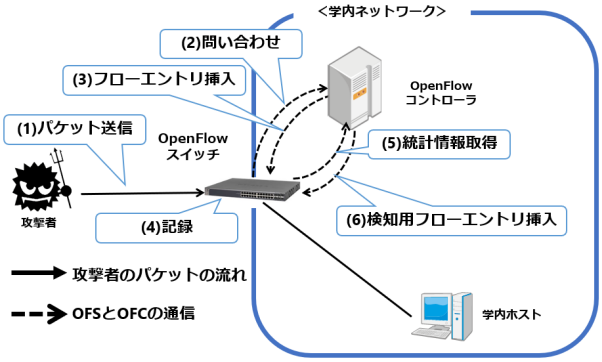

図 3 提案手法の通信制御の流れ

3.4 タイムアウトによる復旧

正常な通信を誤検知した場合の対応として，フローエントリに有効期限を設定する．遮断用のフローエントリは，使用されない状態が一定時間続くと削除する．これによ

り，誤検知が起きた場合でも一定時間通信がない場合，再度通信が可能となる．

また，検知条件と照合するための 2 種類のフローエントリについても，有効期限を設定する．フローエントリを挿入してから，一定時間経過後に削除するように設定することで，送信者数の増加によるフローエントリの飽和を防ぐ．

4. OpenFlow の統計情報を用いた TCP scan 検知手法の評価

本章では，OpenFlow における統計情報を用いた TCP scan 検知手法の評価実験について述べる．

4.1 実験概要

scan には，単一のホストに複数ポート宛パケットを送信する垂直 scan と，複数のホストに単一ポート宛パケットを送信する水平 scan がある．マルウェア感染のための scan を実行する例にマルウェア Mirai [7] の活動がある．Mirai は Telnet(23/TCP) をはじめとして特定ポートに scan を実行することが確認されている．

本研究ではマルウェア Mirai による感染活動を想定して実験した．まず，Telnet(23/TCP) が稼働していないホストに対する scan を想定して実験した．実験 1 では 1 台の Mirai からの単一ホストへの scan を想定し，実験 2 では，Mirai からの複数ホストに対する水平 scan を想定して実験した．実環境では，攻撃者以外の正常な通信も存在する．また，Telnet(23/TCP) が稼働しているホストに対しても scan を行う可能性がある．そこで，実験 3 では正常なホストが存在する状況で実験し，実験 4 では，実験 1 の構成に Telnet(23/TCP) が稼働しているホストを想定して実験した．

4.2 実験環境

実験環境は，Linux マシン上で，ネットワークエミュレータである Mininet [8] を用いて仮想ネットワークを構築した．OpenFlow スイッチ (OpenvSwitch)，OpenFlow コントローラを 1 台ずつ設置する．OpenFlow コントローラとして Ryu [9] を用いた．OpenFlow のバージョンは 1.5 である．攻撃はポートスキャンツール Nmap を利用する．本研究のすべての実験は統計情報の取得間隔を 10 秒に設定し，2 種類のフローエントリのタイムアウトを 200 秒，遮断用のフローエントリのタイムアウトを 100 秒とした．

4.3 実験 1：通信を検知するためのフローエントリの動作検証

4.3.1 実験 1 の概要

実験 1 では，OpenFlow のフロー統計情報を用いて SYN scan を検知するためのフローエントリの動作を検証した．実験 1 の環境構成を図 4 に示す．検証方法としては，攻撃ホストから被攻撃ホストに向けて，閉じている宛先ポート番号を 2 個指定した SYN scan の送信を 3 回繰り返す．

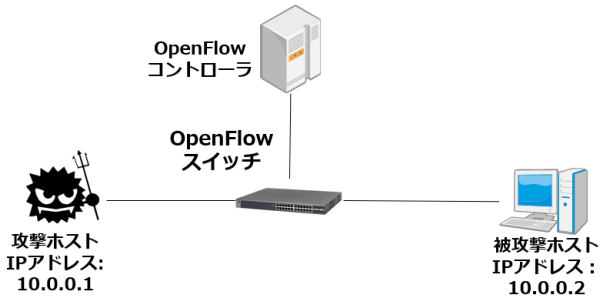

図 4 実験 1 の環境構成図

表 2 SYN scan 送信前のフローテーブル

Match fields	Priority	Received Packets	Instructions
ANY	0	0	CONTROLLER

表 3 実験 1 における SYN scan を 1 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1 tcp_flags=SYN	10	0	Output
src_IP=10.0.0.1	1	0	Output
ANY	0	12	CONTROLLER

この実験で検証する事項は以下の 3 点である．

(1) マッチフィールドを送信元 IP アドレスとしたフローエントリとマッチフィールドを送信元 IP アドレスと TCP SYN フラグに設定したフローエントリの 2 種類のフローエントリが挿入されているか
(2) 遮断用のフローエントリが挿入されているか
(3) 攻撃ホストが送信したパケットを遮断できているか

4.3.2 実験 1 の結果

実験前のフローテーブルは Packet_In 用のフローエントリのみ保持している (表 2)．攻撃ホストから被攻撃ホストに向けて，閉じている宛先ポート番号を 2 個指定し 1 回目の SYN scan 送信後のフローテーブルを表 3，2 回目を送信後のフローテーブルを表 4，3 回目を送信後のフローテーブルを表 5 に示す．表 3 より，1 回目の送信でマッチフィールドを送信元 IP アドレスとしたフローエントリとマッチフィールドを送信元 IP アドレスと TCP SYN フラグに設定したフローエントリの 2 種類のフローエントリが挿入されていることを確認した．また，表 4 より，2 回目の送信後には遮断用フローエントリが挿入されていることを確認した．さらに，表 5 の Received Packet(受信パケット数) より，3 回目の送信では遮断用のフローエントリに合致し，攻撃ホストからのパケットを破棄していることを確認した．この結果より，単一のホストにおける閉じたポートに対する SYN scan を検知できることを確認した．

表 4　実験 1 における SYN scan を 2 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1	100	0	Drop
src_IP=10.0.0.1 tcp_flags=SYN	10	2	Output
src_IP=10.0.0.1	1	0	Output
ANY	0	20	CONTROLLER

表 5　実験 1 における SYN scan を 3 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1	100	4	Drop
src_IP=10.0.0.1 tcp_flags=SYN	10	2	Output
src_IP=10.0.0.1	1	0	Output
ANY	0	23	CONTROLLER

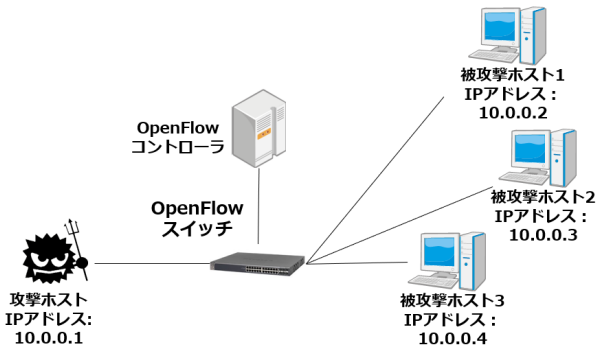

図 5　実験 2 の環境構成図

4.4　実験 2：単一の攻撃者から複数宛先への攻撃時の動作検証

4.4.1　実験 2 の概要

実験 2 では，単一の攻撃者からの水平 scan に対する動作を検証した．

実験 2 の環境構成を図 5 に示す．検証方法としては攻撃ホストから被攻撃ホスト 1 から被攻撃ホスト 3 の 3 台に向けて，それぞれ宛先ポート番号を 2 個指定した SYN scan を 2 回送信したときのフローテーブルを確認した．

4.4.2　実験 2 の結果

実験結果として，攻撃ホストから被攻撃ホスト 1 から被攻撃ホスト 3 の 3 台に向けて，閉じている宛先ポート番号を 2 個指定し 1 回目の SYN scan を送信後のフローテーブルを表 6，2 回目を送信後のフローテーブルを表 7 に示す．表 6 より，マッチフィールドを送信元 IP アドレスとしたフローエントリとマッチフィールドを送信元 IP アドレスと TCP SYN フラグに設定したフローエントリの 2 種類のフローエントリと，遮断用フローエントリが挿入されていることを確認できた．また，表 7 より，攻撃ホストを遮断できている．実験 2 では，実験 1 とは異なり，1 回の SYN scan の送信で検知用フローエントリが挿入された．これは

表 6　実験 2 における SYN scan を 1 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1	100	0	Drop
src_IP=10.0.0.1 tcp_flags=SYN	10	4	Output
src_IP=10.0.0.1	1	0	Output
ANY	0	38	CONTROLLER

表 7　実験 2 における SYN scan を 2 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1	100	12	Drop
src_IP=10.0.0.1 tcp_flags=SYN	10	4	Output
src_IP=10.0.0.1	1	0	Output
ANY	0	44	CONTROLLER

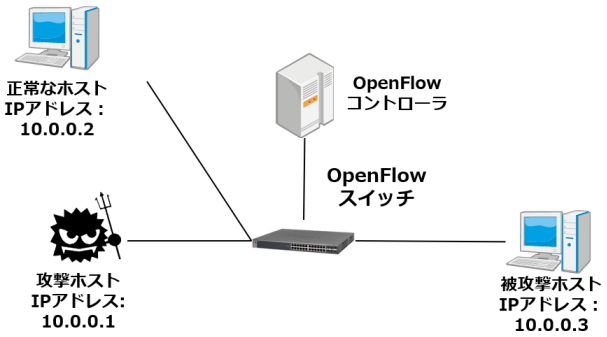

図 6　実験 3 の環境構成図

実験 2 では，SYN scan の送信を 3 つのホストに向けて送信しているためである．マッチフィールドを送信元 IP アドレスに設定していることから，ある 1 つのホストに向けた SYN パケットで 2 種類のフローエントリが挿入される．残りの 2 つのホストに向けた SYN パケットが先に挿入されたフローエントリに合致し，検知条件を満たすため，遮断用フローエントリが同時に挿入されたことになる．

4.5　実験 3：正常なホスト存在時の動作検証

4.5.1　実験 3 の概要

実環境では，正常なホストとの通信中に攻撃者から scan を送信される可能性がある．実験 3 の環境構成を図 6 に示す．実験 3 では，実験 1 の環境に加え，iperf を用いた正常な通信を追加して以下の手順で検証した．

(1) iperf にて，インターネット上の正常なホストから被攻撃ホストに対して，TCP パケットを 100 秒間送信
(2) Nmap にて，攻撃ホストから宛先ポートを 2 個指定した SYN scan の送信を 3 回繰り返す

以上の検証方法により，正常な通信を妨げることなく，遮断できているか検証した．

表 8 実験 3 における SYN scan を 1 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1 tcp_flags=SYN	10	0	Output
src_IP=10.0.0.1	1	0	Output
src_IP=10.0.0.2 tcp_flags=SYN	10	0	Output
src_IP=10.0.0.2	1	23817	Output
src_IP=10.0.0.2 tcp_flags= SYN/ACK	80	10	Output
ANY	0	21437	CONTROLLER

表 9 実験 3 における SYN scan を 2 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1	100	0	Drop
src_IP=10.0.0.1 tcp_flags=SYN	10	2	Output
src_IP=10.0.0.1	1	0	Output
src_IP=10.0.0.2 tcp_flags=SYN	10	0	Output
src_IP=10.0.0.2	1	48104	Output
src_IP=10.0.0.2 tcp_flags= SYN/ACK	80	10	Output
ANY	0	44838	CONTROLLER

4.5.2 実験 3 の結果

攻撃ホストから被攻撃ホストの閉じている宛先ポート番号を 2 個指定し 1 回目の SYN scan 送信後のフローテーブルを表 8，2 回目を送信後のフローテーブルを表 9，3 回目を送信後のフローテーブルを表 10 に示す．表 8 より，攻撃ホストと正常なホストのどちらのホストに対しても，マッチフィールドを送信元 IP アドレスとしたフローエントリとマッチフィールドを送信元 IP アドレスと TCP SYN フラグに設定したフローエントリの 2 種類のフローエントリが挿入されていることを確認した．また，表 9 より，2 回目の SYN scan 送信で，攻撃ホストに対して遮断用フローエントリが挿入されていることが確認できた．さらに，表 10 より，3 回目の送信時には攻撃者ホストからのパケットが遮断されていたことから，正常な通信を妨げることなく，検知できることを確認した．

4.6 実験 4：開放ポートに対する通信時の動作検証

4.6.1 実験 4 の概要

ここまでの実験では，サービスが稼働していないことを想定して閉じているポート番号での動作を検証した．しかし，実環境では，被攻撃ホストでサービスが稼働しているため，開いているポート番号に対しても scan が送信される．そこで，実験 4 では，TCP コネクションの確立が可能な開いたポート番号に対する動作を検証した．実験 4 の環境構成は実験 1 と同様である．

表 10 実験 3 における SYN scan を 3 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1	100	4	Drop
src_IP=10.0.0.1 tcp_flags=SYN	10	2	Output
src_IP=10.0.0.1	1	0	Output
src_IP=10.0.0.2 tcp_flags=SYN	10	0	Output
src_IP=10.0.0.2	1	125230	Output
src_IP=10.0.0.2 tcp_flags= SYN/ACK	80	10	Output
ANY	0	116715	CONTROLLER

表 11 実験 4 における TCP scan を 1 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1 tcp_flags=SYN	10	0	Output
src_IP=10.0.0.1	1	2	Output
src_IP=10.0.0.2 tcp_flags= SYN/ACK	80	0	Output
ANY	0	7	CONTROLLER

検証方法としては，TCP ポート番号 8000 を web でサービスを稼働させ，Nmap の TCP scan を用いて，攻撃ホストから被攻撃ホストに向けて，宛先ポート番号に 8000/TCP を指定した TCP scan を 3 回繰り返して送信したときの動作を検証した．

4.6.2 実験 4 の結果

実験 4 における攻撃ホストから被攻撃ホストに宛先ポート番号を 8000/TCP に指定した TCP scan の 1 回目を送信後のフローテーブルを表 11，2 回目を送信後のフローテーブルを表 12，3 回目を送信後のフローテーブルを表 13 に示す．表 11 より，1 回目の送信後にマッチフィールドを送信元 IP アドレスとしたフローエントリとマッチフィールドを送信元 IP アドレスと TCP SYN フラグに設定したフローエントリの 2 種類のフローエントリが挿入されていることを確認した．しかし，表 12 より，検知条件に合致せず，遮断用のフローエントリが挿入されないことを確認した．この結果より，開いたポートにのみ TCP scan を送信した場合に，現在の検知条件では検知できない問題が発見された．これは，TCP scan に用いられる攻撃ホストが被攻撃者からの SYN/ACK パケットを受信した後に送信される TCP コネクション切断時の RST パケットが原因であると考えられる．

表 12 実験 4 における TCP scan を 2 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1 tcp_flags=SYN	10	1	Output
src_IP=10.0.0.1	1	4	Output
src_IP=10.0.0.2 tcp_flags= SYN/ACK	80	1	Output
ANY	0	12	CONTROLLER

表 13 実験 4 における TCP scan を 3 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1 tcp_flags=SYN	10	2	Output
src_IP=10.0.0.1	1	6	Output
src_IP=10.0.0.2 tcp_flags= SYN/ACK	80	2	Output
ANY	0	14	CONTROLLER

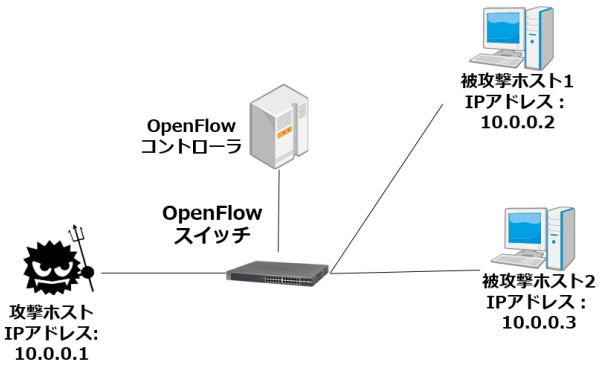

図 7 実験 5 の環境構成図

表 14 実験 5 における TCP scan を 1 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1 tcp_flags=SYN	10	1	Output
src_IP=10.0.0.1	1	2	Output
src_IP=10.0.0.2 tcp_flags= SYN/ACK	80	0	Output
ANY	0	11	CONTROLLER

表 15 実験 5 における TCP scan を 2 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1	100	0	Drop
src_IP=10.0.0.1 tcp_flags=SYN	10	4	Output
src_IP=10.0.0.1	1	4	Output
src_IP=10.0.0.2 tcp_flags= SYN/ACK	80	1	Output
ANY	0	25	CONTROLLER

表 16 実験 5 における TCP scan を 3 回送信後のフローテーブル

Match fields	Priority	Received Packets	Instructions
src_IP=10.0.0.1	100	4	Drop
src_IP=10.0.0.1 tcp_flags=SYN	10	4	Output
src_IP=10.0.0.1	1	4	Output
src_IP=10.0.0.2 tcp_flags= SYN/ACK	80	1	Output
ANY	0	26	CONTROLLER

4.7 実験 5：開放ポートを含めた通信時の動作検証

4.7.1 実験 5 の概要

実験 5 では，TCP コネクションの確立が可能な開いたポートを含め動作を検証した．実験 5 の環境構成を図 7 に示す．

検証方法としては，TCP ポートの 8000 を web でサービスを稼働させ，Nmap の TCP scan を用いて，攻撃ホストから被攻撃ホストに向けて，宛先ポート番号に 8000/TCP と閉じたポート 1 個を指定した TCP scan を 3 回繰り返して送信したときの動作を検証した．

4.7.2 実験 5 の結果

実験 5 における攻撃ホストから被攻撃ホストに宛先ポート番号を 8000/TCP に指定した TCP scan の 1 回目を送信後のフローテーブルを表 14，2 回目を送信後のフローテーブルを表 15，3 回目を送信後のフローテーブルを表 16 に示す．表 14 より，1 回目の送信後にマッチフィールドを送信元 IP アドレスとしたフローエントリとマッチフィールドを送信元 IP アドレスと TCP SYN フラグに設定したフローエントリの 2 種類のフローエントリが挿入されていることを確認した．また，表 15 より，2 回目の送信後に遮断用フローエントリが挿入されていることを確認した．さらに表 16 より，3 回目を送信すると，すべてのパケットが遮断されたため，宛先ポートの半数以上が閉じたポートである場合，攻撃者を検知できることになる．

4.8 実験結果の考察

実験 1 では，単一の攻撃者とみなした攻撃ホストから SYN scan を 3 回送信し，1 回目の送信後に 2 種類のフローエントリが挿入され，2 回目の送信後に遮断用フローエントリが挿入された．3 回目の送信後には検知できていることを確認した．実験 2 では，被攻撃ホストを複数台設置し，水平 scan を想定した環境で動作を検証した．送信元 IP アドレスをマッチフィールドに設定していることから，単一のホストからの流入量について着目できるため，他の

実験より早い段階で検知できた．実験 3 では，iperf を用いて正常な通信を妨げずに，攻撃者を検知できることを確認した．実験 4 では，開いたポートを利用して，スリーウェイハンドシェイクが起きる場合の動作について検証したが開いたポートのみに対する TCPscan は検知できない問題点が発見された．そこで，実験 5 では，開いたポートと閉じたポートを含んだ場合の動作について検証した．閉じたポートが半数以上ある場合に検知できることを確認した．これらの実験結果より，サービスの稼働状態にかかわらずに行われるマルウェア感染攻撃の対象選定時に発生する水平 scan に対しては，1 台のホストのポートの開閉状況にかかわらず，提案手法にて検知可能である．また，本実験では，学内ホストからインターネットへの SYN/ACK パケットに対して新たなフローエントリを挿入していたが，実際には，マッチフィールドに ANY を設定したフローエントリを用いればよいため，今後は必要性のないものとなる．

5. おわりに

本研究では，OpenFlow の統計情報を用いた TCP scan 検知手法の提案として，マッチフィールドを送信元 IP アドレスに設定したフローエントリと，マッチフィールドを送信元 IP アドレスに SYN フラグを加えたフローエントリの，2 種類のフローエントリにより，SYN パケットとそれ以外のパケットを区別し，それらの持つ統計情報を周期的に取得することで，TCP scan を検知する手法を提案した．本システムをいくつかの実験環境で動作させ，攻撃の検知ができるかどうか評価実験を実施した．

今後の課題としては，検知できない例が存在するため，検知条件の改善が必要である．また，Mininet のネットワークエミュレータを用いた仮想環境上で実験を行ったため，本研究で作成したシステムを実環境上で動作させ，システムが正常に動作するか検証する必要がある．さらに，マッチフィールドを送信元 IP アドレス単位で挿入するため，DDos 攻撃など複数のホストから大量のパケットが到達した場合にフローエントリの飽和が想定されるため，今後対策が必要となる．

参考文献

[1] 国立研究開発法人情報通信研究機構サイバーセキュリティ研究所 サイバーセキュリティ研究室，NICTER 観測レポート 2021，https://www.nict.go.jp/press/2022/02/10-1.html
[2] 小刀祢知哉，天本大地，小埜勇貴，有馬竜昭，池部実，吉田和幸，scan 攻撃検知システムを用いた被検知ホストの挙動についての調査，第 65 回電気関係学会九州支部連合大会 pp. 278-278(2012)
[3] 下川大貴，小刀祢知哉，池部実，吉田和幸，OpenFlow を用いた攻撃者遮断システムの提案と評価，情報処理学会マルチメディア、分散、協調とモバイル (DICOMO2014) シンポジウム論文集，pp. 197-204，2014 年 7 月
[4] Neu, C. V, Tatsch, C. G, Lunardi, R. C, Miche-lin, R. A, Orozco, A. M. S，Zorzo, A. F，Lightweight IPS for port scan in OpenFlow SDN networks，2018 IEEE/IFIP Network Operations and Management Symposium，pp.16，2018 年 7 月
[5] 小野大地，和泉諭，阿部亨，菅沼拓夫，OpenFlow ネットワークにおける Packe-In メッセージの特性に基づくポートスキャン検出手法の設計と実装，情報処理学会インターネットと運用技術シンポジウム 2020，pp.63-70，2020 年 12 月
[6] JPCERT/CC，攻撃を目的としたスキャンに備えて 2019 年 7 月，https://www.jpcert.or.jp/newsflash/2019072201.html
[7] Internet Infrastructure Review（IIR）Vol.33，2016 年 12 月 15 日発行，https://www.iij.ad.jp/dev/report/iir/pdf/iir_vol33.pdf
[8] Mininet An Instant Virtual Network on your Laptop (or other PC)，http://mininet.org/
[9] RYU SDN Framework，https://ryu-sdn.org/