SDNに基づくMTDとIPSの併用による ネットワークスキャン防御手法の性能評価

千葉 翔也[1]　ギリエ ルイス[2]　和泉 諭[3]　阿部 亨[1,4]　菅沼 拓夫[1,4]

概要：
標的型攻撃では，攻撃者が Intrusion Prevention System（IPS）や Intrusion Detection System（IDS），Firewall（FW）などの各種防御技術を回避してネットワークに侵入し，ネットワークをスキャンする．これら防御技術は送信元 IP アドレスや宛先 IP アドレス，スキャンレートなどに基づいてスキャンを検知・遮断し，同一ネットワークへの攻撃拡大を阻止できるが，設定したレート未満である低レートスキャンの検知は難しい．持続的標的型攻撃では，長時間に渡ってネットワークに潜伏することもあり，攻撃者は低レートのスキャンでも攻撃に必要な情報を収集できる．そこで次々にネットワークアドレスを変更し，攻撃者が得られる情報に有効期限を設ける Moving Target Defense（MTD）戦略の利用が検討されている．MTD により，低レートスキャンの防御が実現できるが，アドレス変更間隔の短縮による通信の継続や IPS の通信監視などへの影響が懸念される．本研究では，Software Defined Network (SDN) を利用し，IPS が時間変化する IP アドレスの影響を受けないようにスイッチ上でパケットヘッダを書き換えることで，IPS の動作に悪影響を及ぼさない透過的な MTD を実現する．そして，MTD と IPS を併用するスキャン防御手法を提案する．本手法により，IPS は MTD の影響を受けずにトラフィックを監視できる．さらに，IPS が高レートのスキャンを防御し，MTD が低レートスキャンを妨害することで，様々なレートで行われるスキャンの防御が可能となる．本研究では，コントローラの負荷や，通信遅延やスループットに着目したシミュレーション評価を行った．本実験により，高レートスキャンと低レートスキャンを防御する本手法が，既存手法と同程度の CPU 負荷，通信品質で MTD を実現できることを確認した．

キーワード：Moving Target Defense，ネットワークスキャン，Software Defined Network

A Performance Evaluation of Network Scan Defense Method by Combining an SDN-based MTD and IPS

1. はじめに

企業に対する標的型攻撃はメールなどを介してインターネットからローカルネットワークにマルウェアを送り，情報漏えいや金銭的な被害を引き起こす手段が多く用いられ，それによる被害も起きている [1]．標的型攻撃では，侵害されたホストを通じて，IP アドレススキャンやポートスキャンなどのローカルネットワークへの偵察が行われる．スキャンで得た情報は，他のクライアントやサーバに感染を広げる水平展開に利用される．このため，侵入を前提としてスキャン対策を適用し，攻撃者に情報を与えないことが感染拡大の阻止に繋がる．これらの対策として，Intrusion Prevention System（IPS）や Intrusion Detection system（IDS），Firewall（FW）などの防御技術が用いられている．これらに，検知対象とするスキャンのレートを設定し，スキャンの検知・遮断を行う．しかし，攻撃者が IPS や IDS を回避するために行う，設定値よりも低いレートのスキャン（低レートスキャン）の検知は難しい．

そこで，低レートスキャンの対策として Moving Target

[1] 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University
[2] 東北大学電気通信研究所
Research Institute of Electrical Communication, Tohoku University
[3] 仙台高等専門学校総合工学科
National Institute of Technology, Sendai College
[4] 東北大学サイバーサイエンスセンター
Cyberscience Center, Tohoku University

Defense (MTD) [2] が注目されている．MTD では，攻撃者の標的を定期的に移動することで偵察や攻撃から保護する．ネットワークの MTD では，IP アドレスや MAC アドレス，ポート番号などを頻繁に変更することで，ネットワークの情報をスキャンしても容易に悪用できないようにする．例えば，ネットワーク内のホストに設定された IP アドレスを 60 秒間隔で変更する場合，攻撃者がネットワークをスキャンして入手できる IP アドレスのリストの有効期間は 60 秒に限定される．60 秒経過後，各ホストにはそれぞれ異なる IP アドレスが割り当てられる．したがって攻撃者は次のアドレス変更が起こるまでに偵察と攻撃を終える必要が生じ，攻撃者の行動は制限される．

MTD の実装にはパケットヘッダの書き換えや経路の変更が柔軟にできる Software Defined Network (SDN) が広く用いられている．SDN による MTD の実装においては，パケットヘッダの書き換えに基づくアドレス変更を伴う．このアドレス変更は，IPS や IDS などの既存の防御技術に悪影響を及ぼすことがある．例えば，ホストの IP アドレスに基づいてホスト間の通信の監視や遮断をする場合，アドレスが定期的に変更されるとその継続的な監視，遮断が困難となる．この影響を避けるために，MTD のみで高レート・低レートの両方のスキャンを防御する場合，MTD のアドレス変更間隔を極端に短く設定することが考えられるが，MTD を実行しているハードウェアへの高負荷や，通信遅延，スループットへの影響が懸念される．

そこで本研究では，IPS に透過的な MTD を実装し，IPS と併用することで，ローカルネットワーク内のホストを高レート・低レートの両方のスキャンから防御する手法を提案する．提案手法では，SDN の実装のひとつである OpenFlow を用い，OpenFlow スイッチ間に配置された IPS が時間変化する IP アドレスの影響を受けないようにパケットヘッダの IP アドレスやフローテーブルを書き換える．具体的には，スイッチ間に流れるパケットヘッダの宛先，送信元アドレスは時間変化しない各ホストに設定された実際のアドレスに基づき表され，スイッチとエンドホストの間を流れるパケットヘッダの宛先，送信元アドレスは時間変化するアドレスで表される．それにより，IPS への透過性を維持しながら MTD の適用を可能とする．本手法は，MTD と IPS を相補的に利用することで，IPS の攻撃の監視や遮断といった機能に影響を与えることなく，高レートスキャンと低レートスキャンの双方を防御する．

さらに，エミュレーション環境で複数のホストとスイッチからなるテストベッドを構築し，コントローラの負荷や通信遅延，スループットに着目して提案手法の性能評価を行った．実験から，IPS に透過的な MTD を利用して高レートスキャンと低レートスキャンの防御を可能とする本手法が，既存手法 [3] と同程度の CPU 負荷，通信品質で MTD を実現できることを確認した．

2. 関連研究

スキャン対策として，IPS や IDS，FW などの防御技術が一般的に広く利用されている．IPS はトラフィックを検査し不正な通信を遮断し，IDS は不正な通信の検知のみ行う．FW はルールに基づいてトラフィックを遮断する．しきい値に基づいたルールをこれらに設定することで，スキャン検知と遮断に利用する．例えば，あるホストが特定のホストに対して短時間で複数のポートに接続を試みた場合や，短時間で複数のホストに対して ICMP パケットの送信を試みた場合など，それらを n [hosts/min] のようなレートで表したルールを設定する．

しかし，ネットワークスキャナツールには，スキャンのインターバルを設定できるものがあり，スキャンレートに基づく検知の回避に利用できる [4]．例えば，ネットワークに侵入した攻撃者は，n [hosts/min] のレートでスキャンを検知するように設定した IPS があるネットワークに対し，IPS の設定値未満の低レートでスキャンを行う．これにより，攻撃者は IPS を回避してネットワークを偵察できる．また，持続的標的型攻撃のようにネットワーク内に長期間に渡って攻撃者が潜伏している場合，低レートのスキャンでも情報を収集できる．そのため，IPS をスキャン検知に利用する場合は攻撃者がどのようなレートでスキャンするかを考慮し，スキャン回数を何分ごとに集計するかを決定するタイムスロットや，タイムスロットあたりにどの程度のスキャン試行回数があるかを決定するスキャンレートを設定する必要がある．従って，IPS のみであらゆるレートのスキャンを防ぐことは難しい．

そのような課題に有効なアイディアとして，対象となるアドレスをランダム化する MTD が注目されている．MTD の例として，ネットワーク上に配置されたゲートウェイを用いるネットワーク型が提案されている [5]．この手法は，ゲートウェイが各ホストに設定された実際の IP アドレスである *rIP* と時間変化する仮想の IP アドレスである *vIP* の変更を行うことで，エンドホストに対して透過的な MTD を実現している．他のネットワーク型の MTD として，IP アドレスの変更に加えてポート番号の変更も行う手法も提案されている [6]．これらは，アドレス変更の処理に複数の MTD ゲートウェイの配置が必要であり，管理や運用の容易さに課題がある．そのため，集中型のコントローラとスイッチで通信を柔軟に制御できる SDN の一実装である OpenFlow[7] を用いた MTD も研究されている．

OpenFlow スイッチは，OpenFlow コントローラによって自身にインストールされたフロールールに基づいて，パケットヘッダの書き換えや転送を行う．このフロールールに合致しないパケットをスイッチが受信した場合，スイッチはコントローラに対して Packet-In メッセージを送信し，OpenFlow コントローラはパケットに応じて適切なフロー

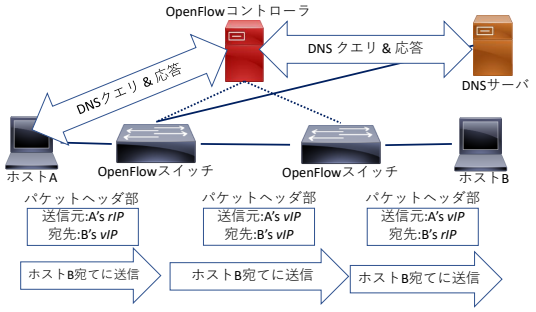

図 1 OF-RHM[3] に基づく MTD

ルールを OpenFlow スイッチにインストールする．この仕組を利用することで，ネットワークの MTD を実装できる．

OpenFlow を利用して，OpenFlow スイッチ上でアドレス書き換えを行う MTD の例として，文献 [3] がある．この手法の動作を図 1 に示す．この手法では *rIP* での通信が許可されていないホスト同士は *vIP* を用いて通信するが，この時間変化する *vIP* へのアクセスを各ホストへの追加ソフトウェアの導入なく提供するため，DNS が利用される．ホスト A がホスト B へパケットを転送するとき，OpenFlow コントローラが名前解決のプロセスに割り込み，DNS サーバが送信したレコードに含まれる *rIP* を *vIP* に書き換え，DNS サーバの代わりにホスト A に送信する．これにより，ホスト A はホスト B の *vIP* を取得し，エンドホスト B の *vIP* 宛にパケットを送信する．OpenFlow スイッチがそのパケットを受け取り，Packet-In メッセージを OpenFlow コントローラに送信すると，送信元アドレスを *vIP* に書き換えて他のスイッチやホストに転送するルールをスイッチにインストールする．以後その OpenFlow スイッチはインストールされたルールに従いパケットを転送する．

また，*rIP* をもつ各ホストに複数の *vIP* を同時に割り当てる MTD も提案されている [8]．SDN を利用してアドレス変更を行う点や名前解決にコントローラが割り込む点は文献 [3] と同様であるが，サービスごとに *vIP* を割り当てるため，複数のサービスが稼働しているサーバには複数の *vIP* が割り当てられ，攻撃者のスキャンがより困難となる．

これらの MTD は，時間をかけて行う低レートのスキャンに対して有効であるが，高レートのスキャンによりアドレス変更が適用されるまでの間にスキャンを完了し，攻撃を実行することは可能となってしまう．このような高レートのスキャンを防ぐためには，アドレス変更を高頻度で行うことが考えられるが，通信品質やコントローラのパフォーマンス等に影響が出ることから，MTD のみで低レートから高レートの様々なスキャンを防ぐことは難しい．

また，DNS を介して *vIP* を各ホストに通知する仕組みは，初期侵入によって侵害されたホストにも同様に提供される．そのため，攻撃者が初期に侵害したホストから他のホストのドメイン名を取得できる場合，それらのホストに関しては *vIP* を入手し，攻撃できる．このような攻撃を防ぐため，MTD 環境においても通信の監視は必要となるが，これらの研究では言及されていない．

MTD はアドレス変更間隔に基づいてアドレス変更を行うため，攻撃者の行動に制限をかけ，高レートでのスキャンやドメイン名の総当りなど，特徴的な行動を強要できるが，MTD のみで攻撃そのものを直接遮断することはできない．IPS や IDS などは，予め設定したしきい値に基づいてスキャンを検知するため，しきい値以下の低レートなスキャンを防ぐことが難しい．そこで，防御できるスキャンや攻撃の戦略が異なる MTD と IPS を併用することでより高精度にスキャン・攻撃対策が可能となると考えられる．

しかし，IPS のような防御技術は IP アドレスを変更する MTD の影響を受ける場合がある [9]．例えば，ネットワークに適用されている MTD[3], [5], [8] はパケットヘッダの書き換えを伴うが，スキャン対策としてそれらの MTD と IPS を併用する際， IPS はレートに基づいてスキャンを検知し，パケットヘッダに記載の IP アドレスを監視し，その送信元を遮断する．しかし，一定時間後に IP アドレスが新しいものに変更されると，その遮断は解除されそのホストは再び通信が可能となるため，IPS の継続的な通信監視・遮断に影響が出る．

また，IDS と MTD を用いたハイブリッド型のアプローチも提案されている [10]．MTD そのものには攻撃を直接検知・遮断する機能はないため，この手法では，IDS を利用して疑わしいホストを識別し，そのホストが関与する通信に適用する MTD のレベルを決定することで効率的な MTD を実現している．しかし，IDS はスキャン検知・遮断のために利用されておらず，MTD が既存の防御技術に与える影響については言及されていない．

3. IPS に透過的な MTD によるスキャン防御手法の提案

3.1 提案概要

2 章で説明したように，MTD のアドレス変更間隔と IPS のスキャン検知のしきい値の決定には，攻撃者のスキャン戦略の考慮が必要となる．また，MTD はアドレス変更の仕組みが IPS の監視や遮断に悪影響を及ぼすおそれがある．そこで，本研究では，MTD を IPS の動作に影響を与えないよう併用することで，それぞれ単体では防御が難しいスキャンを同時に防御する手法を提案する．なお，本提案は感染拡大を狙う攻撃者がローカルネットワーク内のホストに対して行うスキャンの防御を目的とするため，同一サブネット内のレイヤ 2 ネットワークへの適用を想定する．

提案手法では，IPS と MTD を併用することで，IPS が高レートスキャンを監視・遮断し MTD が低レートスキャンを防御する．その際，MTD のアドレス変更間隔を T としてアドレス変更を T 秒毎に実施し，IPS に設定するスキャ

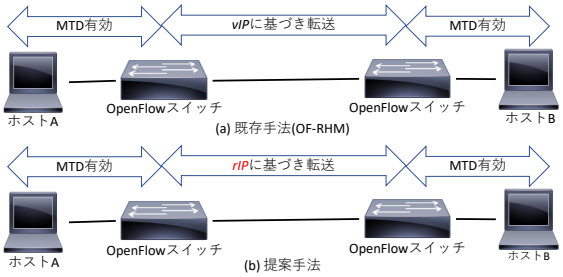

図 2 アドレス書き換えにおける提案手法と既存手法の比較

ンレートを n $[hosts/T]$ とすることで，IPS は T 秒以内に完了する比較的高レートで行われるスキャンを検知・遮断する．また，本研究で提案する MTD 手法は，既存手法 [3] と異なり IPS に透過的になるように OpenFlow スイッチ上でパケットヘッダのアドレスを書き換える．これにより，IPS がスキャンを検出した場合，MTD の影響を受けることなくスキャン元のホストを継続的に隔離できる．

提案手法と既存手法の MTD の動作比較を図 2 に示す．いずれも基本的にはローカルネットワーク上にある OpenFlow スイッチが，OpenFlow コントローラがインストールしたフロールールに基づいてアドレス書き換えと転送を行う．この OpenFlow スイッチによるアドレス書き換えは図 2(a) に示す既存手法 [3] と同様だが，図 2(b) に示す提案手法は OpenFlow スイッチ間を流れるパケットの宛先や送信元が *vIP* ではなく *rIP* に基づく点で異なる．この機構により IPS は，MTD の時間変化する *vIP* ではなく時間変化しない *rIP* に基づいてトラフィックを監視できる．各ホストは既存手法同様，原則 *vIP* に基づいて他のホストに接続するため，各ホストへの MTD の有効性を損なうことなく IPS との併用が可能となる．

3.2 パケットヘッダの書き換え

提案手法では，OpenFlow コントローラと SDN スイッチが連携することで，ARP パケットと IP パケットのヘッダ書き換えを行う．この ARP パケットと IP パケットを書き換える点は既存手法と同様であるが，提案手法では，宛先 IP アドレスと送信元 IP アドレスの書き換えを担うスイッチを既存手法から変更することで，IPS への透過性を確保する．各ホストは，宛先ホストの *vIP* に対応する MAC アドレスを取得するため ARP を利用してアドレス解決を行う．その時，OpenFlow コントローラは，Packet-In メッセージ受信時に ARP パケットを検査し，各 OpenFlow スイッチに IPS への透過性を維持するための処理を行うフロールールをインストールする．具体的には，同一の OpenFlow スイッチに送信元ホストと宛先ホストが接続されている場合，ARP リクエストの宛先アドレスの *vIP* を *rIP* へ，送信元アドレスの *rIP* を *vIP* へ書き換える．また，複数のスイッチを介して送信元ホストと宛先ホストが接続されている場合，OpenFlow スイッチが送信元アドレスの *vIP* を対応する *rIP* に書き換え，他の OpenFlow スイッチへ転送する．宛先ホストが接続されている OpenFlow スイッチがその ARP リクエストを受信したら，その送信元アドレスを *vIP* に書き換えて宛先ホストへ転送する．

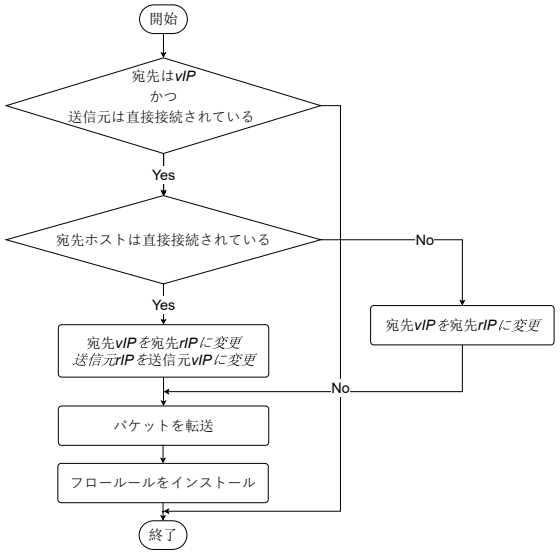

図 3 同一 OpenFlow スイッチ内のエンドホストへの転送及び他の OpenFlow スイッチへの転送フロー

宛先ホストが ARP リプライで応答した際の転送においても，ARP リクエストと同様の処理を行い，各エンドホストに対しては送信元アドレスが *vIP* となるように，OpenFlow スイッチ間の転送では送信元，宛先アドレスがどちらも *rIP* となるように ARP リプライを書き換える．これらのフロールールが OpenFlow コントローラによって各 OpenFlow スイッチへと適応的にインストールされることにより，各ホストは宛先ホストの *rIP* を知ることなく *vIP* と対応した MAC アドレスを入手し通信できる．

アドレス解決後，OpenFlow コントローラは図 3，図 4 に示す処理を行い，IP パケットについても，IPS に対する透過性を維持するフロールールをインストールする．

まず，図 3 に示す OpenFlow コントローラの処理について説明する．この処理は，OpenFlow スイッチがホストから受信したパケットを，同じスイッチに接続されたホスト及び他のスイッチに転送する際に実行される．まず，OpenFlow コントローラは，OpenFlow スイッチから Packet-In メッセージを受け取ると，スイッチが受信したパケットの宛先 IP アドレスが *vIP* であり，その送信元ホストがスイッチに直接接続されているか確認する．宛先ホストが送信元ホストと同じ OpenFlow スイッチに接続されている場合は，宛先 *vIP* を *rIP* に，送信元 *rIP* を *vIP* に書き換えて転送するフロールールをインストールする．それ以外の場合は宛先 *vIP* を *rIP* に書き換え，他の OpenFlow

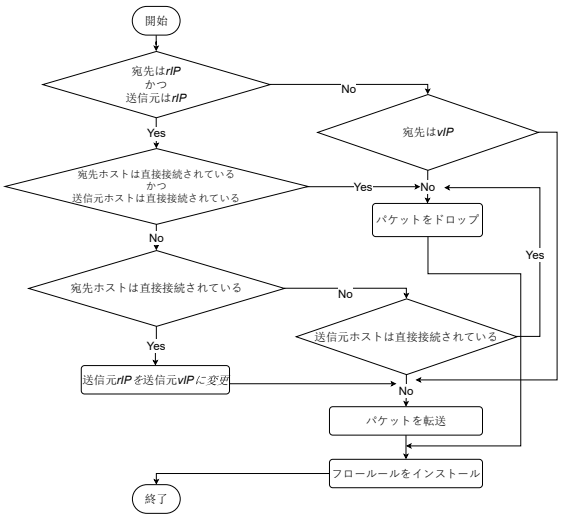

図 4 他の OpenFlow スイッチへの中継及びエンドホストへの転送フロー

スイッチに転送するフロールールをインストールする．

次に，OpenFlow スイッチが受信したパケットを他の OpenFlow スイッチ及び宛先ホストに送信する際にコントローラが実行する処理を図 4 に示す．OpenFlow スイッチがパケットを受信したとき，OpenFlow コントローラはそのパケットの宛先アドレスと送信元アドレスがどちらも *rIP* であることを確認する．その時，宛先ホストが送信元ホストと同じスイッチに接続されていた場合は，そのパケットをドロップする．これは MTD が *rIP* での通信を原則として許可しないためである．宛先ホストのみがその OpenFlow スイッチに直接接続されていた場合，送信元 *rIP* を対応する *vIP* に書き換えて，その宛先ホストへパケットを転送し，そのルールを OpenFlow スイッチへインストールする．宛先ホストが OpenFlow スイッチに接続されていなかった場合，そのホストが他の OpenFlow スイッチに接続されていることを意味するため，アドレスの書き換えは行わずに他の OpenFlow スイッチに転送し，そのフロールールをインストールする．

このような処理を OpenFlow コントローラが行い，各 OpenFlow スイッチにフロールールをインストールすることで，各ホストに対して MTD を適用しながら， IPS への透過性を維持する．

3.3 実装

提案する MTD について，本研究では Ryu SDN Framework[11] を利用し，OpenFlow コントローラの機能の一部として以下の機能を実装した．

(1) IPS に透過的なアドレス変更機能
(2) DNS を利用した *vIP* の通知機能
(3) 特定のホストを MTD の適用対象から除外する機能

(1) は，OpenFlow スイッチによる Packet-In メッセージ発行時に OpenFlow コントローラ上で実行される Packet-In ハンドラの一部として実装した．このハンドラは，Packet-In メッセージを受信するたび図 3，図 4 のフロールールのインストール処理を実行する．フロールールのインストール後，各 OpenFlow スイッチは次回のアドレス変更までそのルールに基づきパケットヘッダの書き換えと転送を行う．

(2) は，アドレス変更機能と同様に Packet-In ハンドラの一部として実装した．実装を簡易化するため，DNS サーバとの名前解決のパケットに関してはフロールールをインストールせず，毎回 Packet-In メッセージを発生させ，その都度 OpenFlow コントローラ上で DNS のパケットを書き換えるよう実装した．また，DNS サーバには各ホストのドメイン名とそのホストに設定されている実際の IP アドレス（*rIP*）の対応が登録されている．

ネットワーク内のホストが名前解決を試み，OpenFlow スイッチが DNS パケットを受信した場合，OpenFlow コントローラは Packet-In メッセージに含まれる DNS パケットのペイロード部を書き換える．そのパケットが DNS クエリであれば，そのパケットはそのまま DNS サーバに転送される．そのパケットが DNS レスポンスであれば，そのレスポンスに含まれる *rIP* を対応する *vIP* に書き換えたうえでそのホストに送信する．この仕組によりネットワーク内のホストは時間変化する *vIP* の代わりにドメイン名を使ってホストに接続できる．

(3) は，Packet-In ハンドラ実行時に通信しているホストが除外対象となっているか確認することで特定のホストにのみ *rIP* での通信を許可する．除外対象は *rIP* で指定し，OpenFlow コントローラは Packet-In ハンドラ実行時にパケットの宛先や送信元が，除外対象のリストに含まれているかどうか確認する．そのホストが除外対象に含まれている場合はアドレス変更を行わず，そのまま通常の L2 スイッチと同様に転送する．この仕組みは，各ホストに静的に設定されているデフォルトゲートウェイや，DNS サーバとの通信を阻害しないことを目的としている．また，ネットワーク内のいくつかのホストは管理者のもとで *rIP* で運用されることも考慮している．

また，MTD の *vIP* と *rIP* のマッピングは，OpenFlow コントローラが保持し，アドレス変更間隔 T 毎に更新される．各機能はこのマッピングテーブルを参照し *vIP* と *rIP* の対応関係を確認する．

4. 実験と評価

4.1 実験概要

本研究では，提案手法を実装した OpenFlow コントローラの CPU 負荷，通信遅延とスループットを測定し，提案手法の性能を評価した．まず提案手法と既存手法 [3] のそれぞれについて，OpenFlow コントローラが受信した Packet-In メッセージ数とコントローラのプロセスの CPU 使用率を

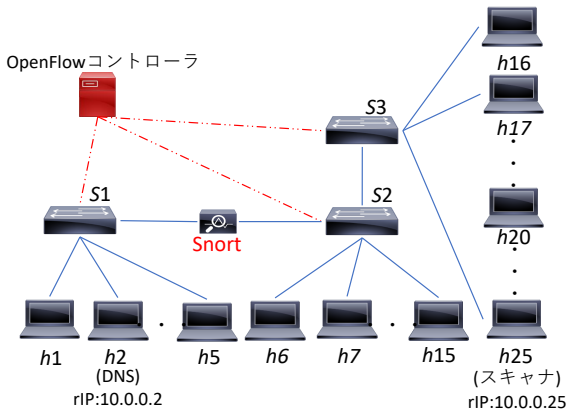

図 5 実験のネットワークトポロジ

測定した．次に，*rIP* での通信が許可されているホスト及び *vIP* での通信のみ許可されているホストから複数のホストに対して ICMP パケットの送受信におけるラウンドトリップタイム（RTT）と，TCP でのスループットを測定し，提案手法と既存手法でそれぞれ比較した．

4.2 実験環境

実験では，下記の物理マシン上にインストールされたネットワークエミュレータ Mininet[12] を利用して，図 5 の 3 スイッチ，25 ホストのネットワークトポロジを構築し利用した．

- OS: Ubuntu 20.04LTS
- CPU: Core-i5 6500 4 CORE
- RAM: 16GB

また，提案で利用する IPS として，Snort2.9[13] をインストールし，トポロジの OpenFlow スイッチ *S1* と *S2* の間に配置された仮想ホストから IPS として実行した．トポロジ上のホスト *h2* では DNS サーバを実行し，*rIP* とドメイン名の対応を登録することで，DNS を介した *vIP* の通知が機能するように設定した．*rIP* の範囲は，10.0.0.1 から 10.0.0.25 までとし，各ホスト *hx* の *rIP* が 10.0.0.*x* となるように設定した．なお，トポロジ中のすべてのリンクは 1000Mbps に設定した．

4.3 CPU 負荷の測定

OpenFlow コントローラに負荷をかけるため，スキャナ（*h25*）から負荷テスト用のスクリプトを実行した．これは，トポロジ内の他のすべてのホストに対して 3 回ずつ ICMP パケットを送信する動作を，並行処理で 15 回行うものであり，これによって 30 秒程度に渡って Packet-In メッセージが複数発行される．この時 OpenFlow コントローラが受信した Packet-In メッセージ数と，コントローラプロセスの CPU 使用率を既存手法と比較した．

図 6 にコントローラが 1 秒あたりに受信した Packet-In メッセージの数を示す．このグラフから，毎秒 100 個未満

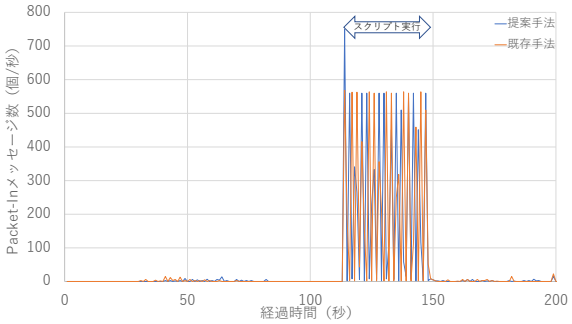

図 6 Packet-In メッセージ数の比較

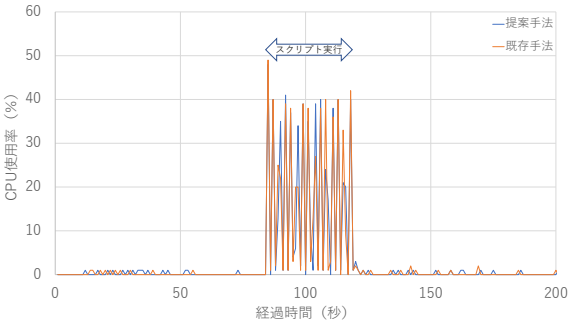

図 7 CPU 使用率の比較

であった Packet-In メッセージが，スクリプト実行中に断続的に毎秒 500 個以上となっていることが分かる．

次にコントローラプロセスの CPU 使用率を図 7 に示す．このグラフから CPU 使用率はスクリプト実行時の Packet-In メッセージの増加に伴い，40%程度まで増加していることが分かる．提案手法を利用した場合と既存手法を利用した場合でその負荷に大きな差は認められず，提案 MTD がコントローラの CPU にかける負荷は既存手法と同程度であると考えられる．

これらのことから，あるホストから複数のホストに対して高頻度で通信が発生すると，OpenFlow コントローラに比較的大きな負荷がかかると分かる．これは，多くの Packet-In メッセージを処理するためであると考えられる．また，図 6 に示したような Packet-In メッセージの増加には，MTD のアドレス変更処理に加えて，本手法の DNS を介して *vIP* を通知する機能の影響もあると考えられる．そのため，本実験よりもホスト数やスイッチ数の多い環境に提案する MTD を適用する場合，より性能の高いプロセッサや名前解決の実装の変更が必要となると考えられる．

4.4 通信遅延及びスループットの測定

通信遅延及びスループットの測定にあたり，図 5 のトポロジ内の複数のホストから ping コマンドを利用し，ICMP パケットの RTT を計測した．さらに，iperf コマンドを実行し，TCP のスループットを取得した．本研究では，DNS を介した *vIP* の通知機能と特定のホストを MTD の適用対象から除外する機能を実装している．そのため，本実験で

表 1 実験における各ホストの接続関係

	接続スイッチ	*rIP* での通信許可
h1-h3	*s1-s1*	許可-不許可
h1-h5	*s1-s1*	許可-許可
h3-h4	*s1-s1*	不許可-不許可
h1-h6	*s1-s2*	許可-不許可
h1-h15	*s1-s2*	許可-許可
h3-h6	*s1-s2*	不許可-不許可
h3-h15	*s1-s2*	不許可-許可
h1-h16	*s1-s3*	許可-不許可
h1-h25	*s1-s3*	許可-許可
h3-h16	*s1-s3*	不許可-不許可
h3-h25	*s1-s3*	不許可-許可
h6-h16	*s2-s3*	不許可-不許可

はそれらの処理も考慮した遅延及びスループットの測定を行う．そこで，*h1*，*h2*（DNS サーバ），*h5*，*h15*，*h25* を *rIP* での通信を許可するホストとして設定し，表 1 の場合について実験を行った．RTT の測定では，（D）各ホストがドメイン名でホストを指定して接続した場合と（I）IP アドレスでホストを指定して接続した場合について測定を行い，既存手法と比較した．iperf を用いたスループットの測定では，各ホストがドメイン名でホストを指定して接続した場合について測定を行い，既存手法と比較した．

図 8 に提案手法及び既存手法における RTT を示す．このグラフから提案手法，既存手法ともに，RTT の測定を行った *h1* から *h25* までのホストにおいて，それらの RTT はすべて 3.5 ミリ秒未満であることがわかる．また，本実験環境においては，（D）ドメイン名を利用してホストを指定した場合は，（I）IP アドレスを直接指定して接続した場合よりも RTT が小さくなる傾向を確認した．

さらに，*S1* に接続されたホストが *S2* や *S3* に接続されたホストにパケットを送信した場合，同一 OpenFlow スイッチに接続されたホストに対してパケットを送信した際の RTT よりも大きくなることがわかる．一方で，宛先ホストが *rIP* での通信を許可されたホストから他のホストへの通信と，*rIP* での通信が許可されていないホスト同士の通信に関して，RTT に大きな差が認められず，遅延に大きな影響を与えるのはパケットを転送する際に経由するスイッチの数であると考えられる．

また，パケットが IPS を通過する *h1-h6* 間の RTT とパケットが IPS を通過しない *h6-h16* 間の通信に関して，その RTT に大きな差はなく，MTD と IPS を併用する際に，IPS は遅延を大きく増加させる要因ではないと考えられる．

図 9 に提案手法及び既存手法におけるスループットを示す．このグラフから，提案手法におけるスループットは表 1 に示したホスト間の通信において，既存手法と同程度であるとわかる．また，図 9(A) に示す，通信が IPS を通過しないホスト間のスループットは，900Mbps 程度である一方で，図 9(B) に示す，通信が IPS を通過するホスト間の

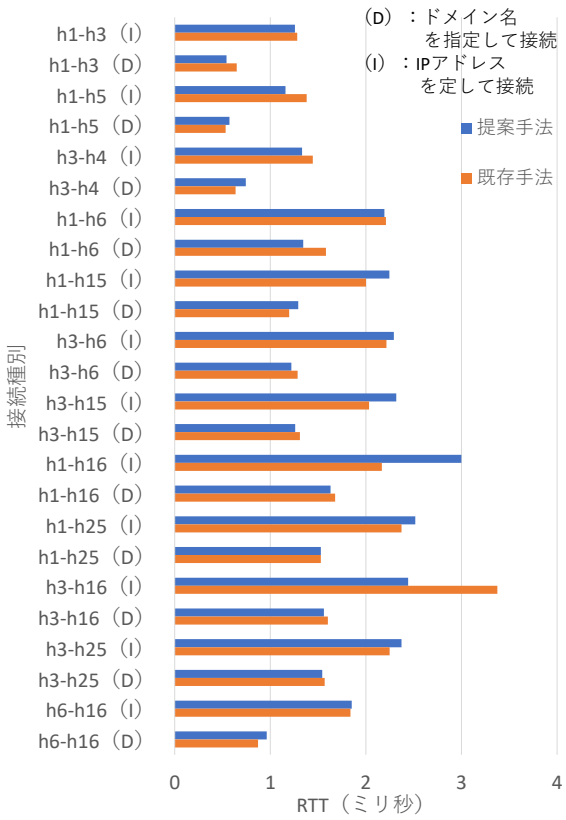

図 8 RTT の比較

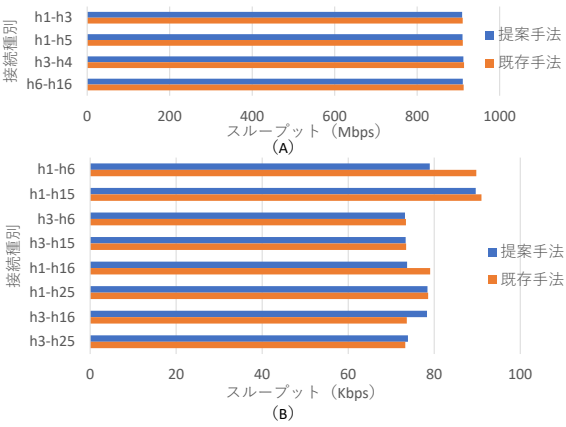

図 9 スループットの比較

スループットが著しく低いことがわかる．この時のスループットは 80Kbps 程度であり，アプリケーションレイヤの通信に大きな影響があると考えられる．ICMP の通信が IPS を通過する場合としない場合で遅延に大きな差はないため，本実験でトポロジ構築に利用した Mininet が，IPS を通過する TCP トラフィックのスループットに影響を与えていると考えられる．

これらの結果から，提案手法と既存手法のアドレス変更に関して，遅延，スループットに目立った差はなく，提案手法を利用して MTD を適用した場合も，ネットワーク内のホストの通信に大きな影響は及ぼさないと考えられる．一方，本実験環境において，通信が IPS を通過する場合に限

り，その TCP スループットが大きく低下する．そのため，実験環境をエミュレーションではなく物理スイッチと物理マシンを使ったものに移行する必要があると考えられる．

4.5 防御可能な攻撃に関する考察

提案手法はローカルネットワーク内で発生する同一サブネット内のホストに対するネットワークスキャンを対象とし，OpenFlow スイッチ間に IPS を配置し MTD と併用することで，それらのスキャンを防ぐ．そのため，図 5 の *h25* から *S1* 配下のホストに対するスキャンを IPS で防御できる一方，同一スイッチに接続されたホストに対するスキャンは IPS を通過しないため，検知・遮断ができない．

また，本手法は文献 [3] と同様に，DNS を *vIP* の通知に利用する．そのため，攻撃者は何等かの手段で FQDN を取得し，DNS を介してその時点で有効な *vIP* を入手することで，*vIP* を用いて接続できる．しかし，スキャン検知・遮断用の IPS は他のルールを設定して攻撃検知・遮断にも利用でき，本手法は既存手法と異なり IPS に対して透過的である．そのため，攻撃者が *vIP* を入手してもその攻撃が IPS を通過する場合，既知の攻撃を継続的に監視し，攻撃元をアドレス変化の影響を受けずに隔離できる．

5. おわりに

本研究では SDN の実装のひとつである OpenFlow を用いて，OpenFlow スイッチ間のパケットを時間変化しない *rIP* に基づいて転送する MTD 手法を提案した．これにより，MTD のアドレス変更が IPS にもたらす影響をなくし，IPS への透過性を確保することで，IPS と MTD との併用を実現し，様々なレートのスキャンを防御することが可能となった．評価するにあたって，CPU 負荷や通信遅延，スループットに着目し性能評価実験を行った．実験により，既存手法と同程度の CPU 負荷で MTD を実行できることを示し，既存手法と同程度の遅延であることを確認した．また，*vIP* の通知機能やインラインに設置した IPS がホストの遅延に大きな影響を及ぼさないことを確認した．スループットの測定においても，既存手法と同程度のスループットで通信が可能であると確認したが，その通信が IPS を通過する際にスループットの著しい低下が認められた．

今後の課題として，実際の運用を想定し，アドレス変更間隔を超えて通信する際に，通信の中断を防止する機能を実装する必要がある．この機能を実装し，より実環境に近い条件で各ホストを他のホストやデフォルトゲートウェイと通信させることで，アプリケーションレイヤに与える影響をより詳細に評価する．例えば，アドレス変更の過渡状態や，長時間に渡る通信時のスループットの変化などを評価する．また，IPS を経由する通信が正常に行えるように，物理機器によるテストベッド構築やエミュレーション環境の変更など，実験環境を再度検討する．

参考文献

[1] 株式会社カプコン：不正アクセスによる情報流出に関するお知らせとお詫び — 株式会社カプコン（オンライン），入手先 ⟨https://www.capcom.co.jp/ir/news/html/201116.html⟩（参照 “2021-08-27”）．

[2] U.S. National Science and Technology Council: Trustworthy cyberspace: Strategic plan for the federal cybersecurity research and development program, *Federal Cybersecurity: Strategy and Implementation for Research and Development*, pp. 69–99 (2011).

[3] Jafarian, J. H., Al-Shaer, E. and Duan, Q.: OpenFlow random host mutation: Transparent moving target defense using software defined networking, *Proceedings of the 1st ACM International Workshop on Hot Topics in Software Defined Networks*, pp. 127–132 (online), DOI: 10.1145/2342441.2342467 (2012).

[4] Nmap: タイミングとパフォーマンス—（オンライン），入手先 ⟨https://nmap.org/man/ja/man-performance.html⟩（参照 “2021-08-27”）．

[5] Al-Shaer, E., Duan, Q. and Jafarian, J. H.: Random host mutation for moving target defense, *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, Vol. 106 LNICS, pp. 310–327 (online), DOI: 10.1007/978-3-642-36883-7_19 (2013).

[6] Luo, Y. B., Wang, B. S., Wang, X. F., Zhang, B. F. and Hu, W.: RPAH: A moving target network defense mechanism naturally resists reconnaissances and attacks, *IEICE Transactions on Information and Systems*, Vol. E100D, No. 3, pp. 496–510 (online), DOI: 10.1587/transinf.2016EDP7304 (2017).

[7] Open Networking Foundation: Software-Defined Networking (SDN) Definition - Open Networking Foundation (online), available from ⟨https://opennetworking.org/sdn-definition/⟩ (accessed “2021-08-27”).

[8] Sharma, D. P., Kim, D. S., Yoon, S., Lim, H., Cho, J. H. and Moore, T. J.: FRVM: Flexible Random Virtual IP Multiplexing in Software-Defined Networks, *Proceedings - 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018*, pp. 579–587 (online), DOI: 10.1109/TrustCom/BigDataSE.2018.00088 (2018).

[9] Van Leeuwen, B., Stout, W. M. and Urias, V.: Operational cost of deploying Moving Target Defenses defensive work factors, *Proceedings - IEEE Military Communications Conference*, Vol. 2015-Decem, pp. 966–971 (online), DOI: 10.1109/MILCOM.2015.7357570 (2015).

[10] Shi, Y., Zhang, H., Wang, J., Xiao, F., Huang, J., Zha, D., Hu, H., Yan, F. and Zhao, B.: CHAOS: An SDN-Based Moving Target Defense System, *Security and Communication Networks*, Vol. 2017 (online), DOI: 10.1155/2017/3659167 (2017).

[11] Ryu SDN Framework Community: Ryu SDN Framework (online), available from ⟨https://ryu-sdn.org/⟩ (accessed “2021-09-05”).

[12] Mininet Project Contributors: Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet (online), available from ⟨http://mininet.org/⟩ (accessed “2021-08-27”).

[13] Cisco: Snort - Network Intrusion Detection & Prevention System (online), available from ⟨https://www.snort.org/⟩ (accessed “2021-08-27”).