Vol.2020-CSEC-91 No.7
Vol.2020-SPT-40 No.7
Vol.2020-EIP-90 No.7
2020/11/25

分散型SDN制御プレーンを標的とするDDoS攻撃の緩和手法の提案

渡邉 俊介[1,a)]　今泉 貴史[2,b)]

概要：SDN では，ネットワークを構成する全スイッチを少数のコントローラが管理するため，コントローラに負荷が集中しやすい．そのことに着目した攻撃として，スイッチからコントローラへの制御メッセージの送信を誘発するパケットを大量に送り付け，コントローラに負荷をかけることを狙った，SDN-DoS 攻撃がある．この攻撃に対して，特定のコントローラへと負荷が集中することに注目し，複数のコントローラで処理を分散して負荷の緩和を狙う手法が提案されている．しかし，この手法は攻撃がプレーン全体へと影響する SDN-DDoS 攻撃に対しては，効率的に機能しない可能性がある．本稿では，SDN-DDoS 攻撃パケットを効率的に破棄する手法を提案し，性能の検討を行う．

キーワード：SDN, DDoS, コントローラ

Proposal of mitigation method for DDoS attacks targeting the distributed SDN control plane

Shunsuke Watanabe[1,a)]　Takashi Imaizumi[2,b)]

Keywords: SDN, DDoS, Controller

1. はじめに

ネットワークを管理する技術の一つに，SDN(Software Defined Network) がある．SDN は，従来のネットワークにおける役割を，アプリケーションと制御プレーン，データプレーンの 3 層に分離したアーキテクチャである．データプレーンは，スイッチと呼ばれる機器によって構成され，データ転送を行う．制御プレーン，アプリケーションプレーンはコントローラと呼ばれる機器によって構成され，データプレーンを制御する．また，複数のコントローラによって構成された制御プレーンは分散型制御プレーンとも呼ばれ，主に大規模なネットワークで利用される．SDN を用いることで，ネットワークの一元的な管理や，柔軟な構成が可能となる．

SDN の制御方式としては，リアクティブ型とプロアクティブ型の 2 つの方式がある．プロアクティブ型では，事前にデータプレーンへと転送ルールを設定し，未定義の転送ルールのパケットは破棄される．リアクティブ型では，未定義の転送ルールのパケットについてコントローラへと問い合わせ，制御ルールを作成する．リアクティブ型制御は，プロアクティブ型制御よりも動的にネットワークを制御することが可能な一方で，様々な問題を抱えている．特に，SDN は，ネットワークを構成する多数のスイッチを少数のコントローラが管理することや，コントローラはス

[1] 千葉大学大学院融合理工学府
Graduate School of Science and Engineering, Chiba University
[2] 千葉大学統合情報センター
Institute of Management and Information Technologies, Chiba University
[a)] watashun_201@chiba-u.jp
[b)] imaizumi_takashi@faculty.chiba-u.jp

Vol.2020-CSEC-91 No.7
Vol.2020-SPT-40 No.7
Vol.2020-EIP-90 No.7
2020/11/25

イッチと比べ複雑な処理を行うため，コントローラへと負荷がかかりやすい．これを利用し，コントローラへと意図的に制御メッセージを送り付け，負荷を与える SDN-DoS 攻撃 [1] という手法が存在する．SDN-DoS 攻撃の一つとして，未定義ルールのパケットを大量にスイッチへと送る攻撃がある．この攻撃では，攻撃者は未定義ルールのパケットを大量にスイッチへと送り付ける．それを受け取ったスイッチはコントローラへと制御ルールを問い合わせ，それによってコントローラに負荷が集中し，最悪の場合ダウンしてしまう．

ルールの定義にかかる負荷は，そのコントローラが管理するスイッチが増えるほどに増加する．そのため，複数のコントローラによって管理トポロジーを分割する分散型制御プレーンはコントローラの負荷を減らすために有効な手法である．分散型制御プレーンにおいて，SDN-DoS 攻撃への対策手法として [2][3] 等が提案されている．これら手法は，SDN-DoS 攻撃下での負荷の偏りに注目し，被攻撃トポロジーの分割や，被攻撃コントローラの負荷を分散することで SDN-DoS 攻撃の緩和を図っている．また，松尾らの手法 [2] では，負荷が集中した場合であっても，被攻撃コントローラの転送能力が負荷分散のネックになってしまっていた．

しかし，これらの手法は，制御プレーン全体を狙う SDN-DDoS 攻撃の緩和を行うことはできない．SDN-DoS 攻撃は単一の端末から行われるが，大規模なネットワークの制御プレーンを攻撃する場合，単一の端末からでは攻撃リソースが不十分である可能性が高く，複数端末からの攻撃である，DDoS 攻撃として行われることが予想される．この時，必ずしも負荷が特定エリアに集中するとは限らず，特定エリアの負荷分散では十分に緩和できない可能性がある．

そこで，本研究では，攻撃発生時に流入する未定義パケットの特徴に基づいて，攻撃パケットを効率的に破棄する手法を提案し，SDN-DDoS 攻撃の影響を緩和することを狙う．

2. OpenFlow

SDN を実現する技術の一つである OpenFlow?は，コントローラとスイッチ間の通信プロトコルを定めたものである．OpenFlow に準拠したスイッチ (OpenFlow スイッチ) は，条件にマッチしたパケットに対して指定されたアクションを行う．パケットマッチングのルールとアクションの組をフローと呼び，OpenFlow スイッチはそれらをフローテーブルに格納している．OpenFlow では，OpenFlow コントローラから，スイッチの持つフローテーブルへとフローを追加・修正・削除することでネットワークを制御する．フローテーブルにパケットとマッチするルールとアクションの組がない場合をテーブルミスと呼ぶ．テーブルミスが発生した場合の動作は，OpenFlow のバージョンによって異なる．OpenFlow1.2 までは，テーブルミスを起こしたパケットを基に Packet_In という問い合わせメッセージを生成し，コントローラに送信する．OpenFlow1.3 以降では，テーブルミスを起こしたパケットはデフォルトでは破棄される．OpenFlow を用いてリアクティブ型制御を行う場合，フローテーブルに存在しないルールのパケットへのアクションとしてほとんどの場合 Packet-In を設定する．そのため，攻撃者は，テーブルミスを起こすようにパラメータを変えながらパケットを送信する．

3. 分散型制御

OpenFlow はコントローラとスイッチ間の仕様を定めているが，コントローラの構成については決められていない．そのため，通常単一のコントローラがネットワーク制御を担うが，数百台規模のスイッチで構成される様なデータプレーンを制御する場合に，コントローラへの負荷が増大してしまうといった，スケーラビリティの問題がある．

スケーラビリティを向上させるために，複数台のコントローラによって制御を行う，分散型制御プレーンが提案されている．[4][5] これらの手法では，データプレーンを複数のエリアに分割し，1 つのエリアを 1 台のコントローラが管理する．コントローラは自身が管理するエリア内のトポロジー情報のみを収集し，フローを制御することから，スケーラビリティを向上させることができる．エリア外の情報 (宛先ホストの位置など) が必要な場合は，他のコントローラやデータベースへと問い合わせる．一方で，分散型制御は，SDN の元々の利点である，ネットワークの一元管理等を達成することが難しくなっている．

分散型制御プレーンの構成は，並列型と階層型に分けられる．並列型は，コントローラのインスタンス同士が，協調してネットワークの制御を行う．並列型のコントローラ構成の代表的なものとして，Onix や ONOS[6] が挙げられる．階層型では並列型と同様に，分割した各エリアを各コントローラが管理するが，コントローラを管理する上位のコントローラが設けられている．階層型は，前述したネットワークの一元管理を並列型と比べて達成しやすいが，ネットワーク構成の決定や，トポロジーの分割等が難しくなっている．

4. SDN-DoS 攻撃と関連研究

分散型制御プレーンへの SDN-DoS 攻撃の対策としては，攻撃分布の偏りに注目して攻撃を緩和する ElastiCon[3] や，松尾らによるオーケストレータを用いた手法 [2] がある．

ElastiCon では，高負荷コントローラが存在する場合，その管理スイッチを別のコントローラが代わりに管理することで，コントローラの負荷を減らす．

松尾らの手法では，高負荷コントローラが存在した場合

Vol.2020-CSEC-91 No.7
Vol.2020-SPT-40 No.7
Vol.2020-EIP-90 No.7
2020/11/25

に，そのコントローラに流入する Packet-In を，プレーンを構成する全てのコントローラと接続したオーケストレータと呼ばれる機器を介して，別の低負荷なコントローラへと送り，代わりに処理させる．図 1 に松尾らの手法で提案されたアーキテクチャを示す．

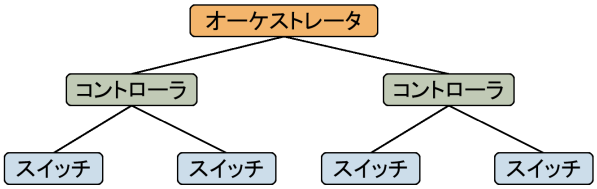

図 1 松尾らの手法で提案されたアーキテクチャ
Fig. 1 Architecture proposed by Matsuo et al.

これら手法は，攻撃の分布や各コントローラの負荷に偏りがあることを前提としている．SDN-DoS 攻撃は，複数の攻撃端末を用いて DDoS 攻撃として行うことが可能である．SDN-DDoS 攻撃が行われた場合，分散型制御プレーンを構成するほとんどのコントローラが高負荷になるため，これら手法は有効に機能しない．

5. SDN-DF による SDN-DDoS 攻撃緩和手法

SDN-DDoS 攻撃パケットの特徴として，パケットのパラメータのランダム性の高さが上げられる．一度，ネットワークに流入したパラメータのパケットは，制御プレーンが正常に動作している場合には処理され，スイッチのフローテーブルへと制御ルールが設定され，以降のパケットについては，コントローラに問い合わせることなく，ネットワークを通過する．そのため，SDN-DDoS 攻撃者は，出来るだけ多くの Packet-In を狙うために，同一のパラメータのパケットを送信せず，ランダムなパラメータで偽装したパケットを送信する．一方で，ネットワークは独立した機器が制御する特性上，パケットドロップを許容するような設計がなされている場合がほとんどである．例えば，TCP における 3 ウェイハンドシェイク中に SYN パケットがドロップし，SYN/ACK パケットが返答されなかった場合，OS によって異なるが，数秒程待った後に再送するように設計されている．そのため，正常な通信者はパケットドロップに対して同一パラメータのパケットを再度送信する．

これらの違いから，初めて流入したパラメータのパケットはほとんどが攻撃パケットであり，2 回目以降に流入したパラメータのパケットのほとんどが正常な通信者のパケットであると考えられる．そのため，初めて流入したパラメータのパケットを破棄し，2 回目以降に流入したパラメータのパケットに対しては，PacketIn の処理を行うことで，攻撃パケットに対して複雑な処理を行わなう可能性を減らし，攻撃下での正常なパケットの処理量を増やすことが出来ると考えられる．

そのためには，パケットのパラメータを一定時間保存する必要があるが，全てのパケットのパラメータを保存した場合，非常に膨大な量のメモリリソースを要する他，探索にも非常に時間がかかってしまい，それ自体がボトルネックになってしまう可能性がある．よって，ハッシュ関数による効率的なデータの管理が適切である．

本研究では，SDN-DDoS 攻撃の対策手法として，ハッシュ関数と配列を用い，流入パケットが攻撃パケットかどうかを低負荷に判定できる SDN-DF(DistributedFiltering) を提案する．

提案手法では，松尾らの提案したアーキテクチャを想定とする．松尾らの提案したアーキテクチャでは，分散型制御プレーンを構成する各コントローラへとオーケストレータが，TLS 通信を介して指示を行う．少数のコントローラの CPU 使用率の上昇等から，DoS 攻撃を検知すると，オーケストレータからの指示によって，各コントローラが負荷分散モードへと移行していたが，負荷の多いコントローラの割合が増えた場合には，SDN-DF を適用する様に指示する．よって，各コントローラはオーケストレータの指示に従って次の図 2 の様に状態を遷移する．

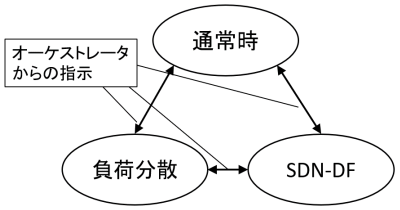

図 2 SDN-DF における遷移図
Fig. 2 Transition diagram in SDN-DF

SDN-DF を適用するように指示された各コントローラは，PacketIn メッセージが流入した際に，一度，そのパラメータをハッシュ関数にかけ，ハッシュ値に対応した番地のビットを確認する．対応した番地のビットが立っていない場合，初めて流入したパケットとして，破棄する．ビットが立っている場合には，正常なパケットとして PacketIn に対応した処理を行う．前述したように，正常な通信者であれば，再送が行われ，2 回目以降のパケットからは普段通りに通信が可能である．一方で，攻撃者はその攻撃のランダム性から，プレーンにパケットを処理させるためには，ハッシュの衝突を起こさせなければならない．ランダムな値によるハッシュの衝突を大量に起こすためには，相応のパケット量が必要であり，攻撃によって制御プレーンがダウンしてしまう状況を避けることが出来ると考えられる．

また，評価において後述するが，本手法のフィルタはパ

Vol.2020-CSEC-91 No.7
Vol.2020-SPT-40 No.7
Vol.2020-EIP-90 No.7
2020/11/25

```
// 配列の長さ : N
// 配列 : L
// ハッシュ関数 : F
一定時間毎に実行:
    L の全ての要素を 0 で初期化
Packet_In が発生 :
        P ← PacketIn から抽出したパラメータ
    if L[F(P) % N] == 0:
        L[F(P) % N] = 1
        PacketIn を破棄
    else:
        PacketIn を処理
```

図 3　SDN-DF のアルゴリズム
Fig. 3　SDN-DF algorithm

ケット数の多さと共に性能が低下する．そのため，一定時間配列のリセットが必要である．

提案手法における SDN-DDoS 攻撃の対策アルゴリズムを以下の疑似コードを図 [3] に示す．

6. SDN-DF の評価・実験

提案手法のアルゴリズムでは，攻撃者が攻撃パケットを処理させるためには，ハッシュ値を予測し，意図的に衝突させる他に，非常に大量のパケットを送る事で，ランダムな値同士の衝突が起こり，偶発的にパケットを処理されることがある．

例えば，手法を適用してから初めてパケットが流入したとする．この時の配列のビットは全て 0 であり，確実に破棄される．しかし，次のパケットが流入した場合には，配列内に 1 つだけビットが 1 の要素が存在している．この時，同じ配列の番地をハッシュ値が示し，パケットが処理される可能性は低いが，更に次のパケットの場合には，1 の要素が 2 個あり，処理される可能性が高くなる．そのため，持続的に攻撃が行われた場合，衝突確率が上がっていき，最終的にほとんどのパケットを処理することになる．

本手法を評価するために，配列長に対して，配列内の 1 を示すビットの割合を占有率 α とおく．攻撃のランダム性，ハッシュ関数への予測の困難さから，ハッシュ値は理想的に，均等に散らばると仮定する．この時，その値はほとんど衝突せず，一様に配列を埋めていくことになる．そのため，占有率 α は配列長 N と流入パケット数 M を用いて，以下の式によって近似できると考えられる．

$$\alpha = 1 - \left(1 - \frac{1}{N}\right)^{M} \tag{1}$$

実際に確認として，計算機を用い，配列長 1000 に対して 2000 のランダムなパラメータのパケットを軽量なハッシュアルゴリズムとして知られる FNV-1 というハッシュ関数に通し，占有率を計測した．以下の図 [4][5] に実験結果を示す．

この α を前提に，フィルタの特性を評価する．

SDN-DDoS 攻撃による影響は，攻撃パケットの量とその速度や，配列長とそのリセット間隔等によって決定される．また，PacketIn 処理毎の負荷は，SDN-DF による負荷と，本来の処理の負荷が考えられる．SDN-DF によって全てのパケットが破棄された場合のパケット処理速度を Ta，全てのパケットが破棄されずに処理された場合のパケット処理速度を Tb とすると，全体のパケット処理速度 T は

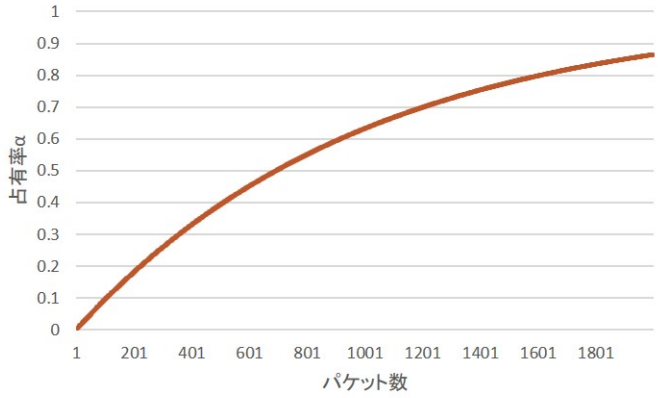

図 4　占有率の近似式
Fig. 4　Approximate formula of α

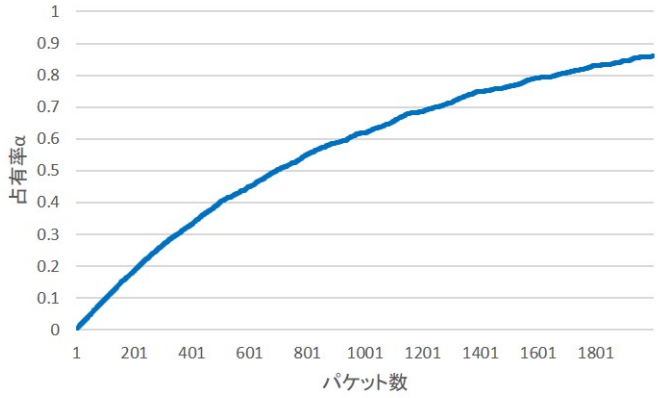

図 5　計算機によって得られた占有率の例
Fig. 5　sample date of $\alpha calicu letedby$

$$T = (1 - \alpha) \times T_a + \alpha \times T_b \tag{2}$$

となる．ハッシュ化と攻撃の判定による負荷は，ネットワーク規模に関わらず，ほぼ一定であるのに対して，パケット処理による負荷は，ネットワーク規模に応じて増大していく．そのため，$Ta < Tb$ であり，占有率 α が低いほど，処理速度が上がる．

また，占有率 α がパケット数に対して増大していくことから，SDN-DF による，DDoS 攻撃下での処理速度は時間と共に低下していくため，ある一定時間毎に配列のリセットが必要となる．その時間は，ネットワーク設計によるが，最低限正常な通信者がタイムアウトを確認し，再送を行うまでの時間が必要であると考えられる．昨今のネットワークにおいては，RTT に対して，タイムアウトが非常に長く

Vol.2020-CSEC-91 No.7
Vol.2020-SPT-40 No.7
Vol.2020-EIP-90 No.7
2020/11/25

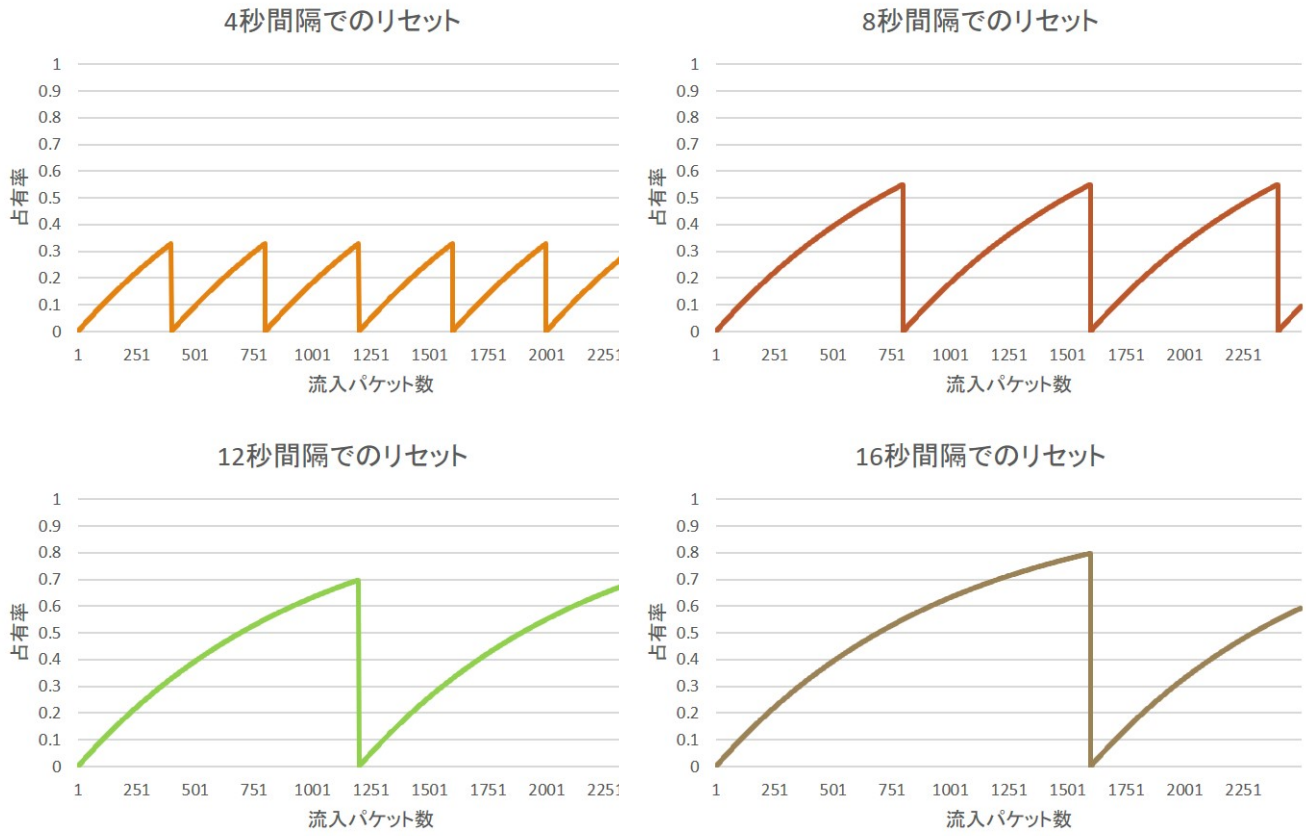

図 6　仮定した状況における占有率の遷移の例
Fig. 6　Example of occupancy transition in a hypothetical situation

設定されている．例えば，TCP の再送までの時間は，OS によるが，数秒程 (例えば windows であれば 3 秒が標準である) に設定されている．少なくとも，攻撃パケット流入速度を予想し，配列長を確保して，その再送を待つ時間までの間，一定以上の割合で攻撃パケットをドロップ出来るようにする必要がある．

例えば，配列長が 2000 であるコントローラに対して，攻撃者は秒間 100 パケットを送ってくる．と想定した場合に，リセットの時間を各 4 秒, 8 秒, 12 秒, 16 秒と設定した場合の占有率は図 6 の通りの様に，α の遷移の一定時間までを繰り返した様な形になる．

4 秒, 8 秒, 12 秒, 16 秒毎の占有率は最大でそれぞれ，0.3, 0.55, 0.7, 0.8 となっており，次第に占有率の増加量が減少していっている．

7. 考察

本手法を実際に運用する際に複数のパラメータが関わってくる．配列長とそのリセット時間や，攻撃パケットとその速度などが主に重要なパラメータとなる．特に，リセット時間は，正常なパケットを通すためにある程度確保しなければならない．これらを決定することは難しく，ネットワークの設計に大きく依存してしまう．そのため，根本的な値である，パケット処理速度と，占有率 α からこれらパラメータを決定することが良いと考えられる．パケット処理速度 T における，各 Ta,Tb を求めた後に，どれくらいの値の α まで，すなわち，どれくらいの処理速度の低下を認めるかを一つ基準に決定する．実際に攻撃が行われ，SDN-DF が適用された場合に，α が一定値まで達するごとに配列をリセットする．タイムアウトなどの最低限確保しなければならない時間を過ぎた場合には，配列長を長くする．この様な方法を取ることで，配列長などのメモリリソースや，リセット時間の決定等の難しい判断を決定しやすくなると考えられる．

SDN-DF において，判定に用いる配列は各 1 ビットで良いため，非常にメモリ効率も良く，かつ，SDN-DF そのものの処理は，ネットワークにおけるパス計算等，PacketIn への処理に対して，非常に処理の軽い物であるため，元々の PacketIn 処理が大きくなる大規模なネットワークに適しているものであり，分散したコントローラが本手法を行うことで，より適切に DDoS 攻撃による影響を緩和できると考えられる．

8. 今後の課題

本研究によって SDN-DDoS 攻撃を緩和する手法である

Vol.2020-CSEC-91 No.7
Vol.2020-SPT-40 No.7
Vol.2020-EIP-90 No.7
2020/11/25

SDN-DF を提案した．SDN-DF は効率が良い反面，攻撃者が SDN-DF の存在を考え，2 パケット同時に送信する，といったケースも考えられる．この場合も，攻撃者の攻撃パケット数を半分に減らすことが出来るが，各配列の要素に，同時にタイムスタンプを付加することによって，この攻撃を緩和できる可能性がある．2 パケット同時に送信される場合，本来のタイムアウトよりもかなり短い間隔で同一パラメータのパケットが流入する．そこから判断をし，破棄をすることによって，攻撃の緩和を狙えるが，各攻撃パケットに対して 1 ビットずつで対応出来る SDN-DF と異なり，より多くのビットが必要になるため，メモリなども攻撃対象となりうる．

また，SDN-DF は，SDN-DDoS 攻撃下を想定しているが，SDN-DDoS 攻撃の判定方法等も検討する必要がある．SDN-DF は，正常な通信者であっても，初めの 1 パケットを破棄される．DDoS 攻撃下であるため，そもそもそのパケットが破棄される確率は高いが，実際にどれくらいの正常な通信者を通すことが出来るのか，また，遅延時間等，通信の特性への影響はどのようになるか等を調べる必要がある．

また，SDN-DF で対応できない SDN-DDoS 攻撃もある．例えば，1 つ 1 つのパケットサイズを大きくすることによって，帯域を取る攻撃などは，対処ができない．スイッチから送信された時点で，コントローラ側へ流入してしまうのは避けられないため，スイッチの設定を変更するなど，また別の手法も必要になると考えられる．

参考文献

[1] Li, W., Meng, W. and Kwok, L. F.: A Survey on OpenFlow-based Software Defined Networks, J. Netw. Comput. Appl., Vol.68, No. C, pp. 126-139 (2016)
[2] 松尾 亮輔, 今泉 貴史 - 分散型 SDN 制御プレーンへの DoS 攻撃の緩和手法の提案, 2017-IOT-39
[3] Advait Dixit. Fang Hao. Sarit Mukherjee. - ElastiCon; an elastic distributed SDN controller. ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS). 2014.
[4] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T. and Shenker, S. - Onix: A Distributed Control Platform for Large-scale Production Networks, Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, pp. 351-364 (2010).
[5] Tootoonchian, A. and Ganjali, Y. - HyperFlow: A Distributed Control Plane for OpenFlow, Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, pp. 3-3 (2010).
[6] Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W. and Parulkar, G. - ONOS: Towards an Open, DistributedSDN OS, Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN'14, New York, NY, USA, ACM, pp. 1-6 (2014).