分散処理によるOpenFlowを用いた端末非依存なネットワーク攻撃検知手法

宮崎 亮輔[1]　川本 淳平　松本 晋一[2]　櫻井 幸一[1]

概要：近年，サイバー攻撃の種類は多種多様に渡っており，可能な限り多くの種類の攻撃を検知する技術が必要である．そこで我々は，OpenFlow とマルチエージェントシステムを組み合わせることでネットワーク攻撃を広く検知する手法を提案してきた．しかし 1 台の OpenFlow コントローラが 1 台の OpenFlow スイッチを制御する完全分散型の手法であったため，ネットワーク攻撃が行われた場合に，全てのコントローラから情報を収集しなければならず，通信の迂回や遮断等の迅速な対応や正確な検知が難しいという問題があった．本稿では，複数のコントローラ同士で適切な情報を共有することで，効率的な検知処理及び負荷分散を実現する手法を提案する．

キーワード：攻撃検知，SDN，OpenFlow

Host Independent and Distributed Detection system of the Network Attack by Using OpenFlow

Ryosuke Miyazaki[1]　Junpei Kawamoto　Shinichi Matsumoto[2]　Kouichi Sakurai[1]

Abstract: In recent years, cyber attacks are in various ways, and we must detect as many kinds of attacks as possible. Therefore, we propose to combine OpenFlow with multi agent system and archive host independent detection system of the network attack. However, it is a completely distributed method in that one OpenFlow controller controls one OpenFlow switch. Thus, we must gather information from all switches when attacks take place, which makes it difficult to respond quickly and detect precisely. In this paper, we propose to share appropriate information among switches, which enables an efficient detection and a load distribution.

Keywords: Attack Detection, SDN, OpenFlow

1. はじめに

近年，コンピュータネットワークは急速に普及してきている．それに伴い，以前は愉快犯によるサイバー攻撃が主であったのに対し，近年では愉快犯に加えて金銭や個人情報を狙ったサイバー攻撃へと，攻撃形態が多様化してきている．その一方で，セキュリティ対策ソフト等を積極的に導入しないような，ネットワークセキュリティに対する意識が低いユーザが存在する．そういったユーザに対しても有効なセキュリティを提供することが必要である．

多様化する攻撃に対して有効な手段として，従来様々な手法が確立されてきたが，その中でも，マルチエージェントシステムを用いて攻撃を検知する先行研究として，Ant-Based Cyber Defense(ABCD)[1][2][3] が提案されている．しかし，この手法では，防衛対象の各端末にソフトウェアの導入を必須としているので，ネットワークに接続する機能を持った組み込み機器に対して適用することが出来ないという問題が存在する．

そこで，我々はマルチエージェントシステムを用いた攻

[1] 九州大学
Kyushu University
[2] 公益財団法人 九州先端科学技術研究所
Institute of Systems, Information Technologies and Nanotechnologies

撃検知を，Software-Defined Network（SDN）技術を用いることで機器を改変することなく導入する手法 [4][5][6][7] を提案した．しかしこの手法では，1 台の OpenFlow コントローラが 1 台の OpenFlow スイッチを制御しており，先行研究である ABCD と同様に完全分散型制御である．したがって，OpenFlow コントローラ同士は互いに独立して検知を行っており，局所的な攻撃の検知に限定されるという問題が存在する．複数の視点で攻撃検知を行う手法として，Xiao らは，ネットワーク上の複数箇所でトラフィックを監視することで，ネットワークの全体的な変化を検知し，従来の単一視点侵入検知システムでは検知できないようなネットワーク攻撃を検知する手法 [8] を提案している．

本論文では，OpenFlow コントローラ同士で適切な情報交換を行い，従来の手法では検知が困難であった広範囲なネットワークに対する攻撃を検知する手法を提案する．1 台の OpenFlow コントローラは複数の OpenFlow スイッチを制御し，各コントローラの担当ネットワーク内の情報を始めとした様々な情報を互いに交換することで，広範囲なネットワークへのより幅広い攻撃の検知が可能となった．

2. 先行研究

攻撃からシステムを防御する手法の一つに，Moving Target Defense(MTD)[9][10] と呼ばれる手法が存在する．MTD とは，防御システムの Attack Surface[11] を変化させることで，攻撃者にシステムの動作を予測されにくくしている，より非決定的で，より動的で，より多様的な防御システムである．Jereme らは，マルチエージェントシステムを用いてエージェントの密度により攻撃を検知する MTD を提案しており [1][2]，各々の端末に対してソフトウェアを導入している．この手法の概略図を図 1 に示す．このシステムではセンサーアントと呼ばれるエージェントが存在している．端末を監視するソフトウェアと，各々で異なるパラメータを持つセンサーアントの 2 つを組み合わせることで，端末に攻撃があった際に，その端末にセンサーアントが集まるシステムとなっている．この手法が見る値は CPU 使用率やメモリ使用率等の記述的な値であるので，未知の攻撃に対しても検知できるという点において優れている．しかし，この手法は各端末に対してソフトウェアを導入することが必要であり，組み込み機器の様な，リソースの限られた端末に対しては導入することが難しいという問題がある．

そこで，我々はその問題点を解決するために，Jereme らの手法と SDN を組み合わせる手法を提案した [4][5][6][7]．SDN として OpenFlow を用いることで，各端末に対してソフトウェアを導入する必要がなくなり，組み込み端末の様なリソースの限られた環境に対しても Jereme らの手法を導入することが可能となった．しかし，各コントローラは完全分散制御で独立して動作しており，ネットワーク攻撃の分析は局所的なトラフィックで判断するのみに留まっていた．そこで今回我々は，OpenFlow コントローラ同士で情報を交換し，大域的なネットワーク攻撃に対する有効性を示した．

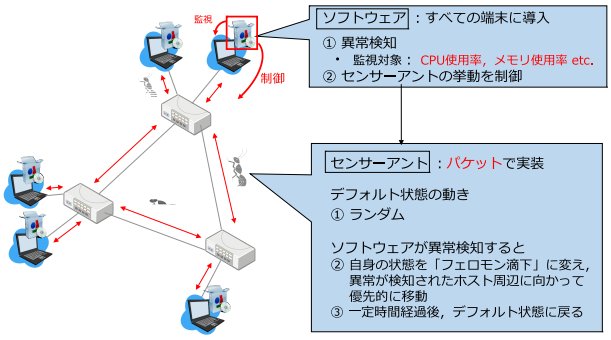

図 1　Ant-Based Cyber Defense の概念図

表 1　OpenFlow スイッチの持つパラメータ

パラメータ名	説明
:ant_count	現在そのスイッチに留まっているアントパケットの数
:pheromone_count	スイッチ間のフェロモンの値

3. SDN によるマルチエージェント方式の攻撃検知手法

この章では，先行研究 [1][2][3] に SDN を適用するために作成した OpenFlow コントローラ及びそれに付随するシステムに関して説明する．

3.1 OpenFlow スイッチについて

本手法で使用する OpenFlow スイッチは，自然界のアリでは巣を表しており，表 1 に示すパラメータを保持している．ここで:pheromone_count とは，スイッチ間に存在する値のことで，各スイッチのポート毎に存在する．このパラメータは自然界のアリが出すフェロモンを模している．3.2 節にて後述する，:ant というパラメータが 1 であるアントパケットが通過する度に 1 ずつ加算されていく．また，この値は時間と共に蒸発するようにしてあり，その蒸発係数を e とする．e は 5[秒] おきにかけられる．今，:pheromone_count = 1 とし，この値が T[秒] 後に少なくとも ε まで残っていることを保証する為には，$e > (\varepsilon)^{\frac{5}{T}}$ であればよい．:ant_count の扱いについては，3.4.1 節にて後述する．

3.2 アントエージェント

本手法では，表 2 に示す情報を持たせたアントエージェントが，コントローラ内部で再現された，スイッチと同じトポロジのグラフ上を動き回ることで，攻撃検知を行っている．各コントローラは，内部に自分が制御するネットワークのトポロジを再現したグラフを持っており，そのグ

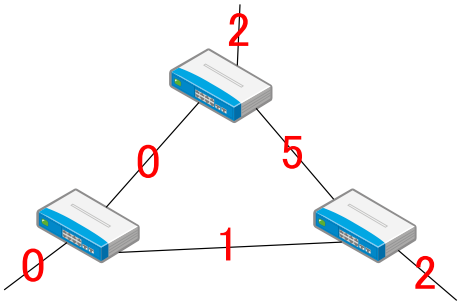

図 2　フェロモンカウント

表 2　アントエージェントの持つパラメータ

パラメータ名	説明
:ant	アントの状態を表す． 0: デフォルト 1: フェロモン滴下中
:nest	エージェントが生起したスイッチの識別子
:life_count	エージェントの余命

ラフ上をアントエージェントは動作する．各々のアントエージェントは寿命を持っており，1 回ホップする毎に齢を取っていき，異常の報告を受けないまま寿命に達した時点で消滅する．:ant はアントエージェントの状態を表している．0 はデフォルト値を，1 はフェロモン滴下中を表している．:nest はアントエージェントが生起したスイッチの識別子である．ここで識別子とは OpenFlow スイッチの datapath_id を指す．:life_count はアントエージェントの寿命を表している．アントエージェントの総数を調整する事と，アントエージェントの移動範囲を設定する事を目的としたパラメータである．各々のパラメータの扱いは，3.4.1 節にて後述する．

3.3　OpenFlow コントローラ

提案手法の為に用いる OpenFlow コントローラ（以下，コントローラ）について説明する．

3.3.1　トラフィックの監視・解析

各スイッチに接続されているホストのトラフィックをトラフィック解析モジュールで常に監視している．監視する対象は表 3 の通りである．なお，全て単位時間あたりの値で扱う．各スイッチ上にトラフィック解析を行うモジュールが存在し，ここで上記の要素を常に監視している．コントローラはこのトラフィック解析モジュールから上記の値の報告を受けている．コントローラはアントエージェントがスイッチを訪れる度に，上記の要素からランダムに 1 つの要素を選択する．選択された要素と報告された値を比較し，予め設定した閾値を越えると自状態を「異常」とする．この状態が「正常」か「異常」かで，アントエージェントの送り方が異なる．その送り方については，3.4.1 節で示す．

表 3　トラフィックの監視対象

フレームサイズが 300[byte] 未満のパケットの割合
宛先 IP アドレス種類数
送信元ポート番号種類数
宛先ポート番号種類数
フレームサイズの分散

表 4　コントローラ間で共有する情報

境界スイッチ間のアントエージェントの移動情報
監視要素の値
トポロジ情報

3.4　コントローラ同士の情報共有

コントローラがそれぞれ独立して異常を判定する場合，その判定はコントローラの持つ情報に限定して行われる．実際のネットワーク攻撃は単一ホストを対象にしたものに限らず，水平ポートスキャンや DDoS など，幅広いネットワークを対象として行われる．その様な攻撃にも対応するためには，コントローラ同士で情報を共有し，広域でのネットワーク情報を保持する必要がある．提案手法では，コントローラ同士で表 4 に示す情報を共有している．

アントエージェントの移動は全てコントローラ内で行われるので，コントローラが担当するネットワークを跨いでアントエージェントを転送する場合，その情報をネットワーク越しに相手コントローラへ伝達する必要がある．即ち，コントローラが担当するネットワークの境界スイッチ間でアントエージェントを転送する時である．したがって，境界スイッチ間のアントエージェントの移動情報を交換する必要がある．この移動情報には，移動したアントエージェントの表 2 に示す情報も含まれる．更に，コントローラは各々が保持しているスイッチ毎の監視要素の値を共有する．コントローラが保持する自ネットワーク内の値と，他コントローラから受信した他ネットワーク内の値に相関が見られる場合は，当該スイッチの状態を「異常」と判断する．即ち，スイッチの状態異常の判断は，表 3 の値の閾値判定以外にも，コントローラ間の値の相関によっても行う．トポロジ情報の共有は，:ant = 1 であるアントエージェントの転送及び，システムがネットワーク攻撃を検知しアラートを発生させた後のトラフィック遮断や迂回に用いる．

3.4.1　アントエージェントの制御

先行研究 [1][2] では，センサーアントと呼ばれるエージェントパケットの制御アルゴリズムが公開されていない．したがって，自然界のアリの動き [*1] を参考に，アントエージェント制御のアルゴリズムを以下の様に設計した．各スイッチに来たアントエージェントは，図 3 に示すフローチャートに従って制御される．ここで，図中の巣とはアン

[*1] アリのフェロモンを使ったえさ取り行動 http://www.alife.cs.is.nagoya-u.ac.jp/~reiji/aw/ants.html [Accessed 10 2 2015]

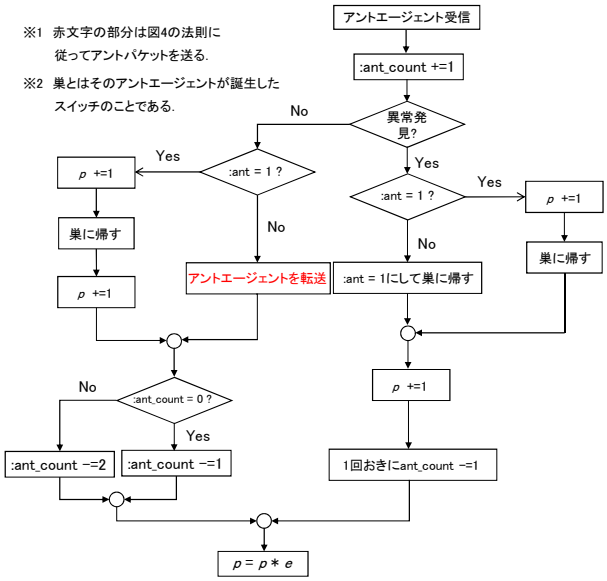

図 3　アントエージェントの制御フローチャート

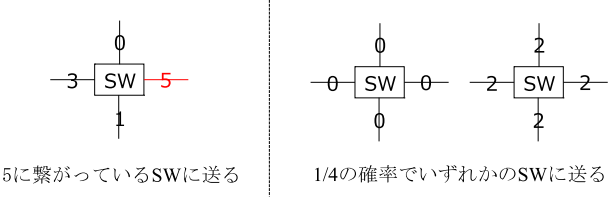

図 4　通常時の「0」アントエージェントの制御

トエージェントが生起したスイッチを指し，異常発見とは 3.3.1 節で示したトラフィック解析によるホストの監視結果を指す．図中の p は:pheromone_count の値を表しており，$p = p \times e$ は，上で述べたフェロモンカウントの蒸発を表している．また，赤色で示した「アントエージェントを転送」の部分は，図 4 に示す方法に従って次の送り先を決めている．すなわち，スイッチの状態が正常な場合は，アントエージェントは周囲のフェロモンカウントの値の割合に従って確率的に遷移先を決定する．また，周囲のフェロモンカウントの値が全て同じ場合は，完全にランダムで次の遷移先を決定する．

まず，スイッチはアントエージェントを受け取ると，コントローラに Packet_IN メッセージを送出し，自身の持つパラメータである:ant_count を 1 増やす．次に，コントローラは表 3 に示す単位時間あたりの各要素をランダムに選択する．選択した要素のその時点での値が事前に設定した閾値を超えていればスイッチの状態を異常，超えていなければ正常と判断し，次へ移る．

1. スイッチの状態が異常のとき

:ant= 0 のときは，アントエージェントの状態，即ち:ant の値を 1 にした後，アントの持つパラメータ:nest を目指して転送する．:ant= 1 のときは，アントパケットを受信したポートの経路の:pheromone_count を 1 増やす．その後，そのアントエージェントを巣に帰す，即ち:nest を目指して転送する．転送後は，:pheromone_count の値を 1 増やす．また，異常のあるスイッチにおいてアントエージェントの数が増加するように，:ant_count の値を 1 回おきに 1 減らしている．

2. スイッチの状態が正常のとき

:ant= 0 のとき，アントエージェントのパラメータは一切変えずに，図 4 の法則に従って決定した次の転送先に送る．:ant= 1 のときは，アントエージェントを受信したポートの経路の:pheromone_count を 1 増やす．その後，そのアントエージェントを巣に帰す，即ち:nest を目指して転送する．その後，スイッチの:pheromone_count の値を 1 増やす．さらにその後，アントエージェントを受け取ったスイッチの:ant_count の値によって:ant_count の減らし方が異なる．:ant_count= 0 のときは，:ant_count の値が変化しないように 1 減らす．:ant_count≠ 0 のときは，状態が正常であるのに:ant_count が 0 でない場合なので，アントエージェントを解散させるために:ant_count を 1 つ余分に減らす．その後，全てのスイッチの:pheromone_count の値にフェロモン蒸発係数 $e(e < 1)$ をかける．

以上が，スイッチがアントエージェントを処理する一連の流れである．

3.5　変化点検知による攻撃検知

Jereme らによる先行研究では，エージェントパケットの数・密度を人が監視し，最終的な判断を下すシステムであった．しかし，ネットワークの規模が大きくなり，監視対象の数が大きくなった場合の人による監視は困難である．そこで，我々は変化点検知アルゴリズムの一つである ChangeFinder[12] を用いて攻撃検知の自動化を導入した．ChangeFinder では，時系列データに対してオンラインで学習を行い，その時点での変化点スコアを算出する．我々の提案手法では，各々のスイッチのアントエージェントの数はネットワーク状況によって常に変化する．従って，アントエージェントの数の変化によって攻撃を検知することが可能である．攻撃が行われていない通常時では，アントエージェントの数に大きな変化はないが，ホストに対して何らかの攻撃が行われると，対象ホストを収容するスイッチにアントエージェントが集まる．この時のアントエージェントの数の変化を検知することで，攻撃検知を自動化することが可能である．提案手法では，アントエージェントの変化に対する変化点スコアを計算し，その変化点スコアが予め設定した閾値を超えることで，システム全体として最終的な検知アラートを上げる．

4.　実験

4.1　監視対象の閾値決定実験

まず，表 3 に示す監視対象の閾値を決定する実験を行った．ホストマシン内部に仮想マシンを起動し，その中で図 5

に示すネットワークを構築した．OpenFlow スイッチには Open vSwitch を用いた．まず，図 5 中のホスト 1 に Tcpreplay を用いて攻撃通信データおよび正常通信データの 2 種類のトラフィックデータを送信した．攻撃通信データには，ボットネット通信，nmap による垂直ポートスキャン通信，水平ポートスキャン通信の全部で 3 つの pcap ファイルを用いた．ボットネット通信としては CCCDATAset2008 の 2008/04/28 01:03:59〜2008/04/28 01:04:29 の部分を使用した．垂直ポートスキャンデータは，同一の IP アドレスに UDP プロトコルを用いてポートスキャンを行った際のトラフィックをキャプチャした pcap ファイルとした．水平ポートスキャンデータは，/24 のネットワークアドレスに対し，単一のポートを確認するポートスキャンを行った際のトラフィックをキャプチャした pcap ファイルとした．また，正常通信データは Web ページの更新を含む Web ブラウジングを行っている際のトラフィックをキャプチャした pcap ファイルとした．

これらの pcap データをそれぞれホスト 1 の仮想ネットワークインターフェースに送信し，1 台のコントローラを用いて提案手法を動作させた．この時，表 3 に示す監視対象の要素を，それぞれ単独で用いた際に集まったアントパケットの数の最大値を記録した．この時の結果を表 5〜9 に示す．ここで，表中の閾値はコントローラがスイッチの状態を異常だと判断する閾値である．正常通信に対する割合とは，正常通信に対して集まったアントパケットの最大値に対する，ポートスキャン（垂直），ポートスキャン（水平），ボットネット通信の時にそれぞれ集まったアントパケットの最大値の割合を表している．3 つの攻撃通信データそれぞれの割合を合計したものが，正常通信に対する割合の合計である．この時，割合の合計が最大になった時の閾値を，その監視要素の閾値として採用した．表 5〜9 より，各監視要素の閾値を表 10 の通りに決定した．表 5 において，過去 10 秒間のフレームサイズが 300[byte] 未満のパケットの割合は，閾値を 1 とすることで，垂直・水平ポートスキャン通信，ボットネット通信の全てには有効であるが，正常通信には反応しにくい要素となっている．しかし，DNS amp 攻撃など，フレームサイズが大きなパケットの割合が大半を占めるような攻撃の場合は，この要素の影響は小さくなり，表 6〜9 といった他の要素が必要となってくると考えられる．

また，提案する分散型コントローラでは，コントローラ同士で表 3 に示す要素の値をお互いに共有し，その相関係数を異常判定の要素として用いる．したがって，相関係数の閾値も決定する必要がある．まず，分散型コントローラを用いたネットワーク環境として，図 7 に示すネットワークを用意した．ここで，コントローラ 1 は SW1 及び SW2 を制御しており，コントローラ 2 は SW2 を制御している．また，コントローラ 1 及びコントローラ 2 は，互いに表 4

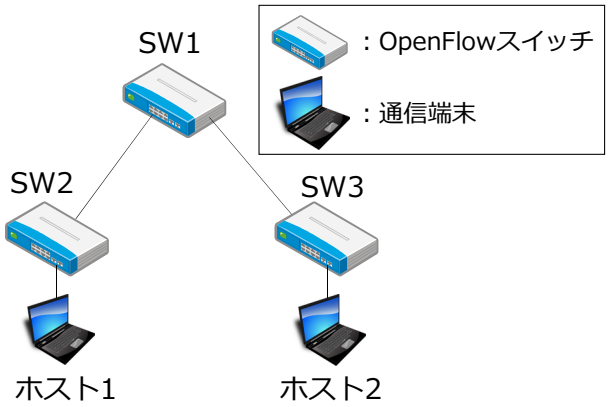

図 5　パラメータ決定実験で用いたネットワーク構成

に示す情報を交換している．境界スイッチ間のアントエージェントの移動情報は，アントエージェントが境界スイッチ間を跨いで移動する際に送信される．それ以外の監視要素の値及びトポロジ情報は 1 秒間隔で交換される．この時，表 7 に示すネットワークにおいて，ホスト 1 及び 2 のネットワークインターフェースに予めキャプチャしておいた正常通信データ 70 秒間送信した．同時に，速度を 10[パケット/秒] とした各々で異なるスローポートスキャンのトラフィックを SW2 及び SW3 のネットワークインターフェースに 30 秒間送信した．この時の，宛先ポート番号種類数における，各スイッチ間の相関係数を図 6 に示す．横軸は正常通信トラフィック及びポートスキャントラフィックを送信開始してから経過した時間 [秒] を表し，縦軸は各スイッチ間の宛先ポート番号種類数の相関係数を表している．全体を通して SW2-SW3 間の相関係数が高いが，特にポートスキャンが行われている最中の，最初の 30 秒間は共に相関係数がほぼ 1 となっている．また，ポートスキャンが終了した 30 秒以後に関して，SW2-SW3 間の相関係数は依然高い値であるが，それ以外の相関係数は低くなっている．これは，ポートスキャン終了後の宛先ポート番号種類数の減少具合が SW2-SW3 間の相関係数を高くしているのだと考えられる．この結果より，相関係数によるスイッチ状態の異常判定閾値は，SW2-SW3 以外の相関係数で最大値である 0.9 とした．

4.2　情報共有による分散型コントローラ実験

次に，閾値決定実験にて決定した閾値を用いて，情報交換による分散型コントローラ実験によるネットワーク攻撃検知実験を行った．まず，図 7 に示すネットワークを用意した．ホスト 1 及びホスト 2 のネットワークインターフェースから，リアルタイムの正常通信トラフィックを送信し，その間に SW2 及び SW3 のネットワークインターフェースに水平スローポートスキャンを 30 秒間行った．この時の速度は 10[パケット/秒] とした．コントローラ間では，表 4 に示す情報を共有し，その情報交換頻度は閾値決定実験の

表 5　過去 10 秒間のフレームサイズが 300[byte] 未満のパケットの割合

閾値	正常通信	ポートスキャン（垂直）	ポートスキャン（水平）	ボットネット通信	正常通信に対する割合の合計
0.5	319	358	359	341	3.32
0.8	269	382	390	337	4.12
0.9	63	374	377	229	15.56
1	60	382	366	191	15.65

表 6　過去 10 秒間の宛先 IP 種類数

閾値	正常通信	ポートスキャン（垂直）	ポートスキャン（水平）	ボットネット通信	正常通信に対する割合の合計
0	292	337	398	362	3.76
5	147	1	395	322	4.88
10	116	1	336	291	5.41
15	1	1	343	308	652

表 7　過去 10 秒間の送信元ポート番号種類数

閾値	正常通信	ポートスキャン（垂直）	ポートスキャン（水平）	ボットネット通信	正常通信に対する割合の合計
0	328	386	336	383	3.37
5	217	1	1	377	1.75
10	152	1	1	348	2.30
20	128	1	1	307	2.41
50	1	1	1	274	276

表 8　過去 10 秒間の宛先ポート番号種類数

閾値	正常通信	ポートスキャン（垂直）	ポートスキャン（水平）	ボットネット通信	正常通信に対する割合の合計
0	329	372	231	393	3.03
10	183	354	1	118	2.58
40	1	351	1	1	363
50	1	361	1	1	363
500	1	1	1	1	3

表 9　過去 10 秒間のフレームサイズの分散

閾値	正常通信	ポートスキャン（垂直）	ポートスキャン（水平）	ボットネット通信	正常通信に対する割合の合計
0	307	1	1	328	1.07
40000	127	1	1	143	1.14
80000	22	1	1	130	6
100000	1	1	1	152	154

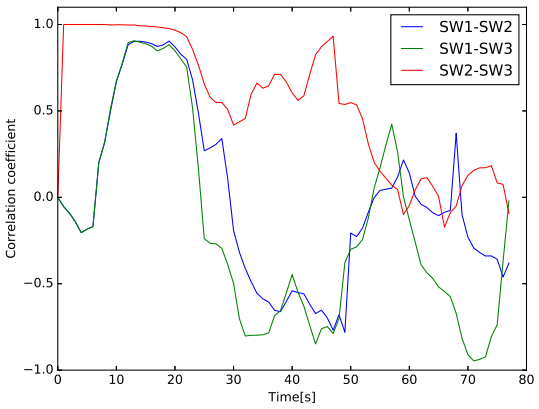

図 6　スローポートスキャン時の各スイッチ間の相関係数

時と同様にした．スイッチの異常判定には，表 3 に示す監視要素の閾値判定だけでなく，各要素の値のスイッチ間に

表 10　評価実験で用いた各パラメータの値

フレームサイズが 300[byte] 未満のパケットの割合	1
宛先 IP アドレス種類数	15
送信元ポート番号種類数	50
宛先ポート番号種類数	50
フレームサイズの分散	100000

おける相関係数も用いた．この時，それぞれ閾値決定実験で決定した通り，監視要素の閾値は表 10 の通りとし，相関係数の閾値は 0.9 とした．また，ここでは:life_count = 10 とし，3.1 節の式において $\varepsilon = 0.1$，$T = 600$ としてフェロモン蒸発係数 $e = 0.98$ とした．変化点スコアは，従来の手法と同様に 40 を超えた時点で検知アラートを上げるようにした．この時，このポートスキャンを検知できるかどうかの実験を，従来の完全分散型コントローラを用いる手法と，今回のコントローラ情報共有を用いる新たな提案手法の 2 つを用いて行った．

まず従来の完全分散型で行った際の結果を図 8 に示す．横軸は正常通信トラフィック及びポートスキャントラフィックの送信を開始してから経過した時間 [秒] を表し，縦軸は各スイッチに集まったアントエージェントの数:ant_count を表している．水平スローポートスキャンとしてスキャンパケットが分散して SW2 及び SW3 に来ているので，単一のスイッチしか見ない従来の完全分散型コントローラでは，アントエージェントが最大でも 1 しか集まらないという結果になった．従って，今回の実験で用いた水平スローポートスキャンは，従来の完全分散型コントローラでは検知することが困難である．

次に，今回の新たな提案手法である，情報共有による分散型コントローラを用いた際の結果を図 9 に示す．完全分散型の時とは異なり，アントエージェントが集まっていることが分かる．しかしながら，SW2 にはポートスキャンが行われたタイミングと合わせてアントエージェントが集まっているものの，SW3 には殆どアントエージェントが集まらなかった．この時の SW3 における相関係数がほぼ全てにおいて閾値として設定した 0.9 を超えていなかった．コントローラ間で交換する情報の一つとして監視要素の値があり，この値から自ネットワーク内のスイッチで観測されたトラフィックと他ネットワーク内のスイッチで観測されたトラフィックの相関係数を算出している．しかし，情報の交換はネットワーク越しに行われるため，他ネットワークの監視要素の値を常にリアルタイムで受け取れるとは限らず，情報の到達に遅延が生じた可能性がある．この遅延により，SW3 ではポートスキャンによるトラフィックの変化の相関が取れず，アントエージェントが集まらなかったと考えられる．また，50 秒を過ぎた辺りから，SW1 のアントエージェントも集まり始めている．ポートスキャンによるトラフィックの変化と，正常通信によるトラフィックの変化が同期したことで，正常通信に対しても一部アントエージェントが集まったと考えられる．次に，各スイッチのアントエージェントの数:ant_count の値に対する変化点スコアを図 10〜12 に示す．図 10 の SW1 においては，50 秒まではスコアは 10 程度で維持しているが，正常通信トラフィックがポートスキャントラフィックと同期してしまった 50 秒を過ぎた辺りから，アントエージェントの変化と共に変化点スコアも最大 80 まで上昇している．図 11 の SW2 に関しては，ポートスキャンに対してアントエージェントが集まり始めた 8 秒付近から変化点スコアが速やかに上昇し，40 を超える 8 秒付近で検知アラートを出した．図の横軸の原点がポートスキャン開始であることから，検知までに約 8 秒を要した．従来の完全分散型コントローラによる垂直ポートスキャン検知 [7] よりも検知が約 3 秒遅くなっているが，これはネットワーク越しに情報を送受信し，その値を用いて全体の相関を算出することにより生じる遅延が原因だと考えられる．図 12 の SW3 に関して，前述した様に，ネットワーク越しの情報共有の遅延により相関係数が上昇せず，結果アントエージェントが集まらなかったので，変化点スコアも 10 前後でほぼ変化せず，SW3 においては検知アラートが上がらないという結果になった．

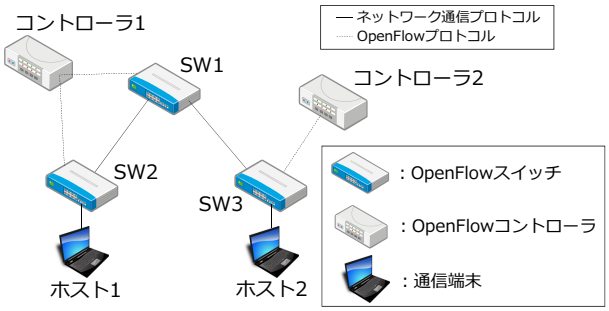

図 7　分散型コントローラ実験で用いたネットワーク構成

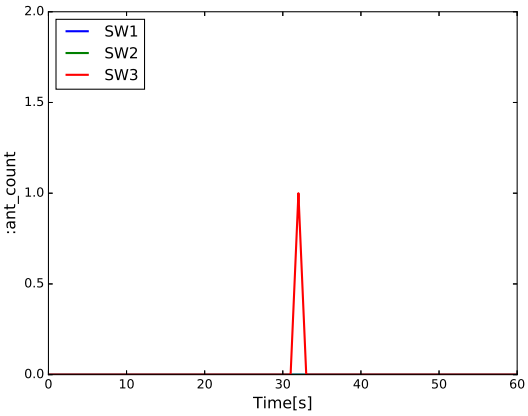

図 8　水平スローポートスキャン（完全分散型コントローラ）

5. 結論

コントローラ同士で情報を共有することで，水平ポートスキャンなどの完全分散型コントローラ方式では検知できないようなネットワーク攻撃に対して有効に検知できることを示した．コントローラ間でネットワーク越しにトラフィック監視要素の値を交換し，自ネットワークと他ネットワークでの各監視要素の相関係数を計算することで，完全分散型より広範囲なネットワークの変化を見ることが可能となった．しかしながら，ネットワーク越しに共有することで生じる遅延により，ネットワーク攻撃に対しトラフィック変化の相関が取れずアントエージェントが集まらないスイッチも存在した．遅延を考慮した相関を算出する計算方法を検討する必要がある．また，今回はスイッチ 2 台と 1 台に対してそれぞれコントローラを 1 台ずつ使用したが，実験ネットワークの規模を大きくした上で，使用するコントローラの数とスイッチの数の最適な比率を求める必要がある．

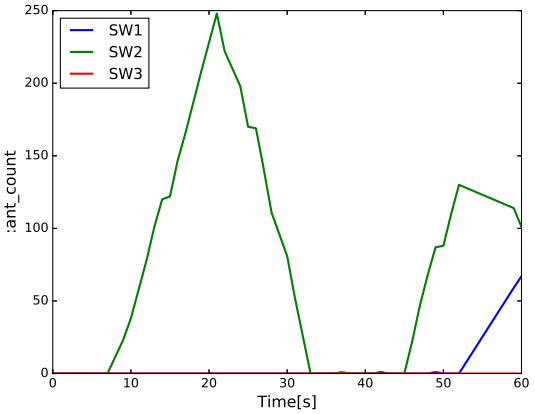

図 9 水平スローポートスキャン（情報共有分散型コントローラ）

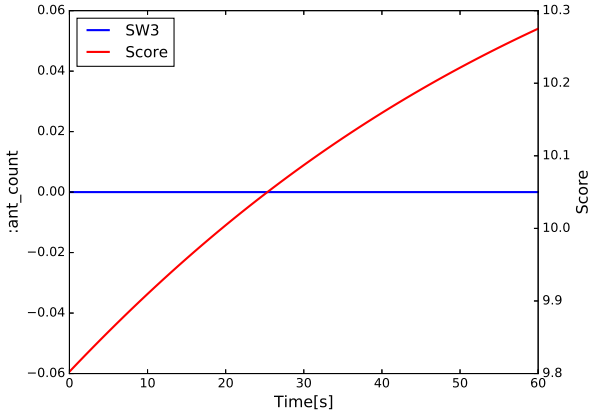

図 12 水平スローポートスキャン時の変化点スコア（SW3）

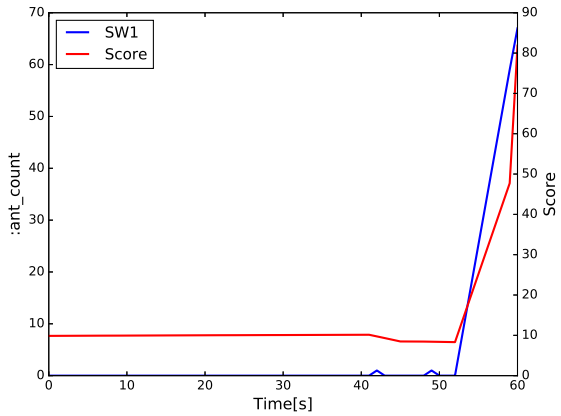

図 10 水平スローポートスキャン時の変化点スコア（SW1）

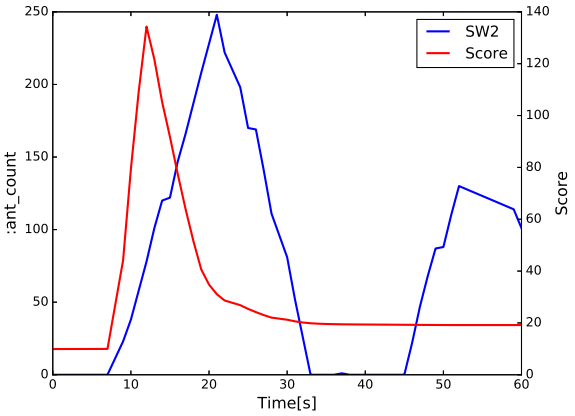

図 11 水平スローポートスキャン時の変化点スコア（SW2）

参考文献

[1] Haack, J. N., Fink, G. A., Maiden, W. M., McKinnon, A. D., Templeton, S. J. and Fulp, E. W.: Ant-Based Cyber Security, *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*, pp. 918–926 (online), DOI: 10.1109/ITNG.2011.159 (2011).

[2] Fink, G. A., Haack, J. N., McKinnon, A. D. and Fulp, E. W.: Defense on the Move: Ant-Based Cyber Defense, *IEEE Security & Privacy*, Vol. 12, No. 2, pp. 36–43 (online), DOI: http://doi.ieeecomputersociety.org/10.1109/MSP.2014.21 (2014).

[3] Fink, G. A. and McKinnon, A. D.: Effects of Network Delays on Swarming in a Multi-agent Security System, *Proceedings of the 1st International Workshop on Agents and CyberSecurity*, ACySE '14, New York, NY, USA, ACM, pp. 11:1–11:8 (online), DOI: 10.1145/2602945.2602955 (2014).

[4] 宮崎 亮輔, 川本 淳平, 松本 晋一, 櫻井 幸一: 攻撃検知のための端末非依存型システムを実現する OpenFlow コントローラの実装と評価, 火の国情報シンポジウム (2015).

[5] 宮崎 亮輔, 川本 淳平, 松本 晋一, 櫻井 幸一: ネットワーク攻撃に対する端末非依存型検知方式の OpenFlow コントローラ上への実装と評価, DICOMO2015 (2015).

[6] 宮崎 亮輔, 川本 淳平, 松本 晋一, 櫻井 幸一: OpenFlow を用いた端末非依存型検知方式における負荷分散法, JCEEE (2015).

[7] 宮崎 亮輔, 川本 淳平, 松本 晋一, 櫻井 幸一: 実ネットワーク環境における OpenFlow を用いた端末非依存型検知方式の評価, SCIS2016 (2016).

[8] Chen, X.-F. and Yu, S.-Z.: CIPA: A collaborative intrusion prevention architecture for programmable network and {SDN}, *Computers & Security*, Vol. 58, pp. 1 – 19 (online), DOI: http://dx.doi.org/10.1016/j.cose.2015.11.008 (2016).

[9] Jajodia, S., Ghosh, A. K., Swarup, V., Wang, C. and Wang, X. S.: *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, Springer Publishing Company, Incorporated, 1st edition (2011).

[10] Jajodia, S., Ghosh, A. K., Subrahmanian, V., Swarup, V., Wang, C. and Wang, X. S.: Moving Target Defense II, *Application of game Theory and Adversarial Modeling. Series: Advances in Information Security*, Vol. 100, p. 203 (2013).

[11] Manadhata, P. K. and Wing, J. M.: *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, chapter A Formal Model for a System's Attack Surface, pp. 1–28 (online), DOI: 10.1007/978-1-4614-0977-9_1, Springer New York (2011).

[12] Takeuchi, J. and Yamanishi, K.: A unifying framework for detecting outliers and change points from time series, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 18, No. 4, pp. 482–492 (online), DOI: 10.1109/TKDE.2006.1599387 (2006).