社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

DarkFlow検出によるリアルタイム・インターネット脅威検出システム

下田　晃弘† 　森　達哉†† 　後藤　滋樹†

† 早稲田大学大学院 基幹理工学研究科 情報理工学専攻 〒 169-8555 東京都新宿区大久保 3-4-1
†† NTT サービスインテグレーション基盤研究所 〒 180-8585 東京都武蔵野市緑町 3-9-11
E-mail: †{shimo,goto}@goto.info.waseda.ac.jp, ††mori.tatsuya@lab.ntt.co.jp

あらまし　インターネットはマルウェアやボットネットによる脅威に常時晒されている．スキャン行為等によって送信されるパケットは攻撃の前兆となり得る．これを早期の検出し警告を発信する必要がある．このようなパケットを検出するために，受信専用のセンサ・サーバを配置する方法や, ゲートウェイにおいて組織内で未使用の IP アドレスを検出して, センサを動的に展開する方法が提案されている．しかし，IP アドレス単位の監視は IPv4 アドレス枯渇の状況下では規模の拡張が難しい．さらに IPv6 環境ではアドレス空間が広いため IP アドレス単位の監視には限界が生じる．本論文ではインターネット上のマルウェアやボットネットによる攻撃をフロー単位で検出する方法を提案する．具体的には仮想的なハニーポットおよびセンサを設置する．この仮想ハニーポットと仮想センサは通常の通信に対しては動作しない．動作の対象を大量のトラヒックの中から選択するところに本研究の特徴がある．本研究では TCP フローのフラグと遅延時間を分析することによって, 応答が返らないフロー (DarkFlow) に着目する．

キーワード　脅威検出, フロー, ボットネット, マルウェア, ダークフロー

Internet Threat Detaction based on Locating DarkFlow

Akihiro SHIMODA†, Tatsuya MORI††, and Shigeki GOTO†

†
††
E-mail: †{shimo,goto}@goto.info.waseda.ac.jp, ††mori.tatsuya@lab.ntt.co.jp

Abstract　The internet is exposed to malware or botnet attacks consistently. Network administrators should send out early warning of attacks typically caused by network scanning, for alerting network users. Earlier researches attempt to distribute passive sensors or utilizing unused-IP addresses inside the gateway for threat monitoring. However, considering the situation of the IPv4 address pool exhaustion, IP address based monitoring will be hardly expand the scale in the near future. By the same token, there is a limitation for monitoring IPv6 address space. In this paper, we propose a flow-based malicious attack detection for malware or botnets. In addition, we adopt virtual hoenypot and sensors in the working network not to effect the existing services. Our method is effectively select malicious flows from the large volume of backbone traffic. We focus on TCP flows with flags and delayed responses. We locate non-responding flows (Darkflow) and integrate them with sensors or honeypots.

Key words　Internet threat, network flow, malware, botnet, darkflow

1. はじめに

インターネット上にボットネットやマルウェアによる悪意のある通信が蔓延している. [1] によるとボットネットによる経済的被害は 2006 年度に 133 億ドルに上る．大きな被害を生むインターネット上の攻撃を早期に検知するため，各地で定点観測システムが運用されている [2]〜[5]．定点観測システムは受信専用に設定されたパケットログ収集サーバ (センサ) をインターネット上に分散配置することにより, スキャンパケット，DDoS Backscatter パケットといった攻撃の兆候を検出する．IP アドレスの割り当て方式には, 単一のグローバル IP アドレスをセンサに割り当てる方法と, 静的ルーティングの変更によりサブネット単位で割り当てる方法 [6] がある．

センサは外部から到達するパケットに対して一切の応答を返さない．そのためセンサに到来するパケットは実在しない宛先に送信されていることになる．センサのログには正常な通信が

紛れ込むことが無い．このため誤検知が発生しないという利点がある．一方で, センサに対して未使用の IP アドレスやホストを割り当てる必要があるから, センサの展開にはネットワーク・リソースを必要とする．特に IPv4 アドレス枯渇 [7] の環境下ではセンサ監視のために新たに大量のリソースを確保することは難しい．

センサが収集するパケットは以下の 3 種類に分類される．

- 設定の不備によるパケット
- DDoS Backscatter
- ホストるいはポートスキャン

IUCC/IDC [8] の観測によると, センサで観測されるパケットの 92%がホストポートスキャン，5%が DDoS Backscatter, 2%が設定ミス，1%がその他と報告されている．

DDoS の Backscatter は送信元が本来のホストとは異なる IP アドレスに偽装されて DDoS 攻撃が行われた場合に，DDoS 攻撃とは無関係なホストに対して応答パケットが送信されるパケットである．TCP の場合，センサでは syn/ack，rst パケットとして観測される．ホスト・ポートスキャンは，マルウェアやボットネットがホストの存在やポートの空き状況を確認するために送信するパケットである．スキャンパケットに応答が返った場合には，セッションを確立して直接に脆弱性を突くペイロードを送信する場合や，tcp/rst の送信により切断する場合がある．

センサではスキャン行為によるパケットを検出することができる．しかしセンサは攻撃元ホストとセッションを確立しないパッシブな観測方式であるため，セッション確立後に得られる攻撃の通信パターンやセッションの詳細な情報は入手できない．一方で, 攻撃元に対して意図的に応答を返して，攻撃のより詳細な情報を入手できるハニーポットによる観測方法がある [9]．ハニーポットには 2 種類の方式が存在する．脆弱性を持つ OS やサービスを稼働させる高対話型ハニーポットと, 脆弱性をエミュレートとした疑似的なサービスを稼働させる低対話型ハニーポット [10] である．

ハニーポットの利用により，感染活動や感染後の挙動に関する詳細な情報を得ることができる．ただし, ハニーポットはセンサと異なり運用に細心の注意を必要とする．例えばハニーポットによる通信が上層の IDS に検知されて警告を受けないようにする配慮や, 他のホストに意図せずに感染を広めないような工夫が必要である．

2. 関 連 研 究

インターネット上の攻撃を大規模かつ効率的に収集するために，センサを IP アドレス単位ではなく，IP サブネット単位で監視する方法がある．Internet Motion Sensor [6] は/8 や /16 等の複数の分散した IP アドレス空間を確保して，静的ルーティングの設定を変更することにより少数のセンサ台数で広範囲のアドレス空間の監視を実現している．この方式のメリットは，省リソースで広範囲のサブネットを監視できる点である．一方で広域の IP アドレスのリソースを観測用途で占有することは一部の機関を除いては困難になりつつある．

サブネット単位による監視をハニーポットに応用した Potemkin Virtual Honeyfarm [11] がある．このシステムは特定の監視サブネットに対して送信されたすべての TCP syn パケットを，一カ所のゲートウェイに転送する．このゲートウェイは独自に開発されており，TCP syn のセッションを保持しつつ負荷分散を考慮してバックエンドの高対話型ハニーポット群に転送して処理する．一般に高対話型ハニーポットは一度攻撃を受けると，OS が感染や破壊によって汚染されてしまうため, 次の攻撃を観測するためには OS の状態を復元する処理が必要となる．このシステムは仮想サーバを用いてハニーポットの動作が管理されており，接続要求の転送からマルウェア感染後の OS の復元処理までをすべて自動化している．このシステムは未使用のサブネット全体を監視対象としており，稼働中のネットワークに対して適用することは述べられていない．

稼働中のネットワークにおける未使用 IP アドレス (Dark IP address) を動的に検出してセンサを展開する方法が [12] で提案されている．稼働中のネットワークとは企業や大学などの組織に割り当てられて，恒常的に利用されているネットワークである．この提案は組織内に未使用の IP アドレスが数多く点在することに着目し，独自の学習アルゴリズムによりそれらを検出する．具体的には組織内から一定期間応答を返さない IP アドレスをセンサと見なす．実験によると，/16 のサブネットから 2 万個以上の未使用の IP アドレスを検出し，センサと同等のログの収集を実現している．しかし，組織内で突然発信を始めた IP アドレスに対しては誤検出の可能性が指摘されている．

稼働中のネットワークに含まれる未使用 IP アドレス群に対してハニーポットを展開する DarkPots [13] がある．この提案は未使用 IP アドレスの誤検出の可能性を限りなく小さくするために, [12] の方式に加えて，ゲートウェイのファイアウォールでブロックされる IP アドレスの情報も利用して確実に未使用な IP アドレスを絞り込んでいる．その上で，検出した IP アドレスに対する接続要求を低対話型ハニーポットに転送し，攻撃元に対して代理で応答を返す仕組みを実現している．DarkPots は稼働中のネットワークに点在する未使用の IP アドレスをインターネット脅威検出のために効率的に活用できる利点がある．一方で，DarkPots は未使用 IP アドレスのリストを常に保持して，ゲートウェイに到達するすべてのパケットに対して宛先 IP アドレスの比較を行う．このため，/8 などの広大なサブネットを持つ組織や，IPv6 環境下においては IP アドレスの比較処理がボトルネックとなり，十分なパフォーマンスを発揮することが難しい．

3. 提 案 手 法

本研究は IP アドレス空間の大きさに依存しない不正パケットの検出方法を提案する．さらにセンサ、ハニーポットと連携するシステムを開発した．本章では最初にフローの説明と DarkFlow の定義を行い，それらを検出する仕組みと評価について述べる．後半は DarkFlow を検出するシステムの構成とセンサ，ハニーポットとの連携について述べる．

3.1 応答の返らないフロー

ネットワーク・**フロー**は以下の 4 つのパラメータで表される．

- 送信元 IP アドレス
- 送信元ポート番号
- 宛先 IP アドレス
- 宛先ポート番号

本論文では TCP 通信の syn を送信したホストをクライアント (c)，一方をサーバ (s) と定義し，クライアントからサーバに対するフローを順方向フロー (`flow_fwd`)，送信元と宛先が逆のフローを逆方向フロー (`flow_rev`) とする．TCP のセッションは順方向フローと逆方向フローの合計 2 つのフローで構成される．このとき順方向フローと逆方向フローは IP アドレスとポート番号を組み合わせて次のように表現できる。

`flow_fwd` =
$$flow(c.addr, c.port, s.addr, s.port)$$
`flow_rev` =
$$flow(s.addr, s.port, c.addr, c.port)$$

本研究が定義する DarkFlow は，クライアントがサーバに syn パケットを送信後に，syn/ack の応答が返らない順方向フローのことである．syn パケットのみでも 1 フローとしてカウントされる．しかし，1 パケットでは 3-way handshake が成立しないため，TCP 上のセッションは成立しない．このようなフローは syn パケットの送信先 IP アドレスが以下の状態の時に起こりうる．

- ネットワーク側の要因

（1） 宛先経路が存在しない
（2） IP アドレスが管理上未割り当て
（3） 経路上のファイアウォールにより通信が禁止
（4） 輻輳によりパケットが損失

- ホスト側の要因

（1） ホストの電源が入っていない
（2） ファイアウォールにより通信が禁止
（3） サービスの不具合や過負荷による遅延

syn/ack が返らない場合にはクライアントは再送を試みる．その際，TCP プロトコルにより再送されるパケットは送信元ポート番号が変化しないため，初回の syn パケットと同一のフローに属する．一方でアプリケーションによって再送されたパケットは，ポート番号が変化するために初回の syn パケットとは異なるフローに属する．この両者の違いは重複した syn パケットの理由を区別するために重要である．

3.2 DarkFlow 検出の仕組み

本研究では syn パケットに対して syn/ack 応答の返らないフローを DarkFlow と定義する．このようなフローを検出するために，本研究では syn に対して syn/ack の応答が一定時間返らないフローを DarkFlow と見なす方式を提案する．図 1 の (a) は一般的な TCP における通信例を示す．通常はクライアントからの syn パケットに対してサーバのポートが listen 状態であれば syn/ack を返す．ポートが listen 状態でなければ通常は rst を返すが，rst を有無はファイアウォールや OS の設定

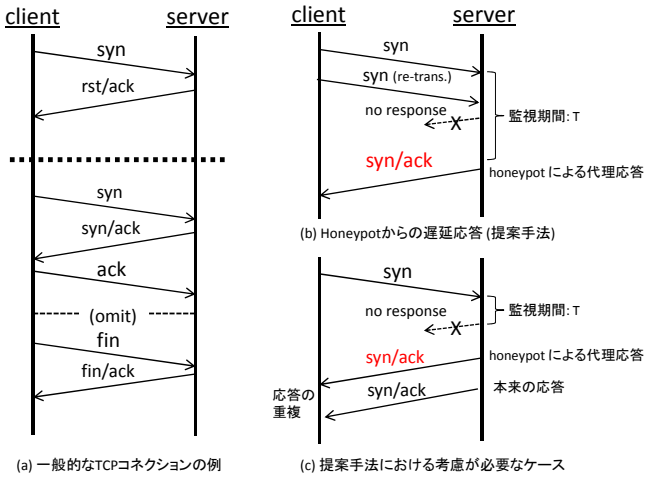

図 1　TCP フラグの遷移例

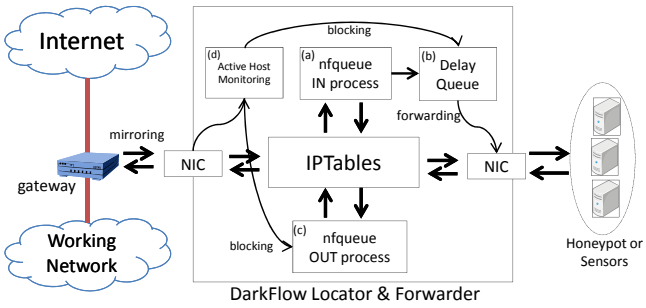

図 2　DarkFlow 検出および転送システム

により異なる．(b) は DarkFlow を検出した後で，後述するシステムを用いてハニーポットが代理で応答を返す場合の通信例である．応答が返らないフローを検出するために，サーバから syn/ack もしくは rst の応答の有無を時間パラメータ (T) の期間監視する．T の時間内にサーバから応答が無かった場合は，syn パケットをセンサもしくはハニーポットに転送する．転送先がハニーポットの場合は，送信元を偽装した上で syn/ack をクライアントに応答してセッションを確立する．T の時間はパラメータであり，短すぎると syn/ack の応答が遅延した場合には DarkFlow の誤検出が発生し，逆に長すぎるとクライアントがタイムアウトする可能性がある．この点の評価は実験で述べる．

DarkFlow の検出において最も考慮しなければならない課題として，図 1 の (c) に示すように，T 時間よりも遅れて本来のサーバから syn/ack の応答があった場合に，ハニーポットによる syn/ack の応答と重複することで TCP の通信を乱す可能性がある．この問題は後述の提案システムに実装したハニーポットにおいて通信を即時に遮断の仕組みと，実際のホストにおける挙動を分析することで解決できることを実証する．

3.3 DarkFlow 検出システムとセンサ・ハニーポットとの連携

図 2 は DarkFlow 検出システムの実装を示す．本システムは組織のゲートウェイ前後のバックボーン・ネットワークに設置することを想定している．これは TCP のセッションを正確に追跡するために非対称経路が無いことを保証するためである．

システムへの入力はバックボーンを流れるすべての通信であり，本実験ではバックボーンのスイッチから DarkFlow 検出システムに対して 10Gbps の回線でミラーリングを行っている．

システムは NIC を 2 つ備えた一般的な Linux サーバである．CPU は Xeon X3430 (2.4Ghz)，メモリは 4GB を搭載し，OS は Ubuntu Server 11.04 x86_64 を利用する．システムの特徴は netfilter/IPtables [14] の機能を活用することで，syn パケット以外はカーネル空間で処理して高速な転送を実現している．図 2 の (a) と (c) は netfilter の機能である nfqueue を用いて実装している．nfqueue は IPtables のチェーンの任意の位置からユーザプロセスに対してパケットを転送することができる．本システムは大きく 4 つのプロセスで構成される．図 2 の (a) は (b) と併せて DarkFlow の検出およびパケットの転送の可否を判断する処理を担う．具体的には，入力が syn パケットの場合は (b) の遅延キューに転送する．入力がフローテーブルに登録されている場合は IPtables 経由でハニーポット，サーバに転送する．それ以外のパケットは破棄するという処理を行う．

(b) の遅延キューは DarkFlow 検出するために，syn パケットをセンサ・ハニーポットに転送する前に一時的に保存しておく領域である．syn パケット 1 つに要する記憶容量は最小の場合で IP ヘッダ 20 バイト，TCP ヘッダ 24 バイトの計 48 バイトと小さいため，多くの接続要求を蓄えることができる．遅延キューの各パケットは入力から T 時間を過ぎると外部のサーバに転送する．同時に次回のパケットから IPtables 経由で転送できるようにするためにフローの宛先 IP アドレスをルーティングテーブルに登録し，フローテーブルにも情報を登録する．遅延キューはパケットを保持する T 時間の間に (d) から syn とは逆向きのフローが検出された場合は，直ちにキューから当該のパケットを削除する．

(c) はハニーポットからインターネット方向のパケットを処理するプロセスである．通常はすべてのパケットをインターネット側に転送する．しかし，万が一転送しようとするパケットの逆向きのフローが (d) によって検出された場合には，非常に短い時間内にハニーポットからの転送を中断することができる．

(d) は syn とは逆向きのフローが存在するか否かをチェックするためのプロセスである．(d) は NIC に対して libpcap ライブラリ [15] を用いてインターフェイスを直接監視する．これは IPtables のチェーンには入らないパケットを捕らえるためである．(d) は入力パケットに対してフローテーブルと照合して，逆向きのフローが存在した場合には (b) と (c) にイベントを送信する．

3.4 センサ・ハニーポット側の実装

図 3 は DarkFlow 検出システムのバックエンドで動作するセンサ・ハニーポットの実装である．センサとして動作させる場合は，NIC から受信したパケットを直接 IPtables や tcpdump 等で監視するようにして，IPtables で外向きの応答を返さないようにする．一方でハニーポットとして動作させる場合は，図 3 の (a) において，パケットをプロセスに転送する直前に，フローの宛先 IP アドレスを持つサブインターフェイスを生成して，ハニーポット・プロセスが当該パケットを受け取れるようにする．ハニーポット・プロセスは Nepenthes [10] のように Linux 上でプロセスとして動作するものを想定している．プログラムの socket の bind 先を ANY もしくは 0.0.0.0 に設定できれば，ソフトウェアの変更無しに本提案を実装することができる．

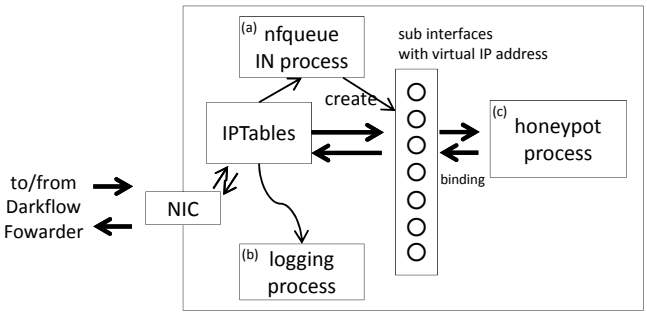

図 3　センサ・ハニーポットの実装

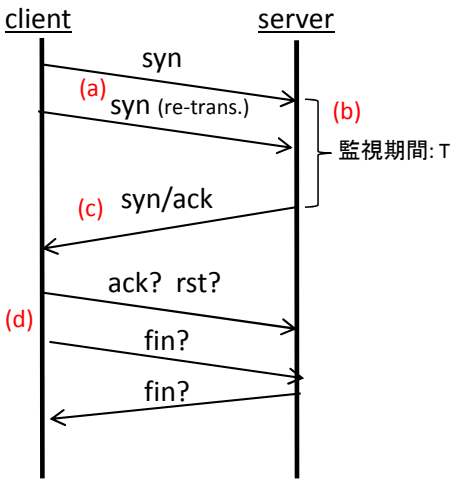

図 4　DarkFlow の検証項目

4. 評価実験

本章では最初に DarkFlow を検出する際に障害となる事象を洗い出し，監視期間 (T) の最適値を求めるための事前検証について述べる．次にシステムの性能評価を行い，最後にケーススタディとしてシステムを動作することによって得られたログを示す．

4.1 DarkFlow の評価

DarkFlow は syn/ack パケットを監視期間 (T) 待ち，応答が無かった場合に不正なフローと見なす．この方式が有効であることを示すため，図 4 の (a)〜(c) の項目について検証を実施した．DarkFlow の評価で用いている測定データは，大学のゲートウェイにおいて 2 日間キャプチャを行い，データを保存した上で静的に解析を行った結果である．

4.1.1 (a) syn の再送

クライアントが送信する syn パケットに対して，ハニーポットによる syn/ack の代理応答が遅すぎる場合，タイムアウトにより 3-way handshake が成立しない可能性がある．syn の再送回数は，Windows の場合はレジストリの Tcp-

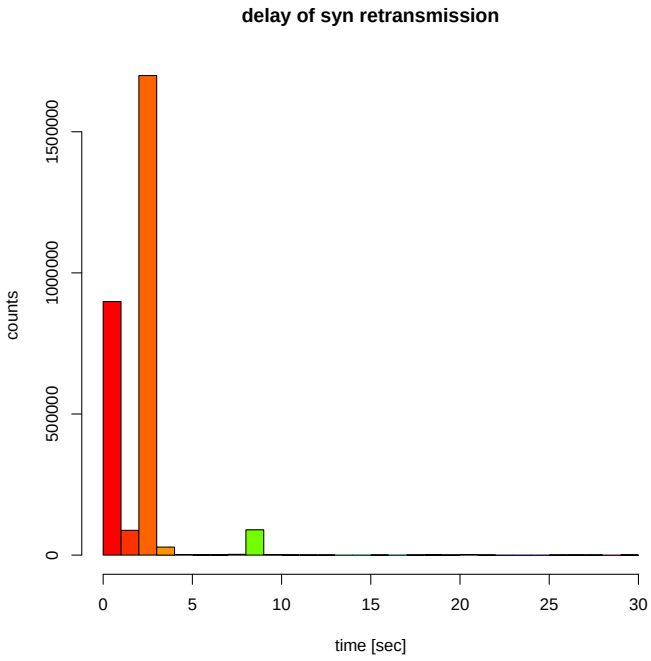

図 5　syn パケット再送時間

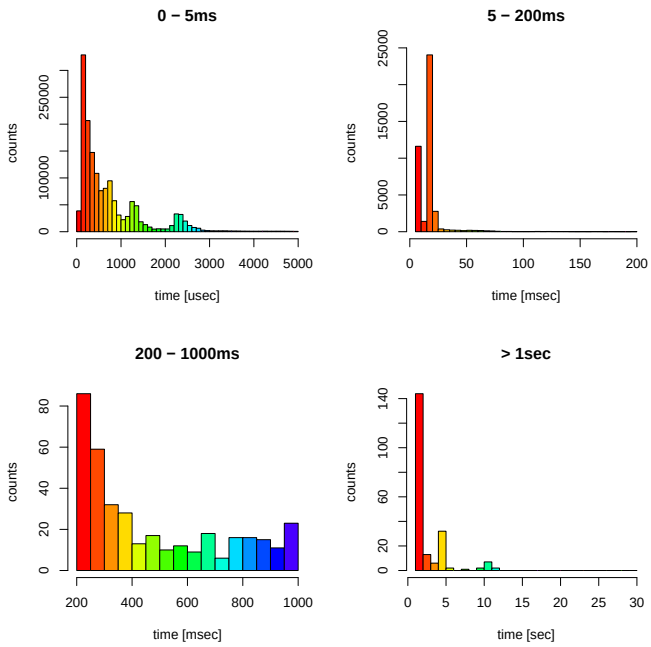

図 6　syn/ack の遅延時間

MaxConnectRestransmissions (デフォルト値: 2) [16]， Linux は/proc/sys/net/ipv4/tcp_syn_retries (デフォルト値: 5) である．Linux (kernel 2.6.38-8) の TCP socket で計測したところ，初回の syn 送信から 3，6，12，24，48 秒の計 5 回 syn の再送を行い，96 秒でタイムアウトするという結果を得た．しかし，アプリケーション側で再送回数を変更することも可能である．一例として，前述と同じ Linux(kernel 2.6.38-8) 上で動作するブラウザ Firefox で計測したところ，初回の syn から 3 秒，6 秒の計 2 回再送をして，12 秒後のタイムアウトする．これはアプリケーション側が OS の syn 再送回数の初期値に依存せず，独自に再送回数とタイムアウト時間を制御できることを意味する．マルウェアやボットネットのプログラムにおけるタイムアウト時間も実装に依存すると考えられる．

実際に稼働中のネットワークで未使用 IP アドレス群に対する初回の syn から最後の再送による syn までの時間とフロー数の分布を測定した結果を図 5 に示す．多くのフローは再送をせず，3 秒， 6 秒に多くのフローが分布していることが分かる．これらの結果は監視期間 T 定める上での指標となる．

4. 1. 2　(b) syn/ack の遅延時間

稼働中のネットワークにおいて syn/ack の遅延時間を調べるために，syn パケットのフローと逆方向の syn/ack が観測されるまでの時間を測定した．その結果を図 6 に示す．結果を見ると syn/ack の総数に対して，74.0%が 1msec 以内，98.1%が 10msec 以内，99.81%が 30msec 以内の応答となっている．これは観測地点がゲートウェイであり，組織内部のホストが回線的に近い位置に存在することに起因する．一方で syn から 5 sec 以上の応答を要した syn/ack パケットは全体の 0.0029%である．これは syn/ack の合計 1，579，832 パケットに対して 46 パケットに相当する．syn/ack の応答が遅れる要因としては，サーバの過負荷やサービスの不具合，ネットワーク・カード等の故障等が考えられる．

TCP flag	count
RST from client	19
RST from server	4
FIN	0
No-response after syn/ack	23

表 1　syn/ack 遅延後のセッション・トレース結果

4. 1. 3　(c) syn/ack が遅延したフロー

前節の検証において syn/ack が 5 秒以上遅延した場合のセッションを追跡したところ，表 1 の結果を得た．サーバ側のユニーク IP アドレス数は 4 である．syn/ack が遅延した後，FIN によって TCP セッションが正常にまで至る例は無かった．syn/ack の後，クライアントからの RST，もしくは応答が無いことによって 3-way handshake が未成立のまま終了している．サーバの 1 つは syn/ack を応答した後に ack が無いために，rst を送信するケースも存在した．

別方向の検証として，ハニーポットが代理で応答をして通信をしている最中に，本来のサーバから syn/ack が返された場合の挙動を調査した．検証は hping3 [17] を用いた．検証手順としては，クライアント，ハニーポット間で数分を要するファイルのダウンロードをしている最中に，検証用の別のホストからクライアントに対して，ハニーポットが応答するパケットと同一の送信元 IP アドレスと初期 TCP シーケンス番号を持つように偽装した syn/ack パケットを送信することにより挙動を分析した．その結果，hping3 により偽装した syn/ack パケットを数回送信しても，すでに確立したクライアントとハニーポット間のセッションが中断されることなく，ファイルのダウンロードは正常に終了した．この結果からハニーポットが通信している最中に syn/ack が遅延してクライアントに到達した場合は，すでに確立したセッションが優先されることを確認した．

以上の (a)-(c) の検証を踏まえて，本研究では監視期間 T を 5 秒と定める．本検証の限りにおいては，(c) によって 5 秒後に syn/ack が到達するケースは僅かであり，そのようなケースにおいてセッションが成立する例は無かった．万が一，ハニーポットが代理で応答をしている最中に syn/ack が遅延した場合は，ハニーポットの通信が優先される．しかし，syn/ack が遅延するケースは限りなく稀であり，実運用上も無視できる程度である．

4.2 システムのパフォーマンス評価

本システムは図 2 に示すとおり，Linux の netfilter の機能を用いて高速化を図っている．本システムは恒常的な IP アドレスのリストを持つ必要がなく，現時点でアクティブなセッションのアドレスのみをメモリ上に保持する．IP アドレスの比較に伴う処理と本システムのスループットを比較するためにシステムのフローテーブル上に 10 万個のフローを登録した状態でスループットを測定したところ，1000Base-T の接続において 950.2Mbps の速度を計測した．この速度はフローテーブルが空の場合とほぼ同じである．実際のハニーポットの通信において，帯域を占めるバイナリの平均サイズは SOPHOS [18] によると 2005 年で 126kB，2010 年で 338kB である．したがってハニーポットの運用にあたりシステムに高いスループットは要求されない．以上から本システムは運用にあたり十分な転送性能を持つことを確認した．

次に本システムの特徴である，逆向きのフローが観測された場合に，通信を遮断するまでの速度を計測した．その結果を図 7 に示す．縦軸は逆向きフローの検出からハニーポットによる通信を遮断するまでの時間を表す．横軸は IP アドレスの線形の比較回数，すなわち O(n) の計算量による比較であり，フローテーブルのアドレスのサイズとは異なる．比較回数が 1 の場合は 19 μ sec，アドレス比較数が 10，000 の場合は 92 μ sec である．実際は TCP のスライディング・ウィンドウの機能により，ハニーポットがパケットを超短時間にまとめて送信する場合があるため，逆向きフローの検出から数パケットはハニーポットによるパケットの転送を許す可能性がある．しかし，4.1 の検証によって，逆向きのフローが重複した場合には成立済みのフローが優先される．そのため通信をブロックするタイミングが遅れた場合も TCP 通信を乱す可能性は小さい．

4.3 システムの実稼働による評価

本研究では 3.3 節で示した DarkFlow 検出システムを早稲田大学のゲートウェイ上に実装をしてセンサと主に稼働させて 2011 年 7 月の 2 日間にわたり検証を実施した．観測対象のサブネットの大きさは/17 であり，32，768 個の IP アドレスを持つアドレス空間である．対象サブネットには稼働中のホストを多数含む．本節では初めにシステムに保持する遅延キューとフローテーブルの挙動を観察し，続いてセンサによって観測されたパケットを分析して，ケーススタディとして攻撃の分析を試みる．

4.3.1 遅延キューとフローテーブル

システムは応答の無い syn パケットを検出するために，監視期間 (T) の間 syn パケットを保持しておく遅延キューを持つ．図 8 は遅延キューに syn パケットが追加された個数 N_{add} と遅延キューから転送された個数 N_{fwd} の関係を示す．逆方向フローの検出により削除されたパケット数 (N_{del}) は両者の差分 ($N_{add} - N_{fwd}$) で表される．監視サブネットに対して外部からスキャン活動が行われると，遅延キューのサイズが一時的に大幅に増加するが，監視期間 (T=5sec) を過ぎると直ちに削除される．ここで，T 時間以内において ΔN_{add} と ΔN_{fwd} の増減が一致している箇所は syn スキャン攻撃の影響と考えられる．ΔN_{fwd} が小さく推移している時間帯は単一の攻撃もしくはスロースキャン等による影響と考えられる．

次に遅延キューとフローテーブルのレコード保持数の関係を図 9 に示す．各キューとテーブルの個数は 10 秒おきにサンプリングしている．フローテーブルは最後の通信が観測されてか

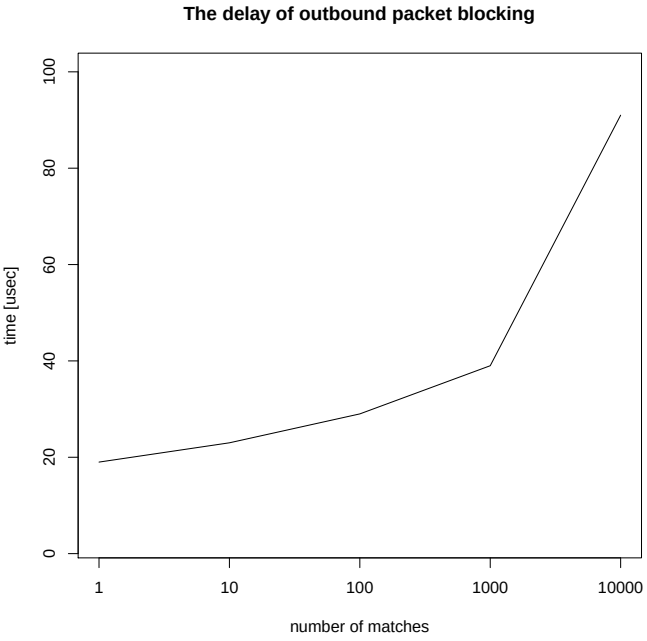

図 7　逆方向フロー検出時のパケット遮断速度

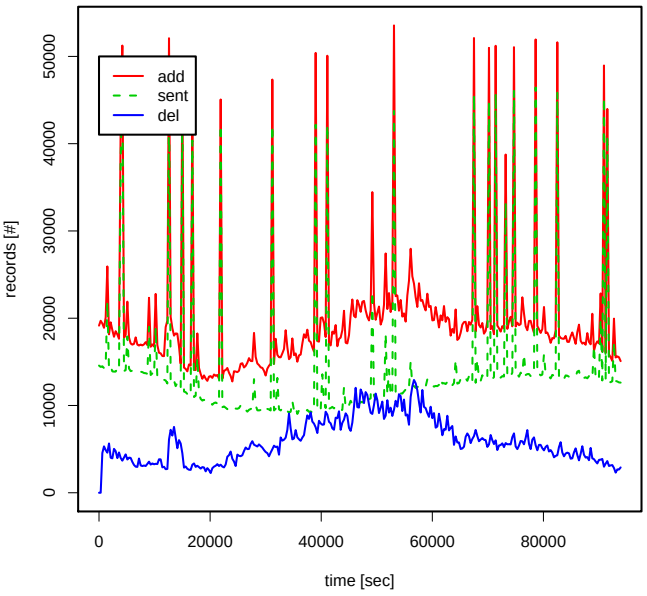

図 8　遅延キューへの syn パケット追加数と転送数

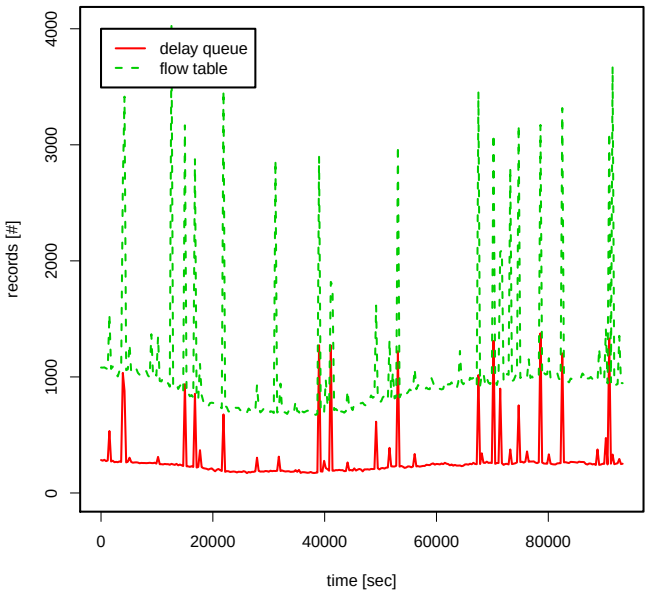

図 9　遅延キューとフローテーブルの保持レコード数の関係

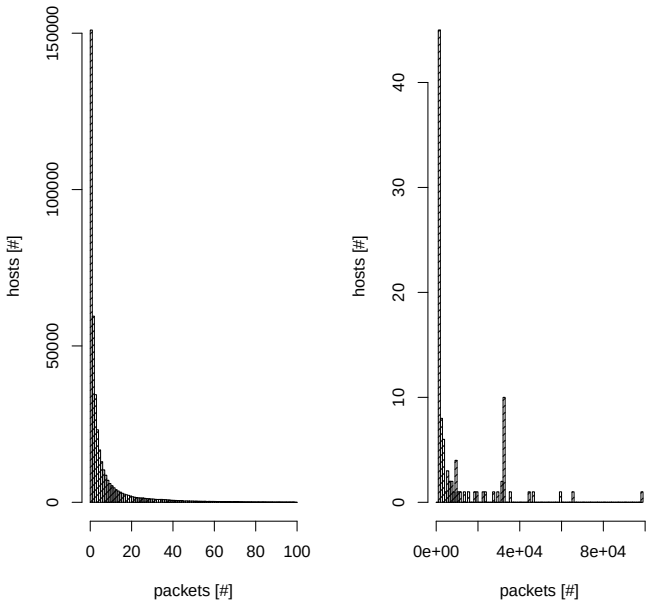

図 10　ユニーク送信元 IP アドレスのパケット送信個数

ら 60sec 以上経過すると削除する仕様である．フローテーブルは遅延キューからパケットが送出される瞬間に登録される．図 8 と同様に外部からの syn スキャンに伴うスパイク状の特徴が見られる．フローテーブルの登録数が 10 万レコードを超えた際もシステムの負荷に大きな変化は見られない．

4.3.2　センサのログ解析

上述のシステムの転送先ではセンサ・サーバを稼働させている．本節ではシステムによって転送されるパケットを元に攻撃の分析を試みる．

	packet[#]	unique host[#]
tcp/445	3,234,928	384,803
tcp/1433	421,806	508
tcp/22	262,383	544
tcp/80	138,171	2,960
tcp/3389	130,667	144
tcp/2134	69,527	3
tcp/135	65,989	67
tcp/25	38,827	1,344
tcp/3306	36,162	127
tcp/5900	35,099	2
tcp/443	21,696	1,156

表 2　TCP ポート番号と観測パケット数およびユニークホスト数

表 2 はセンサによって観測されたパケット・ログから，観測パケット数が多い 11 ポートを抽出し，パケット数と送信元ユニーク IP アドレス数を表示したものである．Microsoft の SMB プロトコルで用いられる tcp/445 は，パケット数，ユニーク IP アドレス数ともに圧倒的に多い．一方で，tcp/2134 や tcp/5900 のように僅かなホストにより大量の syn 要求が送信されるケースも存在する．

図 10 はパケットをユニーク送信元 IP アドレスで集約してパケット送信数の分布を示している．98%のユニーク IP アドレスは監視期間内に 20 パケット以内のパケットしか送信しない．しかし，ごく一部のホストは大量のパケットを送信しており，中には 1 ホストから/17 のサブネットの tcp/445 に対して 100,000 回パケットを送信する極端な syn flood も存在した．

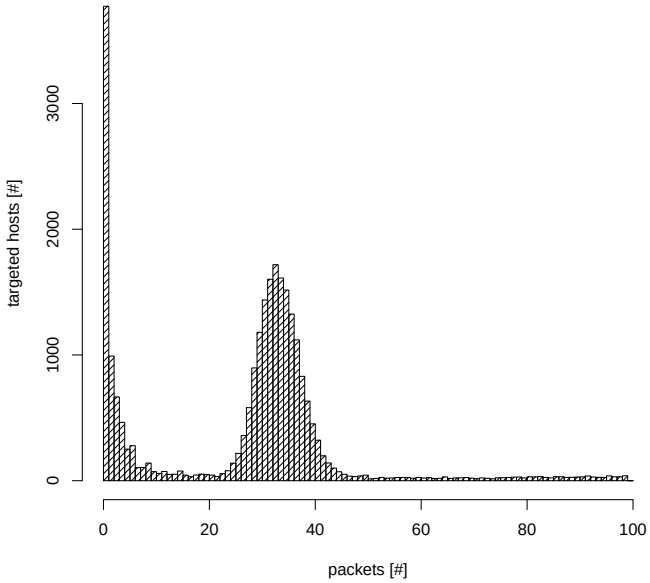

図 11　監視サブネットに対するパケット送信個数

図 11 は syn パケットの宛先ユニーク IP アドレスに対するパケット個数の分布を表している．アドレス空間 /17 に含まれる IP アドレス全体の 3 割は対しては，観測期間中に 10 パケット以下しか観測されていない．一方で，5 割の IP アドレスに対しては 20 から 40 個のパケットが観測されている．また図 11 では省略しているが，12 個の IP アドレスについては 1 万個を超えるパケットを観測している．この結果から，監視サブネットにおける不正なパケット送信個数は IP アドレスごとに偏りが見られることを確認した．

5. ま と め

本研究はトラフィック中から syn パケットに対して syn/ack の応答がない DarFlow に着目し，トラフィック中から不正な通信を抽出する方式を提案した．具体的には syn パケットに対して syn/ack を一定時間返さない場合に DarkFlow と見なす．実環境のデータを用いた検証により，本方式は正常な通信を誤って DarkFlow と見なす可能性が極端に小さいことを示した．さらに，もしも syn/ack の応答が遅延した場合は通信が確立しないことを検証し，本システムが正常な通信を乱さないことを測定実験により示した．

次に DarkFlow 検出システムを稼働中のネットワークのゲートウェイに実装し，/17 のサブネットを対象に観測実験を行った．検証ではスループット測定によるパフォーマンスやシステムの遅延キューとフローテーブルの挙動を測定した．最後にシステムと連携させたセンサ・サーバで得られたログを元に，ケーススタディとして攻撃の分析を試みた．ポート番号や IP アドレスに基づく網羅的な分析結果が得られることを示した．

本手法は未使用の IP アドレスのリストを保持する必要がなくフローベースで攻撃の通信を判別する．従来のブラックリストや未使用 IP アドレスを用いる手法と比較して，本実装は監視対象のアドレス空間の大きさに依存しない．したがって大規模なネットワークや IPv6 環境にも適用することが可能である．現在は提案システムとハニーポットを連携させて長期間稼働させたアクティブ方式による検証を実施している．本システム独自の有益なデータを元にインターネットの脅威をいっそう効率的に検出することを今後の課題とする．

文 献

[1] C. Economics, “Malware Report: The Economic Impact of Viruses, Spyware, Adware, Botnets, and Other Malicious Code,” Technical report, tech. rep., Computer Economics, 2007.

[2] “SANS Institute,” http://www.sans.org/.

[3] “WCLSCAN,” http://www.wclscan.org/.

[4] “JPCERT/CC,” http://www.jpcert.or.jp.

[5] “警視庁セキュリティポータルサイト @police,” http://www.cyberpolice.go.jp/.

[6] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson, “The Internet Motion Sensor: A distributed blackhole monitoring system,” Proceedings of the 12th ISOC Symposium on Network and Distributed Systems Security (SNDSS)Citeseer, pp.167–179 2005.

[7] G. Huston, “The changing Foundation of the Internet: confronting IPv4 Address Exhaustion,” The Internet Protocol Journal, vol.11, no.3, pp.19–36, 2008.

[8] “The IUCC/IDC Internet Telescope,” http://noc.ilan.net.il/research/telescope/.

[9] L. Spitzner, Honeypots: tracking hackers, Addison-Wesley Professional, 2003.

[10] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling, “The nepenthes platform: An efficient approach to collect malware,” Lecture Notes in Computer Science, vol.4219, pp.165–184, 2006.

[11] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A.C. Snoeren, G.M. Voelker, and S. Savage, “Scalability, fidelity, and containment in the potemkin virtual honeyfarm,” ACM SIGOPS Operating Systems Review, vol.39, no.5, pp.148–162, 2005.

[12] 下田晃弘，後藤滋樹，“フローデータからの Dark IP 抽出による脅威観測法，” IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, vol.J92-B，no.1，pp.163–173，2009.

[13] A. Shimoda, T. Mori, and S. Goto, “Sensor in the dark: Building untraceable large-scale honeypots using virtualization technologies,” Proceedings of IEEE/IPSJ SAINT 2010, pp.22–30, 2010.

[14] O. Andreasson, “Iptables tutorial 1.2. 2,” 2010.

[15] V. Jacobsen, C. Leres, and S. McCanne, “Tcpdump / libpcap,” 2005.

[16] “TCP/IP and NBT configuration parameters for Windows XP,” http://support.microsoft.com/kb/314053/en.

[17] “hping Documentation,” http://www.hping.org/documentation.php.

[18] “Sophos naked security: How large is a piece of malware?,” http://nakedsecurity.sophos.com/.