Vol.2018-IOT-42 No.9
2018/6/29

出口対策としてのDNS問合せに基づく端末用不審アクセス検知・遮断システム

柊 邦雄[1,†1]　山井 成良[1,a)]　金 勇[2]　北川 直哉[1]　友石 正彦[2]

概要：インターネットの普及に伴って，マルウェア感染による被害が世界中で深刻な問題になっており，マルウェア感染を早期に検知することが必要とされている．コンピュータに侵入したマルウェアは，攻撃者からの指令を受信するためにC&Cサーバと通信を行うことが多い．C&C（Command and Control）サーバは攻撃対象のホストをIPアドレスで指定することがあり，その場合マルウェアはDNSによる名前解決をせずに直接IPアドレスで通信先のホストを指定してTCP接続を行おうとする．本研究では，DNSの名前解決の動作に着目して，マルウェアが行う通信をDNS問合せの有無に基づいて検知・遮断するシステムを提案する．

キーワード：
DNS，ファイアウォール，マルウェア，出口対策，C&Cサーバ

Suspicious Access Detection and Blocking System on User Terminals for Outbound Measure Based on DNS Queries

Kunitaka Kakoi[1,†1]　Nariyoshi Yamai[1,a)]　Yong Jin[2]　Naoya Kitagawa[1]　Masahiko Tomoishi[2]

Abstract: With the wide spread of the Internet, the damage caused by malware infection has become a serious problem all over the world. To mitigate this problem, early detection of malware infection is required. After intruding into a PC, malware often communicates with a Command and Control (C&C) server to receive instructions from attackers. The C&C server usually specifies target hosts by their IP addresses. In this case, the malware tries to establish TCP connections to the specified hosts directly without DNS name resolution. In this research, we focus on the DNS name resolution and propose a system that detects and blocks the communication performed by malware based on the absence of DNS queries.

Keywords:
DNS, firewall, malware, outbound measure, C&C server

1. はじめに

インターネットを利用した様々なサービスや製品が存在している現在，インターネットは我々の生活に必要不可欠な社会インフラである．その一方で，インターネットを介して行われるサイバー攻撃の手法も高度化している．サイバー攻撃に利用されるマルウェアも日々新しいものが出現しており，マルウェアの感染被害の増加が世界中で深刻な問題となっている [1]．計算機がマルウェアに感染すると，個人情報の盗難，フィッシングサイトへの誘導，spamメー

[1] 東京農工大学工学研究院
Institute of Engineering, Tokyo University of Agriculture and Technology
2-24-16, Nakacho, Koganei, Tokyo 184–8588, Japan
[2] 東京工業大学学術国際情報センター
Global Scientific Information and Computing Center, Tokyo Institute of Technology,
2-12-1, O-okayama, Meguro, Tokyo 152–8550, Japan
[†1] 現在，株式会社日本レジストリサービス
Presently with Japan Registry Services Co., Ltd.
[a)] nyamai@cc.tuat.ac.jp

ルの送信やDDoS（Distributed Denial of Service）攻撃などの悪質な行為への加担等の重大な被害を受ける恐れがある．特に大きな脅威と考えられているサイバー攻撃として，攻撃者が明確な目的を持って特定の組織や企業を狙う標的型攻撃が挙げられる [2]．この中でも特にATP（Advanced Persistent Threat）攻撃では未知の脆弱性を狙ったゼロデイ攻撃や，なりすましメールに新規のマルウェアを添付して送りつける標的型メール攻撃など様々な攻撃手法が利用され，初期の侵入段階で防ぐことが困難になっている．そのため，マルウェアへの感染を前提とし，感染後に実被害が発生しないようにする出口対策が重要視されている．

出口対策として，組織内ネットワークから組織外への通信を分析する方法がこれまでに数多く提案されている．たとえば文献 [3] ではアクセススイッチを通過するパケットを監視して異なり数（cardinality）に基づいて分析を行い，たとえば同一送信元IPアドレスから多くの宛先IPアドレスの特定のポート番号に対するパケットが検出された場合に異常とする方法を提案している．また，文献 [4] ではDNS（Domain Name Service）のログ監視に基づき端末のマルウェアへの感染を検出する方法が紹介されている．しかし，これらの方法は統計情報に基づくため，マルウェアに一定の活動を許すことになる．また，特に後者の方法ではマルウェアが自身では名前解決を行わず，たとえばC&C（Command and Control）サーバから指定されたIPアドレスに直接アクセスしたり，特定範囲のネットワーク全体をスキャンしたりするような動作をすると，検出できない．

そこで本稿では端末がDNSによる名前解決を行わないで他のホストに直接アクセスしようとする通信を検出し，これをマルウェアによる通信と見なして検出および警告するシステムについて述べる．このシステムではDNS（Domain Name System）問合せを端末内で監視してその応答を一時的なホワイリストに登録し，ホワイトリストに登録されていないIPアドレス宛のパケットは遮断するように動作する．また，未知のプロセスによる通信を検出・遮断する機能や誤判定に備えて警告画面を表示する機能も追加し，システムの有用性の向上を図る．

2. 関連研究

マルウェア対策には様々な方法が提案されているが，本稿ではDNSを用いる方法に絞って紹介する．

マルウェアは初期段階でC&Cサーバと通信するために名前解決を行うため，前節で述べた文献 [4] 以外にもDNSトラフィックを基にしたマルウェアの検知手法が数多く提案されている．

たとえばDNS問合せの成否，NSレコードの問合せ履歴，1日に送信するFQDN数など，ホストが送受信するDNS問合せや応答の特徴量に基づきマルウェア感染ホストを検知する方法が多数提案されている [5], [6], [7], [8], [9]．しかし，これらの方法はいずれもDNSトラフィックを十分に取得して解析する必要があるため，マルウェアの感染検知に時間を要し，その結果マルウェアに一定の活動を許すことになる．

また，C&Cサーバや悪性ドメインとの接続を遮断するためのDNS関連技術として，DNSBL[12] やDNSシンクホール [13] が知られている．これらの技術を用いれば，マルウェア感染ホストが悪性ドメインやC&Cサーバにアクセスしようとしたとき，それを検知・遮断したり警告画面を表示したりすることができる．しかし，マルウェア作成者がFast Flux[14] 等を利用してこれらの技術で用いられるブラックリストを巧妙に回避したり，新たなC&Cサーバや悪性ドメインの発生速度にブラックリストの更新が追いつかなかったりする問題がある．なお，マルウェアの中にはDNSシンクホールを回避するために外部へ直接DNS問合せを送信するものがあるが，これを検出することによりマルウェアを検知する方法 [15] が提案されている．

3. 提案システム

3.1 提案システムの概要

前節で述べたように，DNSに基づいたマルウェア検知方法は数多く提案されている．しかし，これらすべての方法はマルウェア感染ホストがDNS問合せを行うことを前提としており，IPアドレスを直接指定して通信を行う場合にはこれを検知できない．特にC&Cサーバへの最初の接続を遮断できなかった場合，マルウェアがC&CサーバによりIPアドレスで指定されたホストや同一ネットワーク内のホストに攻撃する可能性があり，マルウェアのこのようなアクセスを検知・遮断することが重要になる．また，DNS問合せを行わずに名前解決を行うために用いられるhostsファイルをマルウェアが書き換えた結果，意図したものとは異なるホストへのアクセスも同様に検知・遮断することが重要である．

そこで，我々はクライアントが通常利用するスタブリゾルバやフルリゾルバを通した名前解決が行われていないIPアドレス宛の通信が発生している場合，その通信はマルウェアによるものである可能性が高いと判断し，名前解決されていないIPアドレス宛のTCP接続を検出して遮断するシステムを提案する．その際，正当なプログラムによる通信を遮断してしまう危険性があるため，不審な通信をすぐ完全に遮断するのではなく警告画面をポップアップで表示し，ユーザに通信をするかどうかを選択させる機能を本システムに追加する．さらに，提案システムでは未知のプロセスが行うTCP通信についてもマルウェアによる通信の可能性があると判断し，同様の処理を行う．なお，本システムはTCP通信のみを対象としているが，UDP通信を対象に加えることも容易である．但し，TCP通信とは異なり，ユーザに通信をするかどうかを選択させる機能が確

実に動作するとは限らない点に注意する．

提案システムは従来の DNS に基づいたマルウェア検知方法と競合するものではなく，むしろ従来の方法と併用することにより相補する関係にあるといえる．

3.2 提案システムの構成

提案システムは**図 1** に示すようにクライアントの PC 上で動作する DNS プロクシ，パケット・フィルタリング型ファイアウォール，警告画面表示モジュールから構成される．それぞれの構成要素の動作を以下に示す．なお，簡素化のため，以下の説明ではネットワーク層プロトコルは IPv4 とし，また端末の OS は Windows 7 とする．

3.2.1 DNS プロクシ

DNS プロクシはクライアントの PC 上で動作して，スタブリゾルバから DNS 問合せパケットを受け取ると，それを DNS フルリゾルバにフォワーディングする．その後，もし DNS フルリゾルバから名前解決結果として受け取ったレコードが A レコードであれば，IP アドレスのホワイトリスト（IP_WL）にその A レコードに記述されている IP アドレスを書き込み，名前解決結果をスタブリゾルバに返す．

3.2.2 パケット・フィルタリング型ファイアウォール

ファイアウォールは TCP コネクションを確立するためにアプリケーションが送信した SYN パケットを検知すると，IP_{WL,BL} 参照機能を利用してその SYN パケットの宛先 IP アドレスが IP_WL や IP_BL に登録されているかを確認する．さらに，プロセス名特定機能を利用して TCP 通信を始めようとしているプロセス名を特定し，そのプロセス名がホワイトリスト（Process_WL）に登録されているか照合を行う．宛先 IP アドレスが IP_WL にあり，既知のプロセスである場合，その通信は正常なものであると判定し，SYN パケットを通過させ TCP コネクションの確立させる．また，宛先 IP アドレスがブラックリスト（IP_BL）に登録されている場合にはこれを廃棄する．それ以外の場合は疑わしい通信だと判断し，その SYN パケットを保留する．その際，不審な通信を検知したことをユーザに通知するための警告画面を表示し，その通信を許可するかを選択させる．通信が許可された場合は保留していた SYN パケットを中継し，TCP コネクションを確立する．一方，通信が許可されない場合は対象のホスト宛の SYN パケットを廃棄し，当該 TCP 通信を遮断する．

3.2.3 警告画面表示モジュール

このモジュールはファイアウォールで名前解決されていない IP アドレス宛の SYN パケットまたは未知のプロセスが送信した SYN パケットをファイアウォールが検出すると呼び出され，警告画面を表示して，ユーザに通信を許可するかどうかの選択を要求する．

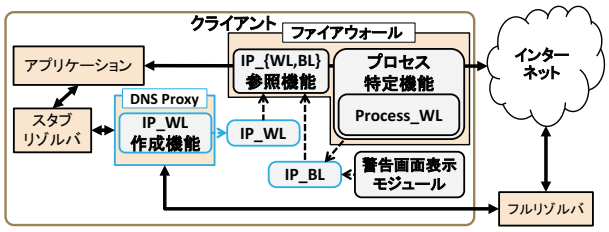

図 1 提案システムの構成

3.3 TCP 接続のタイムアウト延長

本システムは，クライアントからの SYN パケットを廃棄して TCP コネクションを確立させないことで，対象とする通信を遮断する．したがって，不審な通信を検出して警告画面が表示された場合，OS の設定やアプリケーションで実装される設定に依存する TCP コネクションのタイムアウトが発生するまでにユーザは通信の許可・不許可を選択する必要がある．このとき，タイムアウトの時間が OS の設定とアプリケーションの設定で異なる場合は短い方が適用される．

ここで，例えば TCP コネクションのタイムアウト後にユーザが通信の可否を選択すると，たとえ通信を許可した場合にも通信が開始されないことが想定される．したがって，本システムの実用的な利用には十分長い TCP コネクションタイムアウト時間が必要となる．OS の設定による TCP 接続のタイムアウト時間は SYN パケットの再送回数によって決定される．Windows 7 におけるデフォルトの SYN パケットの再送回数は 2 回であり，初期再送タイムアウト時間は 3 秒と設定されている．パケットが再送されるたびに再送タイムアウト時間は 2 倍に増えるため，このときの TCP コネクションのタイムアウト時間は 3+6+12=21 秒となる．しかし，ユーザが表示された警告画面を見て選択するのに 21 秒は短過ぎるように思われる．そこで，本システムを利用する場合，ユーザが余裕を持って選択できるように OS の SYN パケットの再送回数を増やせるようにしたい．実装に用いた Windows 7 では修正プログラムを適用すると netsh コマンドを用いて TCP の SYN パケットの再送回数を変更することができる [16][*1]．このコマンドを用いて「netsh interface tcp set global MaxSynRetransmission=再送回数」を実行することにより，Windows 7 では SYN パケットの最大再送回数を 2 から 8 の間で変更可能である．各再送回数における TCP コネクションのタイムアウト時間を**表 1** に示す．これによりユーザが通信の可否を判断するために与えられる時間は最大で 333 秒まで延長できる．

4. 提案システムの実装

我々は提案システムを Windows 7 が稼働する PC 上で

[*1] Windows 10 では標準で同様のコマンドを用いることが可能である．

表 1 SYN パケットの再送回数とタイムアウト時間

再送回数	タイムアウト時間
2	21 秒
3	45 秒
4	93 秒
5	153 秒
6	213 秒
7	273 秒
8	333 秒

実装した．本節では提案システムの詳細な実装方法について述べる．

4.1 DNS プロクシ

DNS プロクシは Perl モジュール Net::DNSServer::Proxy[17] を利用して作成した．PC の DNS サーバ設定はローカルホスト（127.0.0.1）を指すようにし，DNS プロクシはフルリゾルバとして Googe Public DNS（8.8.8.8）を参照するようにした．

DNS プロクシは基本的にはすべての DNS 問合せおよびその応答を中継する．但し，応答中に A レコードが含まれる場合にはその A レコード中の IP アドレスを IP アドレス用ホワイトリスト（図 1 中の IP_FW）に登録する．

4.2 ファイアウォール

ファイアウォールの実装には WinDivert[18] を用いた．処理の流れを以下に示す．

(1) ファイアウォールは初期化のため WinDivertOpen() を実行する．フィルタの対象としてループバックアドレスを除く宛先へ送信する SYN パケットを指定する．各ブロックリスト，ホワイトリストの初期化を行う．
(2) WinDivert がフィルタの対象となる SYN パケットを検出すると，ファイアウォールはこれを WinDivertRecv() 経由で受け取る．
(3) ファイアウォールはまず取得した SYN パケットの宛先 IP アドレスが IP_BL に登録されているかどうかを調べる．もし登録されていれば，取得したパケットを破棄して，処理を終了する．
(4) ファイアウォールはこの SYN パケットを送出したプロセスを特定し，そのプロセス名が Process_WL に登録されているかどうかを調べる．もし登録されていなければ，新たにスレッドで警告画面表示モジュールを起動して処理を終了する．
(5) ファイアウォールは宛先 IP アドレスが IP_WL に登録されているかどうかを調べる．もし登録されていなければ，宛先 IP アドレスを IP_BL に追加し，新たにスレッドで警告画面表示モジュールを起動して処理を終了する．
(6) 宛先 IP アドレスが IP アドレスが IP_WL にある場合，WinDivertSend() を用いて SYN パケットを送出し，処理を終了する．

表 2 端末用 PC の諸元

OS	Windows 7 32bit
CPU	Intel(R) Core(TM) i5-4210M 2.60GHz
主記憶容量	4GB

ここで，(4) におけるプロセス名の特定には Windows 標準コマンドである netstat と tasklist を用いた．具体的には，まず netstat コマンドを用いて送信元ポート番号が一致する TCP フローのプロセス ID を特定し，次に tasklist コマンドを用いてプロセス ID に対するプロセス名（イメージ名）を特定するようにした．

4.3 警告画面表示モジュール

本モジュールはユーザによる不審な通信の可否判断を待っている間にもファイアウォールで別パケットを処理できるようにするため，新たに生成されたスレッドの中で呼び出されるようにした．

このモジュールを呼び出す際には SYN パケットの宛先 IP アドレスおよび IP_WL への登録の有無，プロセス名および Process_WL への登録の有無が引数として渡され，警告画面が不審と判断した理由とともにユーザに示される．この表示に対して，ユーザが接続を拒否した場合には単にこの SYN パケットを廃棄する．一方，ユーザが接続を許可した場合にはその宛先IP アドレスを IP_BL から削除し，保留していた SYN パケットを WinDivertSend() を用いて送出する．

5. 試作システムの動作検証

5.1 動作の確認

実際に本システムを端末用 PC に構築し動作確認を行った．使用した PC の諸元を**表 2** に示す．また，タイムアウト時間を延ばすために SYN パケットの再送回数を 8 回に変更した．

この状態で Web ブラウザとして 3 種類のプログラム Internet Explorer 11（iexplore.exe），Google Chrome（chrome.exe）および curl（curl.exe）を用いて Web サイト www.example.com（93.184.216.34）にアクセスを行った．

まず，既知のプロセス名として 2 種類のプログラム「iexplore.exe」と「chrome.exe」をファイアウォール中の Process_WL に予め登録し，その状態で 3 種類のプログラムそれぞれを用いて http://www.example.com にアクセスした．その結果，Internet Explorer11，Google Chrome の 2 種類のプログラムについては**図 2**，**図 3** に示すように正しく表示された．一方，curl については**図 4** に示すように curl が未知のプロセス名であることを示す警告画面が表示された．その後ユーザが「接続する」ボタンをクリックす

図 2 Internet Explorer 11 による www.example.com へのアクセス

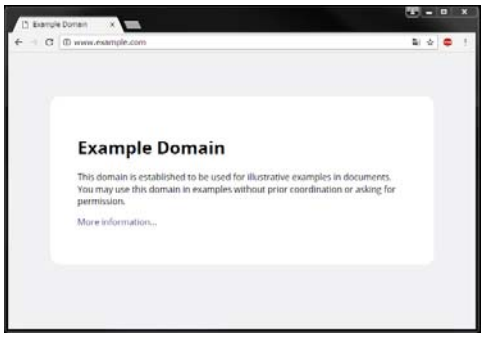

図 3 Google Chrome による www.example.com へのアクセス

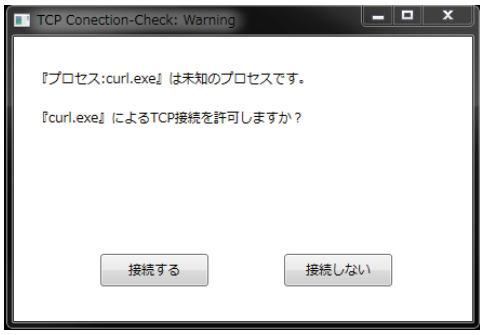

図 4 プロセス名未登録時における「curl www.example.com」実行時の警告画面

図 5 「curl http://www.example.com」のタイムアウト

ると HTML 形式の表示され，「接続しない」をクリックすると TCP 接続が確立されず，**図 5** に示すように最終的にはタイムアウトになることが確認できた．

次に，IP_WL，IP_BL を初期化し，Process_WL に curl.exe を追加した状態でコマンド「curl 93.184.216.34」を実行したところ，**図 6** に示すように宛先 IP アドレスが名前解決により得られたものではないことを示す警告画面が表示された．また，その後ユーザが「接続する」，「接続しない」をそれぞれクリックすると上記のプロセス名未登録時と同様の動作を行うことが確認できた．

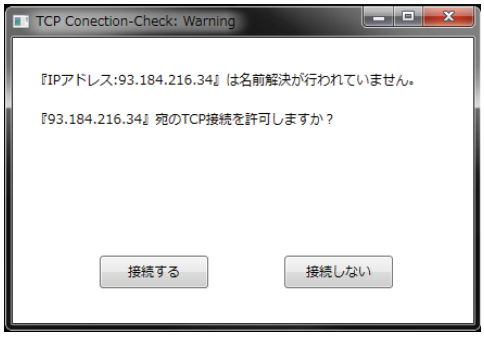

図 6 プロセス名登録時における「curl 93.184.216.34」実行時の警告画面

これらの結果から，提案システムは未知のプロセス名を持つプログラムによる通信や IP アドレスを直接指定した通信を行った場合には警告画面が表示され，その後ユーザが接続を拒否すると実際に通信が遮断されることが確認できた．

5.2 オーバヘッドの測定

次に，試作システムのオーバヘッドを測定するための実験を行った．この実験では Process_WL に予め curl.exe を登録し，curl コマンドで www.example.com にアクセスして「名前解決に要した時間」，「TCP 接続に要した時間」，「コンテンツの受信に要した時間」を測定する試行を 20 回行い，各時間の平均を求めた．また，比較のため，提案システムを用いずに同じ実験を行った場合の各時間の平均を求めた．

測定結果を**図 7** に示す．この図において「namelookup」「connect」「total」はそれぞれ「名前解決に要した時間」，「TCP 接続に要した時間」，「コンテンツの受信に要した時間」を表す．この図より名前解決オーバヘッドが 0.06 秒，TCP 接続のオーバヘッドが 0.17 秒であることが読み取れる．一方，TCP 接続確立後のコンテンツ受信のオーバヘッドは認められなかった．TCP 接続のオーバヘッドが特に大きいため，試作システムで使用している外部コマンドである netstat，tasklist の実行時間を測定したところ，それぞれ 0.06 秒，0.11 秒であり，この 2 つのコマンドの実行時間がそのままオーバヘッドになっていることが判明した．したがって，これらのコマンドを用いずに SYN パケットを送出したプロセス名を特定できればオーバヘッドを削減できる余地がある．

6. まとめ

本稿では端末上で動作する DNS の名前解決動作やプロセス名に基づく不審アクセス検知・遮断システムを提案した．本システムを DNS に基づく既存のマルウェア検知方法と組み合わせることにより，これまでには検知できなかったマルウェアを検知できる効果が期待できる．

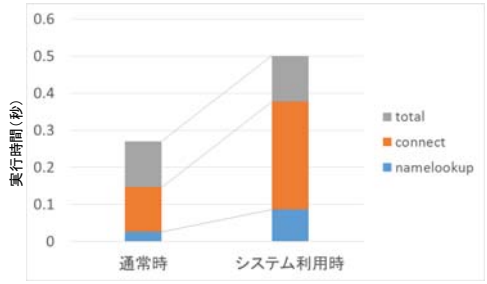

図 7 試作システムのオーバヘッド測定結果

今後の課題として，まず提案システムの有効性検証が挙げられる．また，外部プログラムを用いずにプロセス名を特定する方法の開発により，オーバヘッドを削減することも重要な課題として挙げられる．

参考文献

[1] McAfee Labs: McAfee Labs Threats Report March 2018 (online), available from ⟨https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-mar-2018.pdf⟩ (accessed 2018.05.31), March 2018.
[2] 独立行政法人情報処理推進機構：『新しいタイプの攻撃』の対策に向けた設計・運用ガイド改定第 2 版（オンライン），入手先 ⟨https://www.ipa.go.jp/files/000017308.pdf⟩ (2018.05.31 参照), 2011 年 11 月.
[3] 伊藤昂平，佐藤聡，新城靖：“アクセススイッチにて取得するサンプリングパケットへの異なり数分析の適用”，情報処理学会研究報告，Vol.2017-IOT-37，No.10，pp.1–6，2017 年 5 月.
[4] 佐藤一道，石橋圭介，豊野剛，三宅延久：“DNS トラフィックデータを利用したボット感染者検出方法”，情報処理学会研究報告，Vol.2009-IOT-7，No.11，pp.1–6，2009 年 10 月.
[5] Ricardo Villamarán-Salomón, José Carlos Brustoloni: “Identifying Botnets Using Anomaly Detection Techniques Applied to DNS Traffic”, *Proceedings of The 5th IEEE Consumer Communications & Networking Conference (CCNC 2008)*, pp.476–481, January 2008.
[6] Ricardo Villamarán-Salomón, José Carlos Brustoloni: “Bayesian bot detection based on DNS traffic similarity”, *Proceedings of the 2009 ACM symposium on Applied Computing*, pp.2035–2041, March 2009.
[7] Hyunsang Choi, Heejo Lee: “Identifying botnets by capturing group activities in DNS traffic”, *Computer Networks*, Elsevier, Vol.56, Issue 1, pp.20–33, January 2012.
[8] Hikaru Ichise, Yong Jin, Katsuyoshi Iida: “Detection method of DNS-based botnet communication using obtained NS record history”, *Proceedings of 2015 IEEE 39th International Conference on Computer Software and Applications (COMPSAC 2015)*, Vol.3, pp.676-677, July 2015.
[9] 田辺瑠偉，鉄穎，水戸慎，牧田大佑，神薗雅紀，星澤裕二，吉岡克成，松本勉：“長期動的解析によるマルウェアの特徴的な DNS 通信の抽出”，コンピュータセキュリティシンポジウム 2012 論文集，情報処理学会，pp.712–719，2012 年 10 月.
[10] 渡辺拳竜，池部実，吉田和幸：“DNS クエリログのクエリ数に着目した異常ホストの検出”，マルチメディア，分散，協調とモバイル (DICOMO 2014) シンポジウム論文集，1G-4，pp.205–210，情報処理学会，2014 年 7 月.
[11] 牧田大佑，吉岡克成，松本勉：“マルウェア感染ホストの特定を目的とした DNS 通信の可視化”，情報処理学会研究報告，Vol.2013-IOT-21，No.7，pp.1–6，2013 年 5 月.
[12] J. Levine: DNS Blacklists and Whitelists, RFC5782, IETF, February 2010.
[13] Guy Bruneau: DNS Sinkhole (online), available from ⟨https://www.sans.org/reading-room/whitepapers/dns/dns-sinkhole-33523⟩ (accessed 2018.05.31), August 2010.
[14] J. Riden: How fast-flux service networks work (online), available from ⟨https://www.honeynet.org/node/132⟩ (accessed 2018.06.02), August 2008.
[15] Yong Jin, Hikaru Ichise, Katsuyoshi Iida: “Design of detecting botnet communication by monitoring direct outbound DNS queries”, *Proceedings of The 2nd IEEE International Conference on Cyber Security and Cloud Computing (CSCloud 2015)*, pp.37–41, November 2015.
[16] Microsoft Support: Hotfix enables the configuration of the TCP maximum SYN retransmission amount in Windows 7 or Windows Server 2008 R2 (online), available from ⟨https://support.microsoft.com/en-us/help/2786464/hotfix-enables-the-configuration-of-the-tcp-maximum-syn-retransmission⟩ (accessed 2018.06.02), September 2013.
[17] Rob Brown: Net::DNSServer::Proxy - Forwards requests to another DNS server (online), available from ⟨https://metacpan.org/pod/Net::DNSServer::Proxy⟩ (accessed 2018.06.02), November 2002.
[18] Basil: WinDivert 1.4: Windows Packet Divert (online), available from ⟨https://reqrypt.org/windivert.html⟩ (accessed 2018.06.02).