情報処理学会論文誌 Vol.61 No.9 1428–1443 (Sep. 2020)

TCP Connection Tableを悪用した組織内ネットワークへのマルウェア拡散の特性評価と対策検討

川口 信隆[1,a)]

受付日 2019年11月25日, 採録日 2020年6月1日

概要：本稿では，アドレススキャンのかわりに感染端末内の TCP 接続情報（TCP Connection Table, TCT）を悪用して組織内ネットワークに拡散する TCT マルウェアに対する特性評価および拡散抑止方式を検討する．まず，1 万台以上の端末の通信ログが含まれる公開データセットを用いたシミュレーションにより，TCT マルウェアの拡散速度は数十 scan/s のアドレススキャン型マルウェアに相当し，組織内ダークネット監視や異常検知などの従来方式では早期発見が難しいことを明らかにした．さらに，TCT に偽の接続情報を挿入することで TCT マルウェアの拡散を早期検知・抑制する方式を設計・実装した．シミュレーションおよび実際の TCT マルウェア NotPetya を用いた評価実験を通じ，本方式により拡散台数を数分の 1 に抑えられることを確認した．

キーワード：マルウェア対策，標的型攻撃，感染シミュレーション

Understanding the Malware Propagation in Enterprise Network by Abusing TCP Connection Table

Nobutaka Kawaguchi[1,a)]

Received: November 25, 2019, Accepted: June 1, 2020

Abstract: In this paper, we study characteristics of malware which propagates over enterprise networks by abusing TCP Connection Table (TCT) retrieved from compromised hosts instead of IP address scans, and design a countermeasure method. First, through simulations with open dataset including connection logs from more than 10,000 hosts, we found that the propagation speed of TCT malware is comparable to IP address scan malware with a rate of tens scan/s and traditional approaches such as darknet monitoring and anomaly detections are ineffective against the propagation. Thus, second, we design and implement a detection and containment method that inserts a few decoy connections into TCT of each host. Through simulations and a real TCT malware NotPetya, we confirmed that the proposed method can reduce the number of compromised hosts to a few percent.

Keywords: malware, targeted attack, epidemic simulation

1. はじめに

近年，破壊型攻撃，制御システムを狙った標的型攻撃，ランサムウェアなどサイバー攻撃はますます高度化・多様化している．これにともない攻撃に用いられるマルウェアが具備する機能も複雑化している．代表的なものとしては 2017 年 5 月に発生した WannaCry [1] があげられる．WannaCry はネットワークワーム機能を有したランサムウェアである．Microsoft Windows のファイル共有に用いられる SMB の脆弱性 MS17-010 [2] を悪用することで組織内ネットワーク（企業などのエンタプライズネットワーク）およびインターネット上で急速に拡散し世界中で甚大な被害を及ぼした．

WannaCry に続き，2017 年 7 月にロシアからウクライナに向けた標的型攻撃で用いられたマルウェア NotPetya [3]

[1] 株式会社日立製作所
Hitachi Ltd., Yokohama, Kanagawa 244–0817, Japan
[a)] nobutaka.kawaguchi.ue@hitachi.com

もまた，MS17-010 を悪用しエンタプライズネットワークに拡散し，大きな被害を及ぼした．本稿では特に，NotPetya が有している高度な拡散機能に着目する．従来マルウェアが組織内ネットワーク内に拡散する際は IP アドレススキャンで拡散先端末を特定する方法が一般的であった．これに対して NotPetya は IP アドレススキャンに加え感染端末内の ARP Cache，DHCP アドレス割当リスト，そして TCP Connection Table 情報を悪用して効率的に拡散を行うことで従来の検知手段の多くを回避することができる．本稿では特に TCP Connection Table（以下，TCT と呼ぶ）を悪用した組織ネットワーク内への拡散を取り上げる．TCT は Windows 標準コマンドの netstat.exe などで参照可能な TCP 接続の状態情報であり，端末が現在接続しているピア端末や接続ポート番号などが記載されている．TCT からピア端末を抽出することでマルウェアはサブネット内外の端末に効率的に拡散することができる．この方法は IP アドレススキャンと異なり未使用 IP アドレスに対して IP パケットを送信しないため，組織内ダークネット（組織内ネットワークの未使用アドレス空間）監視や TCP 接続失敗回数に基づく検知，ファイアーウォールなどの端末間アクセス制御ルールを回避できる可能性が高い．しかし，IP アドレススキャンと比べ TCT を悪用して拡散するマルウェア（以下，TCT マルウェアと呼ぶ）に関する従来研究は比較的少なく，その特性や対策は十分に議論されてこなかった．本稿では TCT 拡散マルウェアの特性を評価するとともに，その拡散抑制策をシミュレーションおよび実在のマルウェアを基に検討する．

本稿ではまず，TCT の参照頻度・拡散先端末の選択基準が異なる 6 種類の TCT 拡散モデルを検討し，効率性・検知回避性の観点からその特性を定性的に論じる．そのうえで，米国ロスアラモス研究所が 2017 年に公開した，組織内ネットワーク端末 1 万台以上の TCP 接続ログを約 90 日分含むオープンデータセット（以下 LANL データセットと呼ぶ）[4] を用いて TCT マルウェアの拡散速度をシミュレーション評価し IP アドレススキャン型との定量的比較を行う．

次に，TCT マルウェア拡散抑制方式を検討する．抑制方式は端末の TCT にダミーの TCP 接続情報を挿入することで TCT を悪用したマルウェアを検知し，その後の通信を遮断することで拡散を抑止する．本方式は，アクティブ・ディフェンスの一種である「デセプション」の 1 つと位置付けられる．本稿では LANL データセットを用いたシミュレーション評価により方式の有効性を定量的に評価した．また Windows を対象とした拡散抑制方式のプロトタイプを実装し，NotPetya に対して方式が有効であることを実機評価で確認した．

本研究の結果として，(i) TCT マルウェアの拡散速度は，種類に応じて違いがあるものの，数十 scan/s の IP アドレススキャン型マルウェアに相当すること，(ii) 拡散抑制方式をネットワークの hub となる 1%程度の端末に重点的に導入することで拡散速度を数分の 1 に抑えられること，(iii) 実際のマルウェア NotPetya に対しても拡散抑制方式が有効であること，などの知見が得られた．

本稿は以下のとおり構成される．まず 2 章では従来のマルウェア拡散方法および関連技術，NotPetya に関して概観する．3 章で TCT マルウェアのモデルを述べ，4 章でシミュレーション評価を行う．5 章で，デセプションの考え方に基づく拡散抑制方式の提案，シミュレーション評価を行い，6 章で方式の実現形態である拡散抑制モジュールの設計を行う．次いで，7 章において拡散抑制モジュールの実装および実機評価を行う．8 章で研究を通じ得られた知見について考察し，9 章を本稿のまとめとする．

2. 背景

本章では，マルウェアの拡散方法の分類と拡散抑制に用いる関連技術，NotPetya の特徴について概観する．

2.1 マルウェア拡散方法

本節ではネットワークを介して拡散するマルウェアの拡散先特定方法（拡散方法）について整理する．なお，本稿で述べる「マルウェア」は特に断りがない限り，攻撃者の操作・指示なしに自立的に拡散する悪性プログラムを意味し狭義には「ネットワークワーム」と呼ばれるものである．

文献 [5] によると，マルウェアの拡散方法は IP アドレススキャン型と非アドレススキャン型に分類される．

IP アドレススキャン型はこれまでに発生したマルウェアの多くで実装されている方法であり，ある基準に基づき IP アドレス空間の中から拡散先アドレスを選択する．基本系である IPv4 を対象としたランダムアドレススキャンでは，2^{32} のアドレス空間からランダムに拡散先を選択して TCP 接続を試行する．この方法を効率化したものとして，感染端末の近傍アドレスを重点的に探索するローカルスキャン [6] やシーケンシャルスキャン [7]，BGP でルーティング可能なアドレス空間のみをスキャン対象とするルーティングスキャン [8] などがある．特にローカルスキャンは組織内ネットワークでの拡散に効率的と考えられ，WannaCry や NotPetya でも実装されている．IP アドレススキャンは未使用アドレス空間に大量のスキャン通信が発生する．この特徴に着目したコネクション接続失敗回数 [9] やダークネット観測に基づく検知方法 [10] が考案されている．

非アドレススキャン型は IP アドレススキャン以外の方法で拡散先を特定する方法であり，Hit-list 型とトポロジカル型に分類される．Hit-list 型はあらかじめ攻撃者が入手したアドレスリストに従い拡散するマルウェアの総称である．Hit-list 型マルウェアの挙動は Flash Worm [11]，Silent Worm [12] という形でより詳細にモデル化されてい

る．また，2014 年に米国企業に対して発生した破壊型標的型攻撃に使用された DestOver [14] は数十件の拡散先サーバの IP アドレスリストを保持していた．Hit-list 型マルウェアへの対処手段としては，通信発生頻度に基づく異常検知 [12], [13] や端末に割り当てる IP アドレスを定期的に変更することで事前作成されたアドレスリストを無効化する方法 [15] などが考案されている．

トポロジカル型は感染端末が有するアドレスリストや近傍端末情報を悪用して拡散する方法であり TCT 拡散もこのカテゴリに含まれる．トポロジカル型は Hit-list 型と異なりアドレスリストを事前作成しないため，攻撃先ネットワークの環境の変化に追随できる．また現時点でアクティブな IP アドレスに対してのみ拡散するためダークネット観測や接続失敗回数などに基づく検知が難しい．これまで，このアプローチに基づく P2P Worm [16] や Bluetooth Worm [17] のモデル化や対策検討は様々行われてきた．一方で，組織内ネットワークを対象としたトポロジカル型マルウェアの感染特性に対する研究は比較的少なく，.rhosts ファイルを悪用する Morris Worm [18]，.known_hosts ファイルを悪用する SSH Worm [19]，感染端末から参照可能な Windows ファイル共有を通じて拡散する Stuxnet の分析 [20] などに限られていた．

しかし近年標的型攻撃が大きな脅威になるにつれ，組織内ネットワークに侵入した攻撃者が，コマンドプロンプトあるいは Remote Access Tool（RAT）[21] を通じて Windows 標準搭載のネットワークコマンド（例：net.exe, arp.exe，netstat.exe など）を悪用し，ネットワーク内の近隣端末や主要サーバを探索する方法が広く知られるようになった [22]．これらの探索方法は MITRE が標的型攻撃者の挙動をモデル化した ATT&CK [23] では “System Network Connections Discovery” という TTP（Tactics, Techniques and Procedure）として位置づけられている．すなわち，TCT マルウェアである NotPetya やその同系統にある BadRabbit [24]，Moonraker Petya [20] は，標的型攻撃者が常用してきた探索方法を，自律的なマルウェア拡散のために応用したものといえる．

組織内ネットワークを狙うトポロジカル型マルウェアの特性に着目した検知・対策を扱った従来研究は数少ない．文献 [25] ではダミーアドレスという偽のアドレスを各端末の「コンタクト先アドレスリスト」にあらかじめ挿入し，当該アドレスへの接続を試行したマルウェアを検知・抑制する方法を検討している．文献では時間変化をともなわない静的アドレスリスト全般を抽象化して論じており，具体的なアタックベクタは想定していない．また，実際のネットワークログに基づき性能分析をしたものではない．一方本稿が対象とする TCT は実在に顕現化したアタックベクタであり，時間経過とともに接続情報が変化する，通信接続元・接続先の区別がある，といった特徴がある．また，我々は公開データセットを用い拡散抑制方式の有効性を検証するとともに，方式のプロトタイプを実装し実在のマルウェアを用いて効果を確認している．

2.2 関連技術

本稿で扱う「デセプション」は狭義には情報システム・ネットワークを対象とした「サイバー・デセプション」を指し，「攻撃者をミスリードまたは混乱させることで，防御側を利する行動を引き起させるためのセキュリティ技術」[26] を意味する．デセプションの導入目的としては主に攻撃検知・攻撃抑制，および攻撃者の挙動分析がある．デセプションに用いられる機器やデータは正規の業務やシステム運用では使用されないものであるため，これらに対するアクセスは不正なものである可能性が高い．このため攻撃検知の点では一般的な IDS や異常検知技術と比べて誤検知の発生頻度が低いことが期待できる [27]．

デセプションの具体的な方法としては，ハニーポットを組織内ネットワークに設置する方法 [28]，攻撃者の興味を引くダミー情報（偽のクレジットカード番号など）を配布する honeytokens [29]，ダミーファイルへのアクセスを基に検知を行う honeyfiles [30] などがある．5 章で述べる TCT に偽の接続情報を挿入するという提案方式は honeytokens の一種として位置づけられる．

文献 [31] では，内部端末と複数の囮サーバとの間で decoy connection を定常的に張ることでネットワークに侵入した攻撃者の進行を妨害する方式を検討している．一般に，攻撃者は端末の通信情報を分析して接続先サーバを特定し拡散することが多い．この方式を用いると攻撃者が正規サーバへの通信と囮サーバへの通信とを分別するのに要する時間の分だけ攻撃の進行が遅くなることが期待される．提案する拡散抑制方式は文献 [31] と同様にダミーの TCP 接続情報を使用するが，これを活用して TCT マルウェアの拡散を抑制する仕組みを実現している点が異なる．さらに本稿では仕組みを導入する端末の選択方法についても検討し，シミュレーションおよび実機評価で有効性を確認している．また，文献 [32] では端末上に偽の NIC を作成し，この NIC を通じてアドレススキャンを行ったり，C2 サーバと通信などを行ったりするマルウェアを検知する方式を提案している．

その他，提案方式に関連した研究として先述の文献 [25] や存在しない端末への ARP リクエストに対し偽 ARP リプライを返すことで攻撃者をミスリードする方式 [33] など [34] がある．

デセプションと同様に注目されているプロアクティブディフェンス技術として Moving Target Defense（MTD）がある．MTD はシステムの複雑性や多様性を上げることで，攻撃者の侵入を妨害する [36]．先述の文献 [15] は Hit-list 型マルウェアに対する MTD 技術と位置づけられる．

2.3 NotPetya の概要

NotPetya は 2017 年 7 月にロシアからウクライナに向けた標的型攻撃キャンペーンで使用されたマルウェアである [3]．キャンペーンでは，ウクライナの IT ベンダが提供するソフトウェアのアップデート機能を悪用して NotPetya をエンタプライズネットワークに送り込む [43]．侵入に成功した NotPetya は，様々な拡散方法を用いてネットワーク中の Windows 端末に拡散・感染し，システム停止や破壊などの工作を行う．このように巧妙に設計された攻撃を完全に事前防御することは難しく，事後対処が重要になる．本研究では，事後対処の中でも特に拡散活動の特徴に着目することで早期検知・拡散抑制を行う．

NotPetya は SMB の脆弱性 MS17-010 を悪用して拡散を広げる．2020 年現在，MS17-010 自体への対策は進んでいる．しかし，多くの Windows 端末上で動作しているサービス上に同様の脆弱性が新たに発見し NotPetya のような拡散方法をとるマルウェアに悪用された場合，エンタプライズネットワークに大きな被害を及ぼす恐れがある．このため，本稿で検討する脆弱性のシグネチャや詳細情報に依存しない検知・抑制手法は，将来の脅威対策の 1 つとして有用と考える．

NotPetya の特徴の 1 つは従来のマルウェアと比べて多様な拡散方法を持つことにある．具体的には，(i) TCP Connection Table の参照，(ii) 同一サブネット内端末を対象とするローカルスキャン（ARP スキャン），(iii) NetServerEnum による Windows ドメイン内サーバの列挙や DHCP アドレスリストの悪用，(iv) ARP キャッシュにある IP アドレスの参照などがある．一方で，WannaCry と異なり同一サブネットを超えた IP アドレススキャンは行わない．このため，同一サブネット・Windows ドメインをまたがりエンタプライズネットワーク内に効率的な拡散を行うには (i) が重要となる．

検知側からの観点では，(ii) の ARP スキャン速度は WannaCry の 1/30 と低速であり [44]，(i)，(iii)，(iv) では通信接続失敗が発生する確度は従来の IP アドレススキャン型のマルウェアに比べて低いことから，通信パターンに基づく検知は相対的に難しいといえる．従来技術による検知難易度の評価については 7 章について述べる．

NotPetya このような拡散方法をとる理由としては，従来の IP アドレススキャンと同等の拡散速度を維持しつつ，検知を回避できる確度を上げるためと考えられる．本稿では基本的に (i) の拡散方法にフォーカスし，TCT マルウェアとしての拡散特性について議論するが，7.3 節で述べる NotPetya の検知難易度の観点からは総合的に評価する．

3. TCT マルウェアの拡散モデル

本章では TCT マルウェアの拡散モデルについて述べる．2 章で述べたとおり従来のマルウェア拡散モデルの多くは IP アドレススキャンを対象としており，著者らが知る限り，TCT を悪用して拡散を広げるマルウェアのモデルは検討されていない．本稿は主に Windows OS に感染するマルウェアを対象とするが，Linux など他 OS に対しても本拡散モデルは適用できる．また TCT の各エントリには接続先ピアと自端末・ピアの IP アドレスが含まれるものとする．

表 1 記号・用語
Table 1 Notations and terms.

記号・用語	定義
T_S	TCT の参照・拡散を終了したマルウエアが，次に TCT を参照するまでのインターバル時間
T_R	拡散を開始したマルウエアが，TCT の参照を行う期間
T_W	マルウエアが侵入を完了してから拡散活動を開始するまでのインターバル時間
R_P	マルウエアが，単位時間あたりに，拡散のためにアクセスするピア数
Hub	ネットワーク中の端末で，一定期間内に inbound 接続を受けたペア数が相対的に多いもの
Heavy-Hitter	ネットワーク中の端末で，一定期間内に outbound 接続を行ったピア数が相対的に多いもの

以降の議論で用いる，記号・用語の一覧を**表 1** に示す．

マルウェアが拡散先の選択に悪用する TCT には．rhosts や.known_hosts などの静的リストと比較して以下の 2 つの特徴がある．

1. 時間経過とともに新規 TCP 接続の開始や既存 TCP 接続の終了が発生するたびに，TCT 内のエントリは動的に変動する．
2. TCT には自端末から他端末への outbound 接続と他端末から自端末への inbound 接続という，接続方向が異なる 2 種類のエントリが存在する．

本モデルでは，上記の特徴に着目し，以下のとおりマルウェアの挙動を定義する．

1. マルウェアは侵入から一定期間（T_W）の後に拡散先端末への感染を完了する．
2. マルウェアは端末への感染完了後，Windows 標準の API（GetExtendedTcpTable [35] など）や標準ネットワークコマンド（netstat.exe など）を用い TCT を 1 度，もしくは一定時間ごとに複数回にわたり参照し，各 TCP 接続のピア端末の IP アドレス一覧を取得する．
3. 次いで，IP アドレス一覧内の IP アドレスのうち拡散先選択基準を満たすピアに対して outbound TCP 接

表 2 TCT マルウェアの分類
Table 2 Classification of TCT malware.

		拡散先選択基準		
		全 TCP 接続	Outbound 接続	Inbound 接続
TCT 参照回数	1 回	(1) all-once 系	(3) out-once 系	(5) in-once 系
	複数回	(2) all-repeat 系	(4) out-repeat 系	(6) in-repeat 系

続による拡散を一定の速度（R_P）で試行する．

本モデルでは TCT 参照回数および拡散先選択基準の観点から TCT マルウェアを**表 2** に示す 6 種類に分類する．

拡散先選択基準の観点では outbound/inbound 接続両方の接続ピアを拡散対象とする all 系と，outbound 接続のピアのみを対象とする out 系，inbound 接続のピアのみを拡散対象とする in 系とに分類される．感染先の数と感染速度の点では all 系のほうが out 系より優位と考えられる．一方で，ファイアーウォールなどで TCP 接続方向をふまえた IP アドレスベースの通信アクセス制御を実施しているネットワークでは，マルウェアが inbound 接続中のピアに対して outbound 接続を試行するタイミングで検知・遮断される可能性がある．反対に out 系は拡散規模では all 系より小さくなる一方，すでに outbound 接続が確立しているピアのみに拡散するため，前述のアクセス制御を回避できる可能性が高くなる．一方で，in 系は all 系よりも拡散速度が低く，検知・遮断されるリスクはほぼ同じことから，マルウェアの拡散戦略として採用される利点は低いと推察される．

TCT 参照回数の観点では，TCT の参照と拡散が 1 回のみである once 系と，決められた時間（T_R）が過ぎるまで参照と拡散を一定間隔（T_S）で繰り返す repeat 系とに分類される．先述のとおり TCT のエントリは動的に変動するため，TCT を何度も参照する repeat 系のほうが once 系と比べて多数のピアに拡散できる．一方で repeat 系は T_R の間，感染端末が通信を行うための OS 機能を維持する必要がある．このためマルウェアの目的によっては T_R の長大化は望ましくない場合もある．たとえば破壊型マルウェアの目的は感染端末の OS 機能やシステムファイルを破壊して業務を妨害することであるため，T_R はたかだか数十分〜数時間程度に設計されていると考えるのが自然である．実在のマルウェアである NotPetya，BadRabbit，Moonraker Petya は all-repeat 系に属する．たとえば NotPetya は数十分から 1 時間程度の間，約 3 分ごとに TCT の参照と拡散を繰り返し，その後 OS を破壊し目的を達成する [3]．

以降では，マルウェアが採用する利点の観点から，表 1 の (1)–(4) のモデルを中心に議論を進める．

リスト 1 all-repeat 系 TCT マルウェアの疑似コード
List 1 Pseudocode of TCT malware of all-repeat type.

```
1   Begin
2   conduct Initialization for T_W
3   until T_R elapses:
4           attack_list={}
5           foreach outbound entry in TCT:
6                   add entry's peer to attack_list
7           foreach inbound entry in TCT:
8                   add entry's peer to attack_list
9           attack entries in attack_list with a rate of R_P
10          sleep for T_S
11  achieve the objective // (e.g., system/file destruction)
12  End
```

リスト 1 に all-repeat 系拡散の疑似コードを示す．line 2 で事前準備を終えたマルウェアは line 4–10 をループし，他端末に拡散する．そして T_R 経過後に line 11 にある最終目的を実行する．once 系は line 4–10 の実行は 1 度のみ行う．また out 系では line 7–8 は実行されない．

なお，本稿では TCT 拡散の特徴にかかわりが深い拡散先選択基準・TCT 参照回数をモデルの主要パラメータとして扱うが，そのほかにも通信プロトコル，サーバ側ポート番号や拡散施行回数の上限値など様々な観点が考えられる．モデルの細分化・詳細化については今後の課題とする．また，UDP を悪用した拡散は対象外とするが，TCP と同様の議論ができると考える．なお，7 章では具体的脅威のケーススタディとして SMB を悪用した拡散を特に扱う．

4. 拡散特性の評価

本章ではシミュレーションにより TCT マルウェアの拡散特性と検知の難易度を評価する．さらに拡散速度を IP アドレススキャン型マルウェアと比較する．

4.1 評価用データセット

本評価では，長期間にわたり記録された組織内ネットワーク端末の TCT 情報のログを基に Windows 端末を狙った all/out 系，once/repeat 系の 4 種類の TCT マルウェアの拡散をシミュレーションする．なお，シミュレーションでは，TCT マルウェアの拡散に適した環境における被害規模の評価および提案技術の効果検証を行うことを目的に，(i) 拡散対象ネットワークに存在する端末のほとんどが Windows であること，(ii) 評価対象の全端末が脆弱性を有すること，(iii) 提案技術以外に，拡散の検知・抑止を行うセキュリティコントロールがないこと，を前提条件する．

個々の端末の TCT 情報のログは LANL データセット [4] を基に作成する．LANL データセットは Network Event Data と Host Event Data から構成される [4]．Network

Event Data にはロスアラモス研究所のエンタプライズネットワーク内の端末・ネットワーク機器間の通信情報が 89 日分（day02～day90）含まれている．初日（day02）および最終日（day90）には不完全なエントリが含まれている可能性があるため本評価では day03～day89 の 87 日間を使用する．

Network Event Data の各エントリには通信の開始時刻，通信の持続時間，通信開始端末および通信先端末の ID，通信プロトコル，両端末のポート番号が含まれている．本評価ではこのエントリを基に，ある時刻にある端末に侵入した TCT マルウェアが TCT から取得できる通信情報を推定して，拡散の広がりをシミュレーションする．

Host Event Data には Windows 端末内の監査ログが含まれる．本評価では Host Event Data を基に特定した Windows 端末間の TCP 通信情報を Network Event Data から抽出した．その結果得られた，Windows 端末 10,667 台間で発生した 1,354,771,369 件の TCP 通信情報を分析対象とする．

LANL データセットの中から Host Event Data を基に Windows 端末であることが確実である端末群を抽出して，端末群および端群間の TCP 通信を評価対象データセットとすることで，上記前提条件の (i) が成立する．なお，Network Event Data には各端末の OS 種類を示す項目はなく，Host Event Data に含まれない端末が非 Windows 端末であるとは限らない．一方，LANL データセットからは，前提条件 (ii)，(iii) が確実に成立するかは判断できないが，実験中の仮定として前提が成り立つものとした．なお，(i)，(ii) が成り立たず，感染した Windows 端末が非 Windows 端末または脆弱性を有しない端末と通信を行っていた場合，あるいは (iii) が成り立たずマルウェアの通信が遮断された場合は，当該端末への拡散試行が失敗することで発生する RST パケットなどを基に TCT マルウェアを検知できる可能性がある．この点については 7 章で議論する．

なお LANL データセットにはいくつかの注意事項がある．まず，通信情報は組織内ネットワークのコアルータ群の netflow 情報を基に作成されたものであるため，同一サブネット内に閉じた通信は記録されていない．また通信発生時刻や持続時間についても実際からのズレがある可能性がある．特に持続時間が実際より長く見積もられた通信情報によりマルウェアの拡散能力が過大評価される可能性がある．これを避けるため本評価では day N に発生した通信はデータセット上では day N+1 以降も持続している場合であっても day N の 23:59:59 に終了したと見なすことにした．またセキュリティ上の理由から各端末の IP アドレスは非公開であり，その代わりに独自の識別番号が ID として付与されている（例：Comp492856）．このため組織内ネットワークのネットワーク構成や IP アドレス空間の使用実態は不明である．さらに少数ながら複数の端末に対し同一 ID が付与されている場合がありうる．

以上より LANL データセットを用いた評価結果には実際の環境と比べて多少の差異がある可能性が高い．しかし，約 3 カ月にわたる 1 万台以上の組織内端末間の通信情報が含まれる，大規模かつ比較的新しい公開データセットは他には見あたらない．このため我々は TCT マルウェアの特性を分析するためのベースライン評価対象として LANL データセットの使用は十分に妥当であると考える．

なお，本実験では端末間の RTT やパケット再送などにより生じる遅延は検討の対象外とする．一般的に，エンタプライズネットワーク内では RTT は数 ms 程度であり，秒単位でのシミュレーションからは無視してよいと判断した．パケット再送についても LANL データセットに記述がないことから，検討対象外とした．再送が発生する場合には数秒の通信遅延が生じ，評価に影響を与える可能性があるが，拡散傾向に大きな差異は発生しないと考える．

4.2 評価方法

本評価では day03～day89 の 9 時～17 時の間にランダムに選択された 1 端末が TCT マルウェアに感染し拡散が開始されるシナリオをシミュレーションする．シミュレーション試行回数は各日 15 回，計 1,305 回とした．

TCT マルウェアモデルのデフォルトパラメータ値としては NotPetya の挙動などを参考に，$T_S = 100\,s$, $T_R = 1800\,s$, $T_W = 30\,s$ を設定する．また，R_P のデフォルトパラメータ値としては $R_P = 10/s$ とした．各記号の定義は表 1 に示すとおりである．

端末への侵入方法としては MS17-010 などの脆弱性および Pass-the-hash などによる認証情報の窃取を想定し，攻撃は確実に成功するものと仮定する．またすでに感染済の端末が攻撃を再度受けた場合には再感染は起きないものとする．

また拡散速度の比較対象として IP アドレススキャンを行うマルウェアについてもシミュレートを行う．このマルウェアは自端末の IP アドレスの第 1 オクテットを固定してランダムローカルスキャンを行い（例：IP = a.b.c.d の場合，a.0.0.0～a.255.255.255 がスキャン対象），発見した端末に拡散する．これは，一般的に，1 万台以上の端末を擁する組織内ネットワークではクラス A のプライベートアドレス（10.0.0.0/8）が用いられることが多いと予想されるからである．拡散が完了するのに要する時間は T_W と同値とする．このモデルは WannaCry などが用いるローカルスキャンを一般化したものである．具体的な数値計算には Analytical Active Worm Propagation（AAWP）モデルを使用した [37]．

4.3 評価結果

本節では TCT 拡散とアドレススキャンによる拡散を比較するとともに，系の違いが拡散速度・検知難易度に及ぼす影響を評価する．

4.3.1 all 系と out 系の比較

図 1，**図 2** に all-once 系および out-once 系の拡散台数の変化をそれぞれ示す．なお，最初に感染した端末（初期端末）が最初の拡散を行った時刻を経過時間の起点としている．

まず all 系と out 系とで拡散台数には大きな違いがあることが分かる．all 系の最終的な拡散台数が平均 7,000 台近くであるのに対し out 系では 120 台程度にとどまっている．これは all 系では多数のクライアント端末から inbound 接続を受けているアプリケーションサーバや Active Directory サーバが hub となり初期拡散が加速されるためと考えられる．一方 out 系では多数の端末に接続する heavy-hitter に感染しない限り拡散の広がりは遅くなる．ただし all 系の場合も最終的な拡散台数は平均でネットワーク内の全端末の 70%程度に収束している．これには主に 2 つの理由が考えられる．1 つは，同一の組織ネットワーク内に通信を行う機会が乏しい互いに疎な端末群がいくつか存在するため，拡散がネットワーク全体に必ずしも行き渡らないことである．他には，端末はつねにオンラインであるとは限らずオフラインまたは停止中の端末には拡散されないことがあげられる．

また all 系と比べて out 系では初期端末の違いによって相対的なばらつきが大きい．たとえば平均値と比べたときの 90 パーセンタイル値の拡散台数の比率は，all 系では約 1.3 倍だが out 系では約 2.6 倍に達している．これは out 系では初期端末が heavy-hitter である場合と outbound 接続数が少ない端末である場合とで，初期拡散の広がりに大きな差が生じるためと考えられる．

図 3 に，all-once 系において R_P の値が拡散速度に及ぼす影響を示す．R_P の定義は表 1 に示すとおりである．デフォルトパラメータである $R_P = 10/s$ の 10 分の 1 の速度である $R_P = 1/s$ では拡散速度は数分の 1 になる．たとえば，$R_P = 10/s$ の場合に 6,000 台の端末が感染している経過時間において，$R_P = 1/s$ の場合の感染台数は半数の 3,000 台弱である．反対に，$R_P = 100/s$ のときは $R_P = 1/s$ の場合の数倍の速度で感染が広がる．$R_P = 1$ の場合であっても後述するアドレススキャン型と比べて TCT マルウェアの拡散速度は十分早く，組織内ネットワークでは依然として大きな脅威といえる．

4.3.2 通信パターンの不審性

本項では all 系と out 系での通信パターンの不審性について比較検討する．all 系は感染端末に inbound 接続を行っているピア端末に対して逆向きの接続（outbound 接続）を行うという特性は out 系と比べて不審性が高いと考えられる．

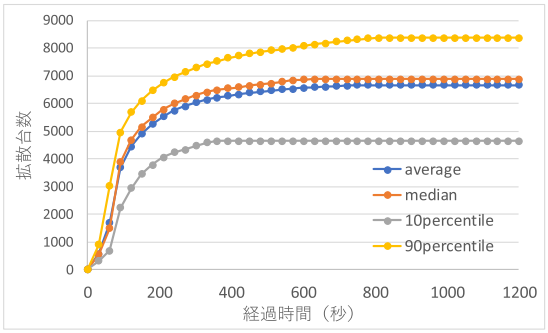

図 1 all-once 系の拡散台数
Fig. 1 Infected hosts by all-once type.

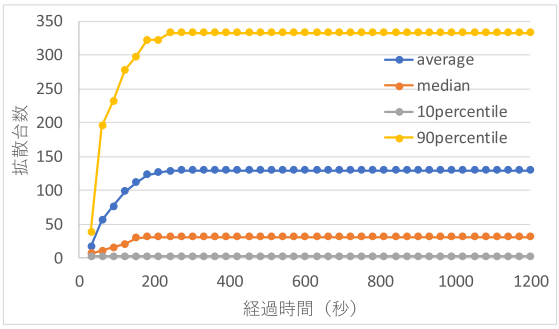

図 2 out-once 系の拡散台数
Fig. 2 Infected hosts by out-once type.

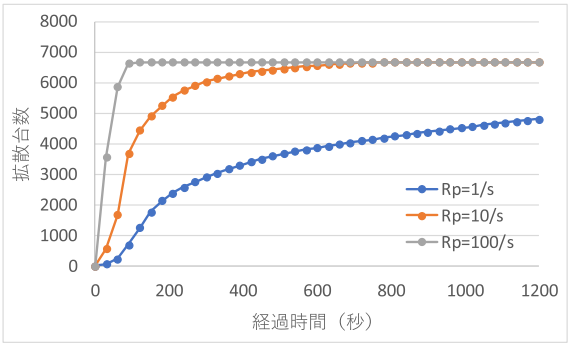

図 3 R_P の拡散速度への影響
Fig. 3 Effect of RP on propagation speed.

れる．

ここで，任意の TCP 接続を 2-タプル ⟨接続元端末, 接続先端末⟩ で表現することにする．マルウェアの拡散対象は 4.1 節で述べた LANL データセット内の Windows 端末であるため，データセットに含まれない接続元・接続先端末がマルウェア拡散にともない出現することはない．本評価では，本稿で扱う day03～day89 に含まれるすべての 2-タプルを要素として持つホワイトリストを作成し，これに含まれないタプルを未知タプルと呼ぶこととする．たとえば，ホワイトリスト内に端末 x から y へのタプル $\langle x, y \rangle$ が含まれるがその逆方向の接続は含まれない場合，マルウェア拡散にともない $\langle y, x \rangle$ が発生すると未知タプルと見なされる．すなわち，ホワイトリスト W を $W = \{\langle x, y \rangle \mid \langle x, y \rangle \in \text{day03}\sim\text{day89}\}$ と定義すると，マルウェア拡散により発生した TCP 接続の

2-タプル t は，t ∉ W であるとき未知タプルとなる．**表 3** に，シミュレーションの中で TCT マルウェア拡散で発生した 2-タプルのうち，未知タプルとなるものの出現件数を示す．なお，値は全シミュレーションの平均値である．

表中の all-once 系は，すでに拡散済みの端末に対しマルウェアが再接続を試行する/しないの観点から 2 パターンに分けられている．通常想定するのは再接続を試行するタイプである．たとえば，ある時刻に端末 A から端末 B に拡散したとする．通常のマルウェアでは，その後別の感染端末から端末 B に再接続する場合がありうる．一方再接続を試行しないマルウェアは，他の感染端末がいったん感染した端末 B に対しては別の感染端末から再接続しないよう制御されている．このような特性を持つマルウェアは Divide and Conquer 型 [38] と呼ばれ，拡散にともなう未知タプルの件数は再接続試行ありの場合より減少する．実現方法としては，感染済み端末のアドレスリストを特定のサーバ上などで共有するといった方法が考えられる．

out-once 系ではマルウェアの全 TCP 接続はデータセットに含まれるため，未知タプル件数は理論上 0 となる．一方 all-once では平均数千～1 万程度の新種タプルが出現する．このため過去一定期間中に発生したタプルをホワイトリストとして用いて all 系の不審性を測定し，検知に応用できる可能性がある．たとえばネットワーク全体で一定時間内に出現する未知タプル数 N が閾値 *TH* を超えたときにアラートを発する，といった方法が考えられる．

しかし，マルウェアが発生していないときでも，業務の変動や揺らぎなどによりホワイトリストにないタプルは定常的に発生しうる．よって閾値 TH が低すぎると大量の誤検知が発生する恐れがある．たとえば，仮に day03～day32 に発生したタプルの集合を S とした場合，day33～day89 において S に含まれないタプルは 1 日平均 1,814 件発生する．このため，ホワイトリスト作成に用いられるログの期間にも依存するが，誤検知を避けるために閾値 TH を大きな値（たとえば 2,000）にすると，その副作用として，all 系マルウェアを検知するまでに 1,000 台以上の端末が感染する可能性がある．一方クラス A のローカルアドレススキャン型マルウェアでは端末 100 台が感染するまでに理論上 15 万個程度の未知タプルが発生する．このため all 系であっても一般的なスキャン型に比べれば TCT マルウェアの不審性を測定し検知につなげることは難しいといえる．

また，ネットワーク全体としてマルウェアの不審性を測定して検知ができたとしても被害抑制のためには個々の端末の感染有無を突き止め隔離する必要があり，その作業の段階で拡散がさらに拡大する恐れがある．さらに先述のとおり接続タプルに基づく異常検知の方法で out 系を検知するのは非常に難しい．

4.3.3 ローカルランダムスキャンとの比較

図 4 および**図 5** に all-once 系および out-once 系とクラス A アドレス空間を対象としたローカルランダムスキャン型マルウェアとの拡散特性の比較をそれぞれ示す．なおランダムスキャン型の評価では，すべての端末がオンラインであることを前提としている．このため，スキャン速度は理論上の最高値となる．

all-once 系は経過時間 500 秒付近で拡散台数が収束するが，50 scan/s のランダムスキャン型マルウェアも同時刻に同程度の拡散台数を示す．このため all-once 型の収束までの拡散速度は 50 scan/s のランダムスキャン相当でありスキャン速度が約 20 scan/s である WannaCry より数倍速いといえる．同様に out-once 系の同時刻までの拡散台数は 20 scan/s のランダムスキャンと同程度である．

all-once, out-once ともに必ずしもネットワーク内の全端末に拡散するわけではないが，収束までの速度が数十 scan/s 程度のランダムスキャンに相当し，かつ検知が相対的に難しいことをふまえると，マルウェア拡散のブートス

表 3 拡散にともない発生する未知タプルの数
Table 3 Unprecedented tuples caused by malware propagation.

	同一拡散先への再接続試行	
	ありの場合	なしの場合
all-once 系	11,062	5,122
out-once 系	0	

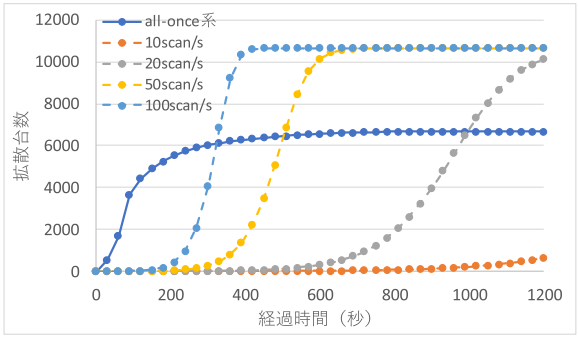

図 4 all-once 系とローカルアドレススキャンの比較
Fig. 4 all-once and local address scan.

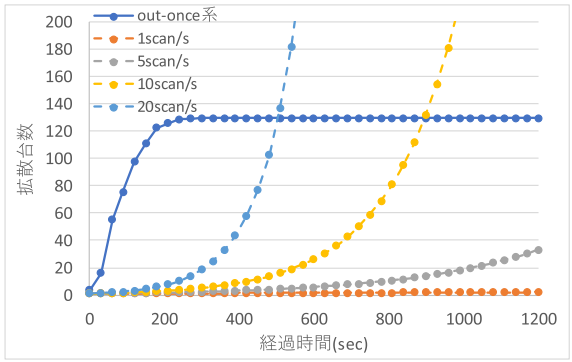

図 5 out-once 系とローカルアドレススキャンの比較
Fig. 5 out-once and local address scan.

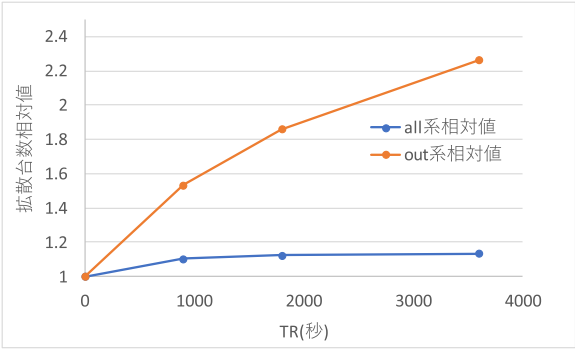

図 6 once 系と repeat 系との比較
Fig. 6 Comparison between once-type and repeat-type.

トラップ手段として大きな脅威といえる．

4.3.4 once 系と repeat 系の比較

図 6 に all/out-once 系と比べたときの all/out-repeat 系の拡散台数の相対値を示す．ここで述べる相対値は，拡散開始 3,600 秒経過時点での all/out-once 系の拡散台数に対する比率である．once 系は $T_R = 0$ と見なしている．

all 系，out 系ともに T_R の増加とともに拡散台数は着実に増えるが out 系のほうがより伸び率が高い．理由として all 系では all-once 系であっても全端末の 7 割に拡散するため all-repeat 系での伸び幅が狭くなることが考えられる．一方で out-once 系での拡散台数が約 120 台であった out 系では 2 倍以上に伸びており，一定時間拡散を繰り返すことの効果が高いことが分かる．

このため out-repeat 系は，拡散台数がネットワーク全体の数パーセント程度で十分な一方，検知を回避する必要性が高くかつ目的達成までに数十分程度要することが許容されるマルウェアにとっては最適な戦略といえる．

5. TCT マルウェアの拡散抑制方式

本章では前述の TCT マルウェアを検知・抑制する方式の提案とシミュレーション評価を行う．提案方式は，デセプションの考え方に基づき，感染端末内の TCT を参照するというマルウェアの挙動を逆手にとることを特徴とする．本章では方式のコンセプトと TCT マルウェアの拡散に対するシミュレーション上での効果について述べる．

5.1 方式のコンセプト

提案する拡散抑制方式では組織内ネットワークの各端末の TCT にダミーの TCP 接続情報を 1 件以上，何らかの手段で挿入する．ダミーの TCP 接続情報のピアアドレスとしては，ネットワーク内で未使用の IP アドレス（ダミー IP アドレス）を設定する．端末からダミー IP アドレスへの通信は正常運用時にはいっさい発生しないものとする．

本方式では，端末に侵入した TCT マルウェアは本物の接続情報とダミー接続情報との区別がつかず，その結果，マルウェアの拡散先は，本物とダミーの接続情報の中からランダムに選択されることを想定する．我々は，マルウェアが拡散先アドレスを分析する高度な機能を持たない限りこの前提は成り立ち，TCT マルウェアに本来存在しないアドレスへの拡散を試行させられると考える．

本方式ではダミー IP アドレスの挿入後，端末のパケット送受信を監視する．そしてダミー IP アドレスへの不正なパケット送信を検知すると，端末のその後のパケット送信を遮断することでマルウェアの拡散を抑制する．ダミー IP アドレスを各端末に配布する具体的手順は本稿の対象外とするが，一般には組織内ネットワーク内のアドレス管理サーバが業務で未使用のアドレスを配布することを想定する．

ここで，時刻 t において端末 h の TCT に記載されている接続情報に含まれるピア端末の集合を $\mathrm{TCT}(t,h)$ とする．本方式を採用していない場合，時刻 t_a に端末 h 上で拡散を開始した all-once 系 TCT マルウェアの拡散先端末数 $N_0(t_a,h)$ は $N_0(t_a,h) = \mathrm{TCT}(t_a,h)$ となる．一方で本方式に従い N_d 個のダミー宛先 IP アドレスへの接続が端末 h 上の TCT に挿入され不正なパケット送信が遮断される場合，検知・遮断までに，端末 h を通じて拡散可能なピア数の期待値 $N_1(t_a,h)$ は式 (1) のとおりとなる．

$$N_1(t_a,h) = \frac{TCT(T_a,h)}{N_d+1} \tag{1}$$

なお，本方式が導入された端末では通信の遮断により繰返しの拡散ができなくなるため once 系と repeat 系との拡散特性は同等となる．

先述のとおりダミー IP アドレスは正常運用では使用されるはずのないアドレスである．このため，4 章で議論した未知タプルに基づく検知と比べて，デセプション型の本方式では誤検知のリスクが減少すると期待できる．

次節では，本方式を具現化した拡散抑制モジュールを組織ネットワークに導入する方式について検討する．モジュールの機能設計および実装については次章で述べる．

5.2 拡散抑制モジュールの導入先端末の選択

数千台～数万台の端末を擁する組織ネットワークにおいてすべてのクライアントやサーバに対して拡散抑制モジュールを導入するのはコストが高い．このため何らかの基準によって一部の端末に対して先行的にモジュールを導入するのが現実的である．本節では導入先端末の選択方法として，(i) ランダム選択および (ii) 接続頻度選択の 2 つを検討する．

ランダム選択では，全端末からランダムに選ばれた R_d%の端末にモジュールを導入する．一方接続頻度選択では過去一定期間中の inbound 接続数および/または outbound 接続数が上位 R_d%内に位置する端末に対してのみモジュールを導入する．前章で述べたように TCT マルウェアはサーバなど接続頻度が多い hub や heavy-hitter を介して短期

間に拡散する傾向がある．このため hub や heavy-hitter に対して重点的に対策を適用することで，拡散を遅鈍化できることが期待できる．導入先を端末の outbound 接続数に従い選択する方式を heavy-hitter 選択，outbound および inbound 接続の合計数に従う方式を hub 選択と呼称する．

5.3 シミュレーション評価

本節では拡散抑制方式が TCT マルウェアの拡散をどの程度抑制できるかを，前章同様 LANL データセットを用いたシミュレーションで評価する．なお，接続頻度選択方式では day03～day09 の 7 日間での outbound/inbound 接続数を基に導入先端末を決定した．また，all 系マルウェアに対しては hub 選択を，out 系に対しては heavy-hitter 選択を適用した結果を示す．

図 7 および**図 8** にランダム選択および接続頻度選択を適用した場合の TCT マルウェアの拡散台数（3,600 秒経過時）を，未適用時との相対値としてそれぞれ示す．なお，初期感染端末にはモジュールは導入されてないものとする．図より，ランダム選択では R_d を高くしても拡散台数の低下は緩やかであることが分かる．$N_d = 100$ のとき $R_d = 50\%$，すなわち 2 台に 1 台に方式を適用した場合でも all-once 系マルウェアの感染台数は 30%程度しか削減されていない．一方接続頻度選択では，all-once 系および out-once 系における感染台数は $R_d = 0.1\%$の時点で 60%，$R_d = 10\%$では 80%以上削減されている．以上より hub あるいは heavy-hitter に対して重点的に方式を導入することでランダム選択に比べて大幅に拡散を抑制できることが分かる．

また接続頻度選択では $R_d = 1$–10%のとき，$N_d = 100$ のときの拡散台数相対値は $N_d = 5$ の場合と比べて 0.2–0.3 程度減少しているのに対し，ランダム選択では大きな違いはない．これは接続頻度選択では多数のコネクションをともなう端末を優先して選択するため，N_d が $N_1(t_a, h)$ の大小に及ぼす影響が大きいためである．

クラス A 相当の組織内ネットワークにおいて 100 個程度の IP アドレスをダミー用に確保するのは一般的に難しくないと考えられる．このため，N_d の上限値は端末側の負荷やユーザビリティに与える影響に依存する．たとえば，端末負荷の点では，ダミー IP アドレスに対する接続が増えるほど，OS が管理する TCB（TCP Control Block）の数が増え，メモリや CPU リソース消費量が増加する．ただし，TCB のサイズはたかだか 1 KB 未満であるため，$N_d = 100$ 程度では大きな影響はないと考えられる．一方，ユーザビリティの観点では，netstat.exe などで端末の通信状況を確認したい場合に，ダミー IP アドレスに対する接続が多数表示されると，煩わしさを感じる可能性がある．

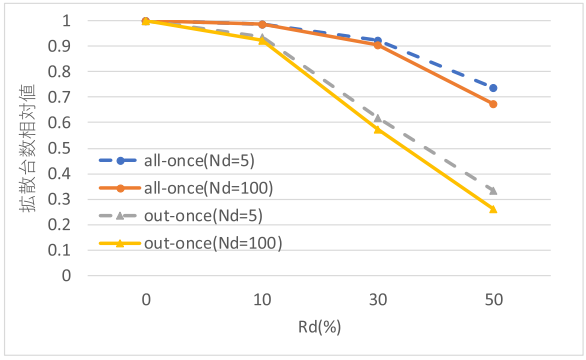

図 7 ランダム選択の性能
Fig. 7 Performance of random selection.

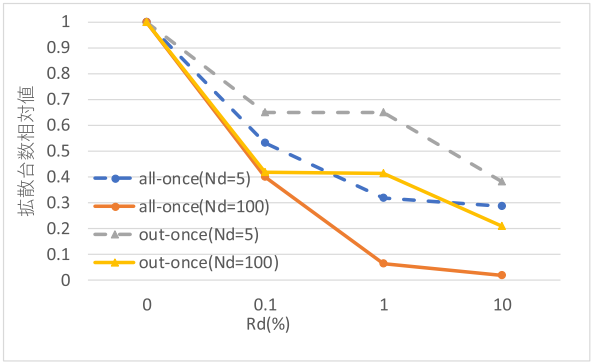

図 8 接続頻度選択の性能
Fig. 8 Performance of connection frequency based selection.

6. 拡散抑制モジュールの機能設計

本章では，拡散抑制モジュールの機能設計について述べる．**図 9** に方式を具現化した拡散抑制モジュールの概要を示す．モジュールはダミークライアントと拡散抑制ドライバによって構成される．

ダミークライアントはダミーの TCP 接続を行うプログラムである．接続先のダミー IP アドレスおよびポート番号は拡散抑制ドライバが保持するダミー接続プロファイルを参照して取得する．拡散抑制ドライバはアプリケーションレイヤから端末外への送信パケットを捕捉して必要な処理を施す．拡散抑制モジュールの要件を以下のとおりに定めた．

1. 端末がマルウェアに感染していない限り，正規アプリケーションの TCP 接続を阻害しない．
2. 標的型攻撃者が多用する Windows 標準コマンド（net.exe，netstat.exe など）[22] からみて，端末内の一般的なアプリケーションとダミー IP アドレスとの間で TCP 接続が確立されており，プロセス ID などの点でも整合しているように見せかける．
3. TCT マルウェアによるダミー IP アドレスへの通信を検知・遮断する．

要件 1 を満たすために，判定モジュールは送信パケットの宛先アドレスとダミー IP アドレスとを突合し，ダミー宛ではないパケットは端末がマルウェアに感染していない

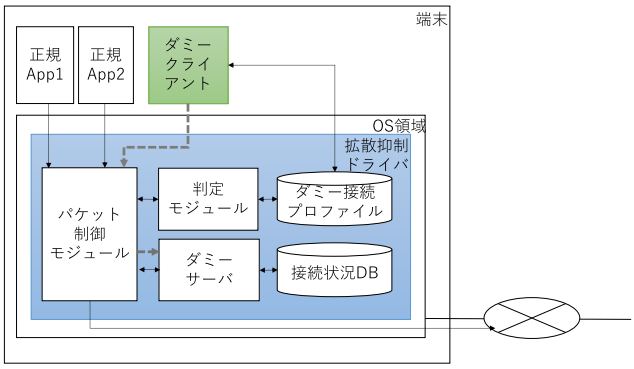

図 9 拡散抑制モジュール
Fig. 9 Propagation limiter.

限りすべてそのまま通過させる．

要件 2 を満たすために，拡散抑制ドライバはダミークライアントに対応するダミーサーバおよび接続状況 DB を備える．ダミー IP アドレス宛ての TCP パケットはすべてパケット制御モジュールを通じてダミーサーバにフォワードされる．ダミーサーバは通信状況を記録している接続状況 DB を参照し，ダミークライアントに対して TCP 3-way handshake およびその後通信の継続に必要なパケットの送受信を行う．このため，ダミークライアントおよび OS からみると正常に TCP 接続が確立しているようにみえる．

また，プロセスと通信情報の整合性を保つことやマルウェアや攻撃者の警戒を避けるために，ダミークライアントとしては独自の通信プログラムを開発するのではなく，net.exe (SMB)，telnet.exe (Telnet)，iexplore.exe (HTTP) など，OS に標準搭載された実行ファイルを用いる方針とする．

最後に要件 3 を満たすために，ダミークライアント以外のプログラムからのダミー IP アドレス宛ての通信を検知すると，拡散抑制ドライバは端末から外部への TCP 通信を遮断する．

ダミークライアントとダミーサーバはあらかじめ決められたプロトコル（ダミープロトコル）に従いパケットの送受信を行う．ダミープロトコルは，ダミークライアントとして用いるプログラムが用いるプロトコルや実際のサーバとの通信を分析することで設計する．プロトコルの制約および現在の通信状況は接続状況 DB 上で管理される．ダミーサーバはアプリケーションレイヤからダミープロトコルに従ったパケットを受信するとダミークライアントからの通信と判断し，対応するレスポンスを返す．これにより netstat.exe や Windows 標準 API などからは，端末とダミー IP アドレスとの間で outbound TCP 接続が確立しているようにみえる．ダミープロトコルは TCP 接続の継続の目的としているため通信頻度やペイロード量は最小限に抑えられる．

TCT マルウェアの通信は攻撃を目的としているため，その内容や手順はダミープロトコルとは必然的に異なる．そこでダミー IP アドレス宛のパケットがダミープロトコルに従わない場合に TCT マルウェアの存在を検知し，その後の全通信を遮断する．たとえば，あるダミープロトコル上では，3-way handshake 確立後は keep-alive のみを行い，100 byte 以上のペイロードを含むパケットを送受信しないと定められていたとする．この場合，3-way handshake 確立後の接続上でダミー IP アドレスに対して 100 byte 以上のペイロードを含むパケットが送信された時点で TCT マルウェアが検知される．

リスト 2 拡散抑制ドライバーの疑似レコード
List 2 Pseudocode of propagation limiter driver.

```
1   Begin
2   set comm_flag to True
3   While True:
4       receive a packet p from application layer
5       if p.dest not listed in dummy IP addresses:
6           if comm_flag=True:
7               send p to the remote host
8           else:
9               drop p
10      else:
11          if p confirms to a dummy protocol:
12              reply a response r to application layer
13          else:
14              drop p
15              set comm_flag to False
16  End
```

上述した拡散抑制ドライバの処理手順をまとめるとリスト 2 に示すとおりとなる．上記要件 1，2，3 に対しては line 5–7，line 11–12 および line 14–15 がそれぞれ対応する．

なお，拡散抑制モジュールの実現形態としてはこのほかにも，API フックを用いて getExtendedTCPTable などの実行結果を改ざんする方法や，実際にダミー IP アドレス宛ての通信をやりとりする組織内ネットワークにサーバを設置するという方法も考えられる．前者はプロセス ID や通信情報の整合をとるために，全プロセスで複数の API をフックする必要があり，複雑性や安定性の点に課題がある（たとえば，あるプロセスが kill された場合，当該プロセスが行っていた通信情報を TCT から削除する必要がある）．後者には，検知からネットワークスイッチなどを用いた遮断までにタイムラグがあり，その間の拡散を許してしまうという課題がある．より最適な実現形態に関しては引き続き検討を行っていく．

本章で設計した拡散抑制モジュールの具体的実装例については 7 章で述べる．

7. 拡散抑制モジュールの実装と評価

7.1 実装

本節では Windows 上での拡散抑制モジュールの実装について述べる．本実装では，ダミークライアントとして net.exe を用い，ダミー IP アドレスに対する接続を試行する．より具体的には，ダミー IP アドレスごとにコマンド「net use ¥¥dummy_ip_address」を実行する．net.exe は SMB に従いパケットを送信するため，SMB 接続を維持するにはダミーサーバ側で適切なレスポンスパケットを返信する必要がある．我々は net.exe が送信するパケットのパターンを解析し，**図 10** に示すダミープロトコルを策定した．TCP 3way-handshake が確立後，クライアントは Negotiate，Session Setup，Tree Connect の順で SMB レベルでのセッション確立に向けたパケットを送信し，セッション確立後は定期的に keep-alive パケットを送信する．ダミーサーバはこのプロトコルに従いレスポンスを返すことで本来のサーバとの間で通信が行われていると net.exe に対して見せかけることができる．反対に，ダミープロトコルに従わないパケットが送信されると TCT マルウェアの発生が検知され，以降のパケット送信は遮断される．

モジュールのプロトタイプ開発には Python 3.6 を使用した．アプリケーションレイヤからのパケットの捕捉とダミーサーバ上でのレスポンスパケット作成には PyDivert ライブラリ [39] を用いた．

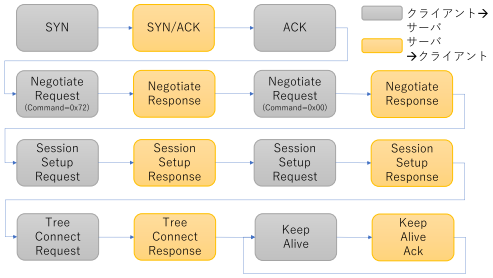

図 10 net.exe とのダミープロトコル
Fig. 10 Dummy protocol with net.exe.

表 4 NotPetya の実行結果の抜粋
Table 4 Output of notpetya.

#	マルウエア実行からの経過(秒)	発生事象(“dropped”は，拡散抑制モジュールが SYN パケットを遮断したことを意味する)
1	5.41	192.168.11.10 → 192.168.10.0:445, ダミーIP アドレスにアクセスがあったため拡散抑制モジュールにより検知・抑制開始
2	5.41	192.168.11.10 → 192.168.10.2:445 dropped
3	5.42	192.168.11.10→(External IP-1):445 dropped
4	5.43	192.168.11.10→192.168.10.1:445, drooped
5	8.13	192.168.11.10→(Internal IP-2):445 dropped
6	14.04	192.168.11.10→(Internal IP-3):445 dropped
7	34.64	192.168.11.10→(Internal IP-4):445 dropped
8	41.93	192.168.11.10→192.168.10.4:445 dropped
9	46.84	192.168.11.10→(Internal IP-5):445 dropped
10	51.36	192.168.11.10→(Internal IP-6):445 dropped
11	55.86	192.168.11.10→(Internal IP-7):445 dropped
12	60.56	192.168.11.10→(Internal IP-8):445 dropped
13	122.00	192.168.11.10→(External IP-2):445 dropped
14	122.10	192.168.11.10→(External IP-3):445 dropped
15	122.21	192.168.11.10→(External IP-4):445 dropped
16	162.66	192.168.11.10→(External IP-5):445 dropped

7.2 実機評価

実機評価では，実際に運用中のネットワーク内の端末 1 台に拡散抑制モジュールを導入し，NotPetya に感染させた場合の検知・抑制状況を検証する．本環境では，拡散抑制モジュールの動作不具合があった場合などに，マルウェア実行端末から他端末に実被害が及ぶことを防ぐために，端末間のスイッチ上でパケットフィルタリングを行い，攻撃ペイロードを含むパケットを遮断するなどの安全策を講じている．また，実験開始時における当該端末の接続先ピアの中には，実際には MS17-010 の脆弱性を持たないものも含まれている可能性がある．しかし，本環境下でも，マルウェアが発出する TCP パケットを基に，(a) 拡散抑制モジュールが NotPetya へのデセプションに成功するか否か，(b) 検知・遮断の完了までにマルウェアが接続試行するピア端末数，は測定することができる．このため，4.1 節で示した前提条件 (i)–(iii) が成り立つ環境下での検知・抑制効果についても同様に検証できると考える．

本実験では，拡散抑制モジュールを，あるイントラネット（192.168.11.0/24）に設置され，書類作成や Web 閲覧に用いられている Windows 10 Home 端末（VirtualBox 上で動作，割当 CPU 数 = 2, 割当メモリ = 4 GB）1 台に導入した．ダミー IP アドレスには，サブネット外のアドレスである 192.168.10.0～192.168.10.4 の 5 IP を設定した．そして，拡散抑制モジュールを有効化した後，端末上で NotPetya を実行した．NotPetya は 3 章で述べた拡散モデルの中では all-repeat 系に属する．NotPetya の起動時，端末は HTTPS，HTTP などで 50 台のピアと接続していた．なお，当該イントラネットには AD サーバは設置されておらず，NotPetya の拡散手段は TCT，ARP Cache，ARP スキャンの悪用などに限られる．

表 4 に，端末上で NotPetya を実行した直後のモジュール上のログを示す．NotPetya は，サブネット外の実在端末（#3）より先にダミー IP アドレス 192.168.10.0 に対して接続を試行するが（#1），3-way handshake 後の送信パケットがダミープロトコルと異なるため以降のパケット送信は拡散抑制モジュールにより遮断された．結果として，サブネット内（#5–7，9–12）およびサブネット外（#3，13–16）の実在端末に拡散通信が発生する前に，拡散活動を抑止することに成功した．当該ネットワークで 4.1 節で示した前提条件 (i)–(iii) が満たされる場合でも，初期端末以外の他端末が被害を被ることなく検知・抑制を行うことが可能と考える．ただしタイミングやマルウェアの攻撃先選択基準によっては，ダミー IP アドレスより前に実在の通信先に

攻撃が及ぶ可能性もある．次章でも述べるとおり，被害をいっそう軽減させるために，配布するダミー IP アドレスを未使用アドレス帯からどう選択すべきかに関しては今後の課題である．

次に，拡散抑制モジュールが正常な通信に与える影響について述べる．SMB ファイル共有を使用してインターネット上のサーバから 100 MB ファイルをコピーしたところ，所要時間はモジュール実行前 79 秒，実行後約 80 秒となり大きな性能低下は認められなかった．このとき，モジュールの CPU 使用率は 1%以下であった．また，モジュールが動作する端末 1 台で業務を 5 日間実施した結果，ユーザの操作ミスなどが原因で誤検知が発生することはなかった．

7.3 他対策技術による検知難易度について

本節では Snort [45] を例に，他対策技術による NotPetya の検知可否について検証する．Snort（実験では ver.2.9.12 を使用）は，オープンソースの IDS であり，パケットシグネチャに基づく検知機能のほかに通信パータン分析に基づくポートスキャン検知機能 sfPortscan を持つ．sfPortscan には，High，Medium，Low という 3 段階の sense_level がある．文献 [46] によると，それぞれの検知基準は以下のとおりである．

- High：600 秒あたりに，200 回以上の outbound 接続を行う端末，もしくは 9 回以上，outbound 接続に失敗する端末（接続失敗は，接続先からの RST パケット，ICMP Destination Unreachable などで判断）
- Medium：90 秒あたりに 200 回以上の outbound 接続を行う端末，もしくは 14 回以上 outbound 接続に失敗する端末
- Low：60 秒あたりに 5 回以上 outbound 接続に失敗する端末

7.2 節で示した実験では，パケットフィルタリングなどの導入により，攻撃が成功しないようにしている．そこで，本節では，NotPetya 感染端末から脆弱な Windows 端末 1 台に対する，攻撃に成功した拡散通信のパケットを採取し，7.2 節と同等の状況で 4.1 節で示した前提条件 (i)–(iii) が成り立つ場合に観測されると考えられる通信ログ（pcap）を作成し評価を行った．また，比較対象として文献 [50] にある WannaCry 感染端末のパケットを用い評価に使用した．評価結果は**表 5** に示すとおりである．

NotPetya，WannaCry ともに，対応したシグネチャが Snort に設定されている場合は検知は可能である．しかしこれには，マルウェアが既知のものである必要がある．一方，sfPortscan では，多数の端末に高速なランダムアドレススキャンを行う WannaCry は，Medium，High において検知できる一方，その性質上，拡散対象端末数に限りがある（すなわち，無数の拡散先候補がある IP アドレススキャンとは異なり，候補数が限られる）NotPetya は本実験では検知ができなかった．この点から，NotPetya は WannaCry に比べて相対的に検知が難しいといえる．なお，NotPetya が行う ARP スキャンは sfPortscan の分析対象外であり検出されなかった．また，sense_level = Low において WannaCry を検知できなかったのは，RST パケットや ICMP Destination Unreachable が端末やネットワーク上でフィルタリングされる環境であったためと考えられる．

表 5 Snort による検知結果
Table 5 Detection result by snort.

		NotPetya	WannaCry
パケットシグネチャ		○	○
sfPortscan	High	×	○
	Medium	×	○
	Low	×	×

○：検知成功　×：検知失敗

sense_level = Medium，High に関しては，端末が感染時に 100 を超えるような多数のピアと接続しており，かつ NotPetya の拡散速度が 7.2 節で示した実験結果より高速に設計されていたなら，検知できる可能性はある．しかし一方で文献 [47], [48] に示すように Medium や High は誤検知が多いことが知られており，オペレーション上の課題がある．

一方，sense_level = Low の場合でも，4.1 節の前提条件 (i)，(ii) が成り立たず脆弱性を持たないピアが一定数以上あり，RST パケットが端末やネットワーク上で遮断されずに拡散元端末に返される状況下では NotPetya を検知できる可能性はある．しかし，Low の検知条件（60 秒間に 5 回の RST パケット）もネットワークによっては高感度すぎ，多くの誤検知が発生しうると考えられる．たとえば，CICIDS2017（端末 14 台から構成されるネットワークの通信データセット）[49] では，攻撃が存在しないことが保証されている約 8 時間の間に，RST パケットが原因で，端末間の通信から 16 件の誤検知が起きる．一方，当該データセットを分析した結果，内部端末以外のプライベート IP アドレスがダミー IP アドレスとして設定されている限り，拡散抑制モジュールでは誤検知は発生しない見込みである．

また，sfPortscan 全体の問題として，一定の時間幅（window）で発生した接続数やパケット数を基に検知を行うことがあげられる．閾値や時間幅の具体値が明らかな場合，TCT マルウェアあるいは NotPetya の拡散速度を調整し検知を潜り抜けることが容易である．一方，拡散抑制モジュールは時間幅（window）には依存しないため，拡散速度に影響を受けずに検知・抑制を行うことができる．

8. 考察

今後，組織ネットワークを対象としたマルウェアの拡散

は，従来の IP アドレススキャンにとどまらず TCT 拡散などを用いいっそう巧妙化していくと考えられる．こうした拡散による被害を抑制するには脆弱性対策や OS 更新に加え以下の対策を順々に進めていく必要がある．

1. 未使用 IP アドレス空間への通信の監視
2. 通常業務での発生頻度が低い SrcIP，DstIP 間の通信の異常検知
3. マルウェアの拡散手法を逆手にとったデセプション技術の導入

一般的な IP アドレススキャンに対しては上記 1. が，all 系 TCT マルウェアに対しては 2. および 3. が，out 系に対しては 3. が有効といえる．NotPetya は TCT 以外にも DHCP サーバや AD サーバ経由で端末リストを入手するという方法を併用している．防御策の 1 つとして，このようなリストに対してもあらかじめダミーの IP アドレスや識別子を混入しておくといったアプローチが考えられる．

本稿のシミュレーションで使用した LANL データセットは，含まれる端末数・観測期間の長さの点で本研究に適している一方で，観測ポイントや匿名化方法に課題がある．この点をふまえ今後は実環境での実験を進めていきたい．

本稿ではシミュレーションおよび実機上でデセプションに基づく拡散抑制方式が TCT マルウェアに対して有効であることを示した．一方でダミーの TCP 接続が本来業務に及ぼす影響については引き続き検証が必要である．拡散抑制モジュールのプロトタイプ実装では，端末や通信速度に及ぼす悪影響は小さいことが示された．今後はユーザが誤ってダミー接続先にアクセスするリスクや，netstat.exe や getExtendedTCPTable の出力を利用するプログラムなどが誤動作するリスクについて評価を行うとともに，ユーザインタフェースの観点などからの対策を検討する [40]．

また今後，攻撃者側がデセプション技術を回避する方法をマルウェアに実装していくことも考えられる．この場合，式 (1) が必ずしも成り立たない場合がありうる．たとえば，tcpdump などを用いトラヒックを解析し実在の端末と偽の端末を見分けるなどが想定される．これを防ぐにはネットワーク中に実在の端末を装った偽トラヒックを混入させるなど，複合的な対策 [41] が必要となると考えられる．また，ダミー接続先 IP アドレスの分布が業務で使用するアドレス帯から著しく乖離している場合，マルウェアに見分けられる恐れがあったり，検知・遮断が遅くなったりする可能性がある．今後はこうした攻撃への対策もあわせて検討していく．ダミーファイルの配布でも同様の課題が指摘されており，その解決手段を応用できると考える [42]．

あるいは，何らかの方法で拡散抑制モジュールの存在を認知した攻撃者やマルウェアが，端末をネットワークから隔離する DoS 攻撃を目的に，ダミーの接続先に意図的に通信を行うという場合も考えられる．この場合，マルウェアが検知され通信が抑制されている状態でも，業務推進上必要なサービスへのアクセスは許容するといった緩和策が必要になる．

9. おわりに

本稿では TCT を悪用して組織内ネットワークに拡散するマルウェアの特性を検証するとともに，デセプションの考え方に基づく拡散抑止方式を検討した．その結果，TCT マルウェアはクラス A アドレス空間を対象とした 20〜50 scan/s の IP アドレススキャンと同等の速度で拡散できること，一般的な異常検知による早期検知が難しい場合があること，TCT を繰り返し参照するタイプでは拡散台数が数十%以上増える，などの知見を得た．一方，組織内の 1%程度の hub や heavy-hitter 端末に対し拡散抑制方式を適用することで拡散速度を数十%以上制限できること，NotPetya に対しても方式が有効なことをシミュレーション・実機評価で確認した．

今後は実際のネットワークに拡散抑制方式を適用して提案方式が実業務に及ぼしうる課題について検討を進めていく．

参考文献

[1] Panda Security: #WannaCry Report, available from ⟨https://www.pandasecurity.com/mediacenter/src/uploads/2017/05/1705-Informe_WannaCry-v160-en.pdf⟩ (accessed 2020-03-22).

[2] マイクロソフトセキュリティ情報 MS17-010，入手先 ⟨https://docs.microsoft.com/ja-jp/security-updates/securitybulletins/2017/ms17-010⟩（参照 2020-03-22）．

[3] Cylance：Petya-Like ランサムウエアは厄介なワイパー，入手先 ⟨https://www.cylance.com/ja_jp/blog/jp-threat-spotlight-petya-like-ransomware-is-nasty-wiper.html⟩（参照 2020-03-22）．

[4] Turcotte, M.J.M., Kent, A.D. and Hash, C.: Unified Host and Network Data Set, *Data Science for Security Chapter 1*, pp.1–22 (2018).

[5] Kawaguchi, N.: Detection of Hit-list Worms based on Propagation Behavior, doctoral thesis (2008), available from ⟨http://iroha.scitech.lib.keio.ac.jp:8080/sigma/bitstream/handle/10721/2258/document.pdf?sequence=4⟩ (accessed 2020-03-22).

[6] Zou, C.C., Gong, W. and Towseley, D.: Codered Propagation modeling and Analysis, *Proc. 9th ACM Conference on Computer and Communication Conference* (2002).

[7] Zou, C.C., Towseley, D. and Gong, W.: On the performance of internet worm scanning strategies, *International Journal on Performance Evaluation*, Vol.63, No.7, pp.700–723 (2006).

[8] Zou, C.C., Towsley, D., Weibo, G. and Cai, S.: Routing worm: A fast selective attack worm based on ip address information, *Proc. Workshop on Principles of Advanced and Distributed Simulation*, pp.199–206 (2005).

[9] Chen, S. and Tang, Y.: Slowing down internet worms, *Proc. IEEE Int'l Conference on Distributed Computing Systems*, pp.312–319 (2004).

[10] Zou, C.C., Gao, L., Gong, W. and Towsley, D.: Monitoring and early warning for internet worms, *Proc. 10th*

ACM Symposium on Computer and Communication Security (2003).
[11] Staniford, S., Moore, D., Paxon, V. and Weaver, N.: The top speed of flash worms, *Proc. ACM 2004 Workshop on Rapid Malcode*, pp.33–42 (2004).
[12] Kawaguchi, N., Shigeno, H., Ueda, S., Shiozawa, H. and Okada, K.: ACTM: Anomaly Connection Tree Method for Detection of Silent Worms, *IPSJ Journal*, Vol.48, No,2, pp.614–623 (2007).
[13] McDaniel, P., Sen, S., Spatscheck, O. and Aiello, B.: Enterprise Security: A Community of Interest Based Approach, *NDSS Symposium* (2006).
[14] Security, H.: JIB-14-20199C: Destructive Malware, available from ⟨https://www.oodaloop.com/wp-content/uploads/2015/02/JIB-Destructive-Malware.pdf⟩ (accessed 2020-03-22).
[15] Antonatos, S., Akritidis, P., Markatos, E.P. and Anagnostakis, K.G.: Defending against hitlist worms using network address space randomization, *Proc. 2005 ACM Workshop on Rapid Malcode*, pp.30–40 (2005).
[16] Yu, W., Chellappan, S., Wand, X. and Xuan, D.: On defending peer-to-peer system based active worm attacks, *Proc. IEEE Globecom 2005*, pp.1757–1761 (2005).
[17] Mickens, J.W. and Noble, B.D.: Modeling epidemic spreading in mobile environments, *Proc. 4th ACM Workshop on Wireless Security*, pp.77–86 (2005).
[18] Eichin, M. and Rochlis, J.: With microscope and tweezers: An analysis of the internet virus of November 1998, *Proc. IEEE Computer Society Symposium on Security and Privacy* (1989).
[19] Schecher, S.E., Jung, J., Stockwell, W. and McLain, C.: Inoculating ssh against address-harvesting worms, *Proc. Network and Distributed System Security Symposium 2006* (2006).
[20] Cherepanov, A.: GreyEnergy A successor to BlackEnergy, available from ⟨https://www.welivesecurity.com/wp-content/uploads/2018/10/ESET_GreyEnergy.pdf⟩ (accessed 2020-03-22).
[21] PlugX, available from ⟨https://attack.mitre.org/software/S0013/⟩ (accessed 2020-03-22).
[22] JPCERT/CC：攻撃者が悪用する Windows コマンド（2015-12-02），入手先 ⟨https://blogs.jpcert.or.jp/ja/2015/12/wincommand.html⟩（参照 2020-03-22）.
[23] MITRE, ATT&CK, available from ⟨https://attack.mitre.org/⟩ (accessed 2020-03-22).
[24] Allot, BadRabbit Ransomware: Real-Time Report, available from ⟨https://www.allot.com/wp-content/uploads/Threat-Bulletin_BadRabbit_10-2017.pdf⟩ (accessed 2020-03-22).
[25] 稲場太郎，川口信隆，岡田謙一：ダミーアドレスからのコネクショントレースバックによるワーム早期抑制手法，情報処理学会論文誌，Vol.50, No.6, pp.1610–1621 (2009).
[26] Wang, C. and Lu, Z.: Cyber Deception: Overview and the Road Ahead, *IEEE Security & Privacy*, Vol.16, No.2, March/April 2018, pp.80–85 (2018).
[27] Fraunholz, D., Anton, S.D., Lipps, C., Reti, D., Krohmer, D., Pohl, F., Tammen, M. and Schotten, H.D.: Demystifying Deception Technology: A Survey, arXiv:1804.06196[cs.CR] (2018).
[28] Fraunholz, D., Krohmer, D., Anton, S.D. and Schotten, H.D.: Catch Me If You Can: Dynamic Concealment of Network Entities, *Proc. ACM MTD'18* (2018).
[29] Heckman, K.E., Stech, F.J., Schmoker, B.S. and Thomas, R.K.: Denial and Deception in Cyber Defense, *IEEE Computer*, Vol.48, No.4 (2015).
[30] Yuill, J., Zappe, M., Denning, D. and Feer, F.: Honeyfiles: Deceptive Files for Intrusion Detection, *Proc. 2004 IEEE Workshop on Information Assurance* (2004).
[31] Park, K., Woo, S., Moon, D. and Choi, H.: Secure Cyber Deception Architecture and Decoy Injection to Mitigate the Insider Threat, *Symmetry*, Vol.10, No.14 (2018).
[32] Rrushhi, J.L.: NIC displays to thwart malware attacks mounted from within the OS, *Computers & Security*, Vol.61, pp.59–71 (2016).
[33] 竹中 幹，高野祐輝，宮地充子：Deception ネットワークを構成するフレームワークの提案，情報処理学会研究報告，Vol.2019-DPS-178, No.5 (2019).
[34] 角丸貴洋，島 成佳，吉岡克成：組織ネットワークにおける内部攻撃に対する模擬的欺瞞方式，*Computer Security Symposium* (2014).
[35] GetExtendedTcpTable function, available from ⟨https://docs.microsoft.com/en-us/windows/desktop/api/iphlpapi/nf-iphlpapi-getextendedtcptable⟩ (accessed 2020-03-22).
[36] Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C. and Wang, X.S.: Moving Target Defense, Springer, DOI: 10.1007/978-1-4614-0977-9 (2011).
[37] Chen, Z. et al.: Modeling the spread of active worms, *Proc. IEEE Infocomm2003* (2003).
[38] Wu, J., Vangala, S., Gao, L. and Kwiat, K.: An Effective Architecture and Algorithm for Detecting Worms with Various Scan Techniques, *Proc. NDSS2004* (2004).
[39] Pydivert, available from ⟨https://pypi.org/project/pydivert/⟩ (accessed 2020-03-22).
[40] 青池 優，神薗雅紀，衛藤将史，松本倫子，吉田紀彦：欺瞞機構に伴う利便性低下を防止するためのファイル非表示化，情報処理学会コンピュータセキュリティシンポジウム 2019（CSS2019）予稿集 (2019).
[41] Bowen, B.M., Kemerlis, V.P., Prabhu, P., Keromytis, A.D. and Stolfo, S.J.: Automating the Injection of Believable Decoys to Detect Snooping, *Proc. ACM WiSec'10* (2010).
[42] Genc, Z.A., Lenzini, G. and Sgandurra, D.: On Deception-Based Protection Against Cryptographic Ransomware, *Proc. DIMVA2019* (2019).
[43] NotPetya 攻撃に関する 3 つの疑問，入手先 ⟨https://www.cybereason.co.jp/blog/malware/1643/⟩（参照 2020-03-22）.
[44] HIRT-PUB17010: Security Alert: Ransomware NotPetya (Petya Variant), available from ⟨http://www.hitachi.com/hirt/publications/hirt-pub17010/index.html⟩ (accessed 2020-03-22).
[45] Snort Network Intrusion Detection & Prevention System, available from ⟨http://www.snort.org/⟩ (accessed 2020-03-22).
[46] Vit Bukac, IDS System Evasion Techniques, available from ⟨https://is.muni.cz/th/d4739/MT_Bukac.pdf⟩ (accessed 2020-03-22).
[47] 小埜勇貴，有馬竜昭，永山聖希，吉田和幸：Snort とのログ比較による scan 攻撃検知システム検知結果の精度調査，情報処理学会九州支部火の国情報シンポジウム 2011 論文集 (Mar. 2012).
[48] available from ⟨https://www.snort.org/faq/readme-sfportscan⟩ (accessed 2020-03-22).
[49] Intrusion Detection Evaluation Dataset (CICIDS2017), available from ⟨https://www.unb.ca/cic/datasets/ids-2017.html⟩ (accessed 2020-03-22).
[50] Packet Capture of WannaCry 2.0 Scanning SMB Port 445, available from ⟨https://precisionsec.com/wannacry-pcap-smb-445/⟩ (accessed 2020-03-22).

川口 信隆 （正会員）

2008 年慶應義塾大学大学院理工学研究科後期博士課程修了．博士（工学）．現在，株式会社日立製作所にて，サイバーセキュリティ，情報セキュリティの研究開発に従事．慶應義塾大学非常勤講師（2019 年～），兵庫県立大学客員研究員（2020 年～）．IEEE 会員．CISSP．