修士論文 2008年度 (平成20年度)

通信の類似性に着目した
ネットワークインシデント検知手法

慶應義塾大学大学院 政策・メディア研究科

金井 瑛

修士論文要旨 2008 年度 (平成 20 年度)

通信の類似性に着目した ネットワークインシデント検知手法

論文要旨

悪意のあるプログラム (マルウェア) に感染したホストの活動はインターネット利用者にとって大きな脅威となっている．マルウェアに感染したホストの集合はボットネットと称されるネットワークを構築し，DDoS 攻撃や迷惑メールの送信などの組織的犯罪に利用される．既存のマルウェア対策手法の多くは，マルウェアの感染を予防する点に注目している．しかし，近年，攻撃者は新しい感染方法の開発とマルウェア活動の隠蔽化に注力しており，既存の手法を用いた感染の予防が難しくなっている．また，亜種の多量な発生により，各マルウェアの特徴を利用したシグネチャ型の検知手法は効果が期待できない．ボットネットによる脅威を低減するための手段として，ネットワーク上でマルウェアに感染してしまったホストを，亜種の発生に影響されることなく検知できる手法が必要となる．

本研究では，マルウェアの通信をモデル化し，未知のマルウェア亜種に対して効果的な検知を実現した．モデル化にあたり，新規に出現したマルウェアであっても，過去に出現したマルウェアの通信と類似性がある点に着目している．本モデルを用いることで新種のマルウェア固有の情報がなくても，ネットワーク上のマルウェアに感染したホストの検知が可能となる．本手法を評価するためにインターネット上のマルウェアを収集し，仮想マシンを用いてマルウェアの通信を自動的に収集する機構を構築した．本機構で得られたマルウェアの通信を用いて既存の手法と本手法の検知結果を比較し，マルウェアの通信モデルを用いた検知が未知のマルウェア亜種に対して有効な検知手法であることを示した．これにより，既存の手法では難しかったマルウェアへの対応が可能となり，インターネット上のボットネットによる脅威の低減が期待できる．

キーワード
1. インターネット，2. セキュリティ，3. ボットネット, 4. 通信モデル

慶應義塾大学大学院 政策・メディア研究科

金井 瑛

Abstract of Master's Thesis

Academic Year 2008

Network Incident Detection Method Based on Similarity of Communications

Abstract

Hosts that are infected with malicious software, or malware, have become a major threat for Internet users. Infected hosts gather up and form a network called botnet, which is used for brutal crimes such as making DDoS attacks and sending SPAM e-mails. Most of the existing anti-malware software focuses on " preventing " malware infection. However, since attackers started to develop new infection channels and trying to conceal malware activities, prevention using the existing methods has become difficult. In addition, because a large number of variants are developed every day, signature-based detection using characteristics of malware is not expected to function sufficiently. In order to decrease the threats from botnet, a method for " discovering " malware-infected hosts, without being bothered by variants, is necessary.

This research has modeled malware communications and developed an effective detection against malware, even if it is an unknown variant. For modeling, detection of newly developed malware was done by measuring similarities with known malware communications. With this model, discovery of malware-infected hosts, without having patterns specific to the malware, becomes possible. To evaluate this method, a system for automatically collecting malware communications utilizing virtual machines was developed. The thesis proves efficiency of variant detection using malware communication model by comparing the system and existing methods on communication data obtained from the system. With this method, threats from botnet, which was difficult to defend from using existing methods, is expected to decrease.

Keywords :
1. Internet, 2. Security, 3. Botnet, 4. Communication Model

Keio University , Graduate School of Media and Governance

Akira KANAI

目 次

図 目 次

表 目 次

第1章　研究概要

1.1　背景

近年，インターネット上で感染を広げる悪意のあるプログラム (マルウェア) への対応が急務とされている．従来のマルウェアは自律的に感染を拡大させるプログラムが大部分であったが，2000 年頃より感染に成功したノードを攻撃者が集中管理できるようにするマルウェアが急増している．攻撃者が管理できるマルウェアに感染したノードは**ボット**と呼ばれ，ボットの集合は**ボットネット**と呼ばれている．攻撃者はボットネットを用いて，迷惑メールの送信や感染ノード上に保存されているクレジットカード番号の収集など，金銭的利益を目的とした不正な活動を行っている．2000 年 2 月にはボットネットを利用した大規模な DDoS 攻撃が発生し，Amazon.com や Yahoo!などの Web サイトが数時間停止するという被害が発生した [1]．2006 年 5 月には攻撃者が大量のノードからクリック課金型の Web 広告を不正クリックし利益と得ているとの報告がされた．この大量のノードの操作にはボットネットが利用されている [2]．2005 年 10 月にオランダで発見されたボットネットは 10 万ものボットから構成されたという報告があった [3]．さらに，このボットネットは後の報告で，150 万以上ものボットから構成されていた事実が判明し，深刻な規模のボットネットとして関心を集めた．このボットネットを用いて攻撃者らは企業への脅迫や，ハッキング行為を行ったとの報告がある．ボットネットはノードの管理者とネットワーク管理者の両者の観点から大きな脅威になっていると言える．

不正な活動により利益を得ようとする攻撃者はボットの存在を隠蔽し，規模を拡大させるために高い技術力を導入していると言われている [4]．インターネットが社会の基盤となっている背景から，ボットネットの対策には多くの組織において研究や対策手法の開発が進められているだけではなく，国が主導となった計画も出現している．たとえば，日本ではボット感染予防推進事業が総務省・経済産業省と独立行政法人 情報処理推進機構の間で開始されている [5]．しかし，現在のボット対策技術は，従来の脅威の対策技術の延長となっている．ボットは亜種発生頻度の高さや，高度な隠蔽技術により従来の脅威とは異なる特徴があるため，既存の対策技術では十分な効果が期待できない．

1.2 本研究の目的

本研究の目的は、各ネットワーク内部のボットを検知・排除することで、ボットネットによる脅威を低減することである．そのため、本論文では、ボットの定期的な管理サーバへのアクセスや、ボットの活動に見られる特徴的なネットワーク通信に着目して、ネットワーク内部におけるノードのボット感染を検知する手法を提案した．

ネットワーク内のボットの対策には，内部のノードをボットネットに加入させない事前対策と，ボットが内部に存在する場合の事後対策として，ボットの検知と駆除が必要となる．しかし，ボットネットは攻撃や活動の種類を頻繁に変更するため，多くの既存の手法が用いているシグネチャ型の検知手法では十分な検知ができない．ボットの事前対策や検知が難しい理由の１つとして，マルウェアの亜種が短期間で大量に出現するというボットネットの特徴がある．現在，多くのマルウェア対策ソフトウェアでは脅威に対応する技術として，対象に関する特定の特徴を示すシグネチャを用いている．このため，ボットに対する特徴のアップデートが追いつかず，ボットの検知は対応が困難となっている．

本研究では新規に出現したマルウェアであっても各ボット動作の通信には類似性がある点に着目し，検知手法を提案した．通信の類似性を利用して，既知のボット動作から特徴のある通信の前後関係に基づいたノードの通信に着目したボットの通信モデルを作成する．通信モデルを用いるとボット亜種固有の情報を用いないネットワークベースのボット亜種の検知が可能となる．また，インターネット上のボットを収集し，仮想マシンを用いてボット通信を自動的に収集する機構を実現した．この機構で得られたボット通信の既存の手法との検知結果を比較し，ボットの通信モデルを用いた検知が未知のボット亜種に対して有効な検知手法であることを示す．

1.3 本論文の構成

本論文は全7章で構成される．第2 章では，ボットネットとボットの現状についてまとめる．第3章では，既存のボット対策技術と研究について述べ，それぞれの特徴を整理する．第4章においてボット通信を収集する機構に求められる要求と今回実現した環境について述べる．第5章では，ボットの通信モデル化についての提案と解説をする．第6章において，実際のボット通信にみられる通信のモデルを示し，モデル化によるマルウェアに感染したノードの検知についての評価をする．第7章では本論文を総括し，本研究分野における今後の展望について述べる．

第2章　ボットネットの脅威

本章では，本研究においてボットネット検知対策を検討するにあたりボットネットの概要ならびにボットネットを構成するボットの概要をまとめ，ボットネットの特徴を整理する．

2.1　ボットネットの概要

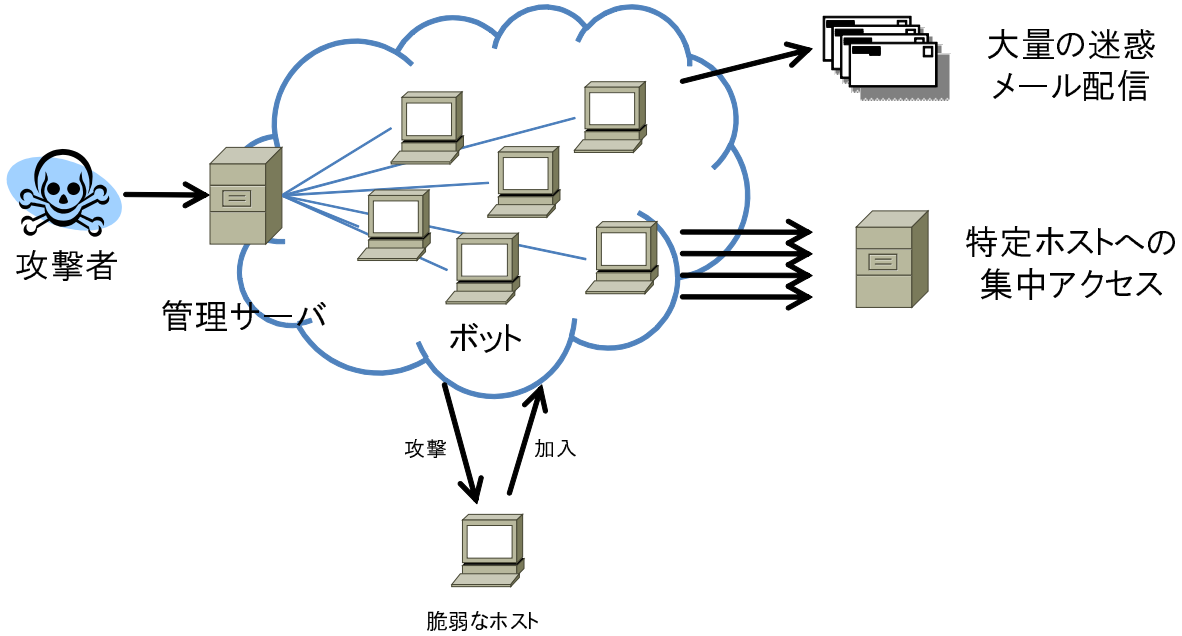

図 2.1: ボットネットの概要

ボットネットの概要について，図 2.1 を用いて示す．**攻撃者**は，ボットネットを管理する悪意を持ったインターネット利用者である．近年では，ボットネットを拡大するインターネット利用者や，ボットネットのボットを他人に貸し出すだけのインターネット利用者など役割が分割されているとされているが，本論文中ではボットネットを管理・利用するインターネット利用者を総じて攻撃者と称する．**ボットネット**は，ボットネット管理者により不正に操作されるノードであるボットの集合である．**ボット**は，マルウェアに感染し，ボットネットを構成するノードの１つである．ボットは管理サーバ

を介した攻撃者からの命令に従って，ノードの管理者が意図しない通信や行動を実行する．

2.2 ボットネットによる被害

ボットネットにより被害を受けるのは，ボットネットから攻撃を受けたノードやその管理者だけではない．多くの場合，ボットのノード管理者はノードがボットネットへ加入した事実を把握おらず，自分の管理しているノードが悪意のある活動をしていると認識していない．そのため，ボットに感染したノードの多くは，ノード管理者に認識されず，対応がなされない．ネットワーク内のボットの存在は，次に述べる理由により，ネットワーク管理者にとっても脅威となる．

機密性に対する脅威
ボットに感染した活動によって，ノード内部に含まれる個人情報の流出や，組織の機密情報の流出を引き起こす．この結果，クレジットカードの不正利用や内部情報の流出によって，金銭的あるいは情報的な不利益を引き起こす．

可用性に対する脅威
ネットワーク利用者はネットワークやコンピュータの資源が使えなくなるおそれがある．ボットの活動は，SPAM 送信によるメールサーバへの負荷集中やマルウェアの感染活動のためのトラフィック圧迫など，ネットワークが提供するサービスの可用性を脅かす．本来の目的外でネットワークの帯域が利用され，ネットワークの帯域が増加すると，ネットワークの接続性が不安定になる．ネットワーク管理者が安定したサービスを提供するには，ネットワークの可用性を確保しなければならない．

2.3 ボットの感染活動

脆弱なノードに感染する際のボットの一般的な感染活動は**攻撃**，**転送**と**接続**の 3 段階に分けられる [6].

1. 攻撃:他のノードから攻撃が発生し，ダウンローダと呼ばれる小さなソフトウェアが送り込まれる
2. 転送:インターネット上の配布元ノードからマルウェアをダウンロードする．
3. 接続:ボットネットの管理サーバに接続する．

本節ではこれらのボットの基本的な活動について述べる．

2.3.1 攻撃: マルウェアの感染

ボットネット管理者はより多くのノードを自らが管理するボットネットに参加させるために，他のノードに対してマルウェアの感染活動を試みる．マルウェアの感染は**能動感染型**と**受動感染型**の２つに分けられる．以下にそれぞれの特徴を述べる．

能動感染型

能動感染型のマルウェアの感染とは攻撃者あるいはボットが能動的に感染のための通信を開始する感染方式である．一般的に本手法は脆弱性を持つネットワークサービスが動作しているノードに対して，エクスプロイトコード (Exploit Code) と呼ばれる不正な動作を行うコードを送り込む．通信をすべて許可するネットワーク下において脆弱なサービスを動作させているノード全てが能動感染型は外部から感染を受ける対象となる．そのため，多くのノード上で利用されているサービスに脆弱性が発見された際は即座に世界中のノードがボットに感染する可能背がある (一般に Zero-day Attack と呼ばれる)．

受動感染型

受動感染型のマルウェアの感染とはノード管理者が自発的に悪意のあるプログラムを実行し，ボットに感染する感染方式である．現在，多くの受動感染型のマルウェアは，ウェブ経由あるいはメール経由で感染を行う．ウェブ経由の受動感染はウェブページにウェブブラウザの脆弱性を利用したコードが含まれており，不正な動作を行うコードを送り込む感染方式である．メール経由の受動感染はメールにマルウェアあるいはマルウェアをダウンロードするためのソフトウェアを添付し，利用者に対し自発的にソフトウェアをクリックさせる感染方式である．

また，すべての感染に用いられる攻撃は**既知の攻撃**と**未知の攻撃**に分類される．

既知の攻撃

既存の攻撃は，セキュリティベンダ (セキュリティ対策の製品開発や研究などを行っている企業) に認知されている攻撃を示す．認知された攻撃の多くは，攻撃の際に用いられるパケットの特徴，通信の特徴を攻撃のシグネチャとして定義し，このシグネチャを用いて攻撃の検知を試みる．既存の攻撃に関しては従来から多くの研究が行われており，インターネット利用者は既存の対策技術を正しく用いることで，十分な対策ができる．

未知の攻撃

未知の攻撃は，セキュリティベンダに認知されていない攻撃を示す．これらの攻撃は解析によって既知の攻撃となる．しかし，未知の攻撃の解析には早くとも数時間から一日程度かかる．また，解析結果を対策技術に適応するには攻撃のシグネチャを配布する必要がある．しかし，その配布にかかる時間の問題から，セキュリティベンダにより解析が完了しても即座に攻撃を検知することは困難である．

2.3.2 転送: ボットのダウンロード

攻撃を受けたノードはマルウェアの転送を開始する．近年の攻撃者は，攻撃が成功したノードをボットとして利用するが，その際にはマルウェアをノードに送り込む必要がある．これは，多くの攻撃が利用する攻撃の手法は文字列処理など，メモリ管理の脆弱性を利用したものであり，大きなプログラムの転送が困難だからである．しかし，攻撃が利用する脆弱性によって送信できるソフトウェアの容量には制限がある．そのため，攻撃者は**ダウンローダ**と呼ばれる，小さなソフトウェアを送り込みノードに実行させる．ダウンローダはインターネット上の他のノードからマルウェアを転送する機能のみを持ったソフトウェアであり，多くの場合マルウェアに比べてファイルサイズが小さい．

ダウンローダによってマルウェアのダウンロードが開始されるが，その手法には**プッシュ型転送**と**プル型転送**の 2 つの種類がある．この 2 つはそれぞれマルウェアの転送方法が異なる．それぞれの転送方法について図 2.2 を用いて解説する．図左のプッシュ

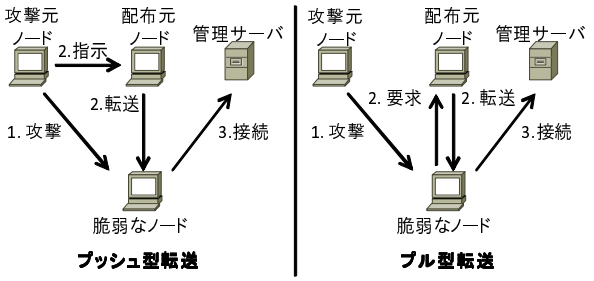

図 2.2: プッシュ型転送とプル型転送の例

型転送ではダウンローダが実行されると脆弱なノードはあらかじめプログラムされたポート番号でインターネットからの通信を受け付ける．攻撃元ノードは配布元ノードに対して脆弱なノードの IP アドレス情報とポート番号を伝える．攻撃元ノードと配布元ノードは同一ノードの場合もある．配布元ノードは受け取った情報を用いて，脆弱なノードにマルウェアをアップロードする．プッシュ型転送はインターネット上のノードから通信が開始されるため，NAPT など，外部からの通信が制限された環境においてプッシュ型の転送は動作しない．図右のプル型転送ではダウンローダが実行されると脆弱なノードはあらかじめプログラムされた IP アドレスおよびポート番号で管理サーバに接続し，マルウェアをダウンロードする．プル型転送は脆弱なノードから通信が開始されるため，外部からの通信が制限された環境においても動作する．

2.3.3 管理: 攻撃者との通信

マルウェアの転送が終了し，マルウェアに感染したノードはボットネットの管理サーバに接続する．以後は管理サーバからの命令に従い活動する．管理とはボット管理者

からの命令がボットネット上のノードに伝えられる状態を示す.

Barford らの調査 [7]，Honeynet プロジェクトの調査 [6] や筆者らの調査によると，現在のボットの管理方法は **IRC プロトコルを利用した継続的な管理**と．**HTTP プロトコルを利用した定期的な管理**に分類できる.

IRC プロトコルを利用した継続的な管理
IRC プロトコルを利用した継続的な管理をするボットネットでは，ボットに感染したノードが管理サーバに対して IRC プロトコルを用いて接続する．ボットは指定されたチャンネルあるいは PRIV メッセージ (IRC プロトコルにおいて定義されているチャンネルを介さないユーザ間のメッセージコマンド) で攻撃者から命令が発行されるのを待つ.

HTTP プロトコルを利用した定期的な管理
HTTP プロトコルを利用した定期的な管理をするボットネットでは，ボットに感染したノードが管理サーバに対して一定間隔で HTTP プロトコルを用いて接続する．ボットからの接続に対して管理サーバが命令を返答すると，ボットは応答で返された活動を実行する.

IRC を利用したボットネットの管理は HTTP を利用したボットネットの管理に比べて，攻撃者が命令を発行した際に即座に命令を実行できる点で優れる．また，IRC の性質から TCP の通信を継続するため，ノードがインターネットに接続している状態であるかの判断が容易である．HTTP を利用する管理は，命令を実行する際の即時性に欠けるが，HTTP は汎用的に利用されているプロトコルであるため，ネットワーク管理者やセキュリティベンダによる対策を難しくし，ボットの活性率を高められる.

調査では，これらの管理方法はプロトコルが利用する一般的なポート番号 (Well-Known ポート番号 [8]) が変更されている事例や，それぞれのプロトコルに独自の変更が加えられている事例が観測されている．筆者らの調査では，第 2.5.4項で述べる隠蔽化のために，これらの変更を利用している攻撃者が多い.

2.4 ボットの活動

ボットとなったノードは第 2.3.3項で述べたとおり，管理サーバに接続し，ボットネット管理者からの命令を待ち受ける．ボットネット管理者からの命令を受けた時，あるいは管理サーバに接続すると同時にボットは多様な動作を行う．動作はボットの種類やボットネット管理者の命令によって異なる．また，すべての動作が可能なマルウェアもあれば，単一の活動機能のみを実行できるマルウェアもある．本節では，ボットの活動の詳細について述べる.

2.4.1 能動的な感染活動

第2.3.1節で述べたとおり，いくつかのマルウェアでは積極的な感染活動にボットネットを拡大させるための手段として，能動感染型の攻撃を行う．そのため，ノードがボットに感染すると即座に，あるいは管理サーバに接続すると即座に同一ネットワークあるいはインターネット上のホストに対して脆弱性の走査 (脆弱性スキャン) あるいは攻撃を試みる．

2.4.2 分散サービス拒否攻撃

分散サービス拒否攻撃はDDoS攻撃とも呼ばれ，多量のインターネットノードにサービスを提供するサーバ，あるいはネットワークを提供するルータに対して一斉にサービス要求を送信する攻撃である．サービスを対象としたDDoS攻撃では，ボットに感染したノードから特定のサーバ上のWebサーバやメールサーバを対象として集中的なアクセスが発生させる．サービスを提供しているサーバでは自らの計算機資源を消費してサービスを提供する．そのため，極端に多量のサービス要求が発生すると，資源が不足してすべての要求に対応できない．攻撃者は特定の企業のWebサーバやメールサーバにDDoS攻撃を実施するあるいは実施すると脅しをかけることで，企業のイメージを低下させたり，金銭的な脅迫を仕掛けることができる．

2.4.3 迷惑メールの送信

迷惑メール (一般にSPAMメールとも称される) の送信とは単一のユーザから無差別にインターネットメールを送信する活動である．迷惑メールは**営利目的**あるいは**悪意を持つWebページへの誘導**のために送信される．本節ではこの2つについて述べる．

営利目的

営利目的にSPAMメールを送信するインターネットユーザ (SPAM送信者) は自らの利益となるようなWebページへのURLを記載したメールを送る．たとえば，自らが経営するインターネット上の電子商取引サイトのURLなど自らの収益に結びつくようなURLを記載する．

悪意を持つWebサイトへの誘導

第2.3.1節で述べたようにWebブラウザやWebコンテンツが利用するアプリケーションの脆弱性を用いた攻撃手法が存在する．そのため，ボットネット管理者の一部はこれらの攻撃コードを含んだWebページを用意する．そして，ボットネットを拡大するために無差別なユーザに対して用意したWebページのURLを含むメールを送信する．

2.4.4 他ホスト攻撃のための踏み台

ボット管理者あるいは悪意のあるインターネットユーザが他のノードにアクセスする際，あるボットを**踏み台**にする事例が存在する．踏み台となったノード上ではSOCKS[9]などのサービスがノード管理者の意図に反して動作させられる．攻撃者はそのホストを中継して他のインターネットノードに攻撃を試みる．踏み台となったノードは本来，マルウェアの被害者であるが，この場合は攻撃を中継する加害者となる．また，攻撃を受けたノードから踏み台となったノードが所属するネットワークの管理者にインシデントを通達しても，対応がなされる保証はない．さらに，踏み台となったホストから攻撃者を特定することは連携した協力を得る点から難しい．

2.4.5 昇格

ボットネットはボット管理者からの命令を実行するボットと，管理サーバや転送サーバといったいくつかの特殊なノードから構成されている．ボットネット上に参加する大部分のノードはボットである．特殊なノードからマルウェアが駆除されて機能しなくなった場合，攻撃者はボットネットの冗長性を高めるためにボットを特殊なノードに昇格させる．昇格したノードは新たなプログラムをダウンロードし，ボットネットの運用に荷担する．

2.5 ボットネット対策の困難さ

これまでに述べたとおり，ボットネットは悪意のある活動を目的としている管理されたノード群である．セキュリティベンダや多くのセキュリティ研究者は，ボットネットの対策を提案・検証している．しかし，未だインターネット上から効果的にボットの駆除や，ボットネットからの被害の低減は困難である．本研究ではボットネット対策の困難さを，**冗長性**，**分散性**，**規模性**と**隠蔽性**に分けて考える．

2.5.1 冗長性

冗長性とは，攻撃者がノードあるいはネットワークの管理者にボットネットの対策を施されてもボットネットを維持可能である様子を示す．ボットネットの維持とは，第2.1節で述べたボットネットの機能のすべてが動作している状態を維持することである．ボットネットは管理サーバがボットに命令を伝達する．そのため，管理サーバがネットワークから切断されてしまえば，ボットネットは活動を継続できない．また，転送サーバの動作が停止すれば，マルウェアの転送を阻止し，ボットネットの拡大を防げる．

しかし，近年のボットネット管理ではこれらの手法への対策が見られる．多くのマルウェアでは，ある管理サーバに対する接続が失敗した場合，異なる管理サーバに接続を試行する．主な２つの例として，あらかじめプログラムされた予備のホストに接続

する例と，管理ホストを DNS 名で指定しておき，対策をなされたと同時に DNS のエントリを変更する例が挙げられる．そのため，ある管理サーバを対策しても，ボットネットの管理が別の管理サーバに移管され，ボットネットの動作の阻止には至らない．

2.5.2 分散性

分散性とは，ボットネットを構成するボットが，複数なネットワークに分散している様子を示す．分散性の高いボットネットは，ボットが世界中に分散している．ボットは多くの異なるネットワークに所属し，各ネットワークの管理者が異なるため，個々のボットへの対策が困難となる．あるボットに対策しようとするとそれぞれのノード管理者あるいはネットワーク管理者に通知をしなければならない．多くの場合被害者が得られるボットの情報だけではノードの管理者に連絡は取れないため，ネットワークの管理者に通知されるが，対応に時間がかかる場合が多い．更に，ノード管理者あるいはネットワーク管理者が通知に対応しない場合も考えられる．

2.5.3 規模性

規模性とは，ボットネットを構成するボット数の大きさを示す．ボットネットを構成するボットの規模は数百台から数千台のものが多い [10]．過去に検出された最大規模のボットネットは 50 万以上のノードから構成されている [3]．2006 年度の調査ではインターネット全体におけるボットの感染率は 5%とも 10%以上ともいわれており [11]．平成 17 年末における日本でのインターネット利用者は 8529 万人 [12] の環境下でボットの感染率が 5%と仮定すると日本だけで約 400 万以上ものインターネット利用者がボットに感染していると推測される．

一度，マルウェアに感染したノードは，ネットワークから切り離し，アンチウィルスソフトなどによる駆除，もしくはシステムの再インストールをしなければならない．インターネット上の全てのボットからマルウェアを駆除するのはその規模から困難である．

2.5.4 隠蔽性

隠蔽性とはボットが対策を施されないように自らがマルウェアに感染していることを感染ノードのネットワーク管理者に知られないようにしている状態を示す．近年のボットはますます隠蔽性を高める傾向にあり，この様子は攻撃時と感染後の活動から見て取れる．

まず，攻撃においては受動的な攻撃の移行が主な傾向として挙げられる．従来の攻撃は Blaster[13] や Code Red[14] に代表される能動的な攻撃であり，各ネットワークの管理者は外部からの攻撃を観測して攻撃元の情報を得られた．しかし，2007 年より

Web を介した攻撃が増加し，既存の手法では攻撃元の特定が困難になってきた．また，攻撃により送り込まれるボットに高度な対検知技術が用いられている．現在，ボットの隠蔽化に用いられている主な 3 点の手法について述べる．

- 多重パッカーを利用した隠蔽化
  パッカーは実行可能なファイルを圧縮するソフトウェアである．パッカーはインターネットの帯域が大きなファイルの転送に十分な帯域を持っていない時代に実行ファイルのファイルサイズを縮小する目的で使われていた．また，自分が作成したコードに対する第三者からのソフトウェアの逆アセンブルを困難にするための実行ファイルの暗号化にも用いられていた．

  マルウェアの作成者はパッカーを既存研究の検知を回避する目的で用いる．まず，パッカーで圧縮した実行形式ファイルは，バイナリファイルの内容が元のファイルと異なるため，既知の攻撃の特定のファイル内容を用いている検知手法などを回避できる．近年では，パッカーを解読することでパッカーを用いた検知を試みる手法が存在するが，パッカーで圧縮されたファイルをさらにパッカーで圧縮する検知の回避手法が存在する．回避手法は複数回パッカーを解読すれば元のファイルを得られるが，解読する最大数をあらかじめ決めなければならない．もしも攻撃者が無限回パッカーを実行しても，さらにパッカーに復元されるソフトウェアを送り込むと，検知手法は常にパッカーの解読に資源を奪われてしまうからである．

- 管理サーバへのアクセス手法の変化
  2006 年当時，管理サーバへの接続はボットから管理サーバに接続し，特に障害などが発生しない限り接続を継続する事例がほとんどであった．しかし，近年ではボットネット管理者からボットに命令を伝達する際に，ボットが定期的にインターネット上の管理サーバに命令を取得しにいく事例が多く見受けられる．特に管理サーバへのアクセスに HTTP を利用する事例が多く，正常なトラフィックと区別が難しい．継続したトラフィックに比べて特徴が少なく，ネットワーク管理者による検知を難しくしている．

- 受動的な攻撃への遷移
  受動的な攻撃への遷移とは観測されている攻撃の手法が能動感染型の攻撃から受動感染型の攻撃に移っている事象を示す．能動感染型の攻撃では感染したノードが自ら他のインターネットノードに攻撃を試行するため，ネットワーク管理者やセキュリティベンダに対して自らの存在を露呈してしまう．そのため，対策をとられやすいが，受動感染型の攻撃ではノードからの接続を待ち受けるため，能動感染型に比べて自らの存在を露呈しにくい．

2.6 ボット対策の要求

本章ではボットネットとボットの機能について述べた後，対策の困難さについて述べた．ボットネットはボットネット管理者によって管理された世界中に分散する１つのオーバレイネットワークである．ボットネット管理者はボットに対して任意のタイミングで任意のコマンドを実行可能であり，ボットの本来の管理者や，インターネットシステムにとって大きな脅威となりうる．さらに，ボットネットには４つの点から対策が困難である．そのため，ボットネットの対策には次の２点が必要である．

ネットワークベースの検知

ノードベースの検知はボット対策を取らない意識の低い利用者に無力である．また、ボットネットベースの検知は近年、攻撃者が対策を練っている．管理できる範囲内においてネットワーク内のボット感染のインシデントを即座に知るのが、ボット対策には望ましい．

亜種への対応

隠蔽化が進み多くの亜種が発生している現在において，亜種への対策は難しい．しかし，ボットの亜種の発生はボットネット対策を考える上で最も重要な課題であり，如何に亜種を発見するかを考えなければならない．

第3章　ボットネットに対する既存の対策技術

第2章において，ボットネットは高度な技術が投入されたオーバレイネットワークであり，隠蔽性の高さや分散性から対策が困難であることについて述べた．マルウェアの感染や攻撃者の活動を検知あるいは抑制する既存の研究が存在する．本章では，既存のボットネット対策を対象と利用する情報によって4つに分類する．

3.1　サービスを対象とした対策

サービスを対象とする手法は，受動的な攻撃の感染源となるWebやメールのサービス上で何らかのボット対策を施す手法である．これらの対策はサービスの利用者やネットワークに依存せずサービスを利用する利用者全員に利益がある．サービスによる感染の予防は，Webサイトへのアクセスを解析し，ボット感染の危険が予想されるサイトへのアクセスを未然に防ぐ方法である．本手法はセキュリティに関する知識を持たないインターネット利用者に対しても有効である．

メールサービスによる感染防止対策は，メール送信サーバあるいはメール受信サーバにおいて配信するメッセージを検査するシステムである．広く使われているメールサービスにおける対策手法として，ClamAV[15] やIronPort[16] が挙げられる．前者は無償で提供されているアンチウィルスソフトウェアで，Postfix[17] やSendmail[18] といったUNIXシステム上でよく利用されているメール配送システム (Mail Transfer Agent) で容易に利用することが出来る．このソフトウェアはシグネチャベースの検知が可能である．後者は迷惑メール防止機能も有した統合型のメールサービスアプライアンスである．流行しているメールウィルスのシグネチャを高速に伝搬する機能や，メールが中継していたドメイン名やIPアドレスを独自のレピュテーション評価機構を用いて，疑わしいメールを検知する．これにより，この手法によってあるネットワークにおけるメールによるマルウェア感染の機会を低減できる．

Web検索サービスの１つであるgoogle[19] では，検索結果にボット感染の可能性を持つ可能性のページを表示させない技術を導入している．googleでは，GhostBrowser[20] と呼ばれる技術が導入されている．この技術は，利用者がWebコンテンツにアクセスした際に，動的に利用者の意図しないファイルがダウンロードされるWebコンテンツを悪意の持つWebコンテンツと定義する．googleはサービス利用者に検索結果を返す際にWebページにマルウェア感染の可能性があるページについて警告を発する．これ

により，google を利用するインターネット利用者がマルウェアに感染する機会を低減できる．

3.2 インターネットにおける対策

インターネットを対象とする手法とは，ボットネットが管理サーバに集中管理される仕組みを利用してボットネット自身の動作の抑制を試みる手法である．管理サーバを停止すれば，該当ボットネットに所属するボットの活動を停止できる．そのため，ボットの規模性と分散性に対して高い効果が期待できる．

3.2.1 DNS による検知

DNS による検知は，多くのボットが管理サーバとの接続にドメイン名を利用している特性を利用したボット検知方法である．そして，攻撃者の多くは動的な DNS のアップデートの仕組みを用いたダイナミック DNS と呼ばれるシステムを用いて，冗長化を図り，対策を困難にしている．

そこで，ボットネットの特性を利用した DNS トラフィックの監視により，ボットの検知を試みる検知手法が研究されている．大学や教育・研究機関へのネットワークプロバイダである SURFnet[21] は，DNS 通信を調査しマルウェアに感染したノードの検出を試みている [22]．この調査では，以下のような DNS の通信が観測されるノードはマルウェアに感染している傾向が強いと述べられている．

- マルウェアに利用されているドメイン名解決
- 頻繁に名前解決されるドメイン名解決
- 提供されているリゾルバ以外でのドメイン名解決
- 一般的に用いられないレコードでのドメイン名解決

この報告では，ネットワーク内部の該当ドメイン名に対するノードを摘発したり，ダイナミック DNS の組織に該当レコードの削除を要請できると述べられている．DNS を用いた検知は特定の実装に依存しない検知手法である．しかし，本手法は研究の段階であり，実用化に至っていない．

3.2.2 管理サーバの発見と監視

本手法は，管理サーバを発見し，その通信を監視することで，ボットネットを構成するボットに関する情報を収集をする．たとえば，IRC プロトコル [23] を管理に使用したボットネットにおいては，ボットの通信から管理サーバの制御に使われるチャン

ネルを特定・接続することで，同じチャンネルに接続しているノードの情報を得られる．ボットネットの管理に用いられているチャンネルに接続しているノードはボットであるため，該当ノードの管理ネットワークに通達しボットへの対応を依頼できる．

3.3 ネットワークベースの対策

ネットワークベースの対策手法は，ネットワーク内のノードのボットの活動や，ネットワーク外部からの感染攻撃を検知する手法である．

3.3.1 ミスユース型 NIDS

攻撃やスキャン活動の検知，あるいは遮断を目的とした管理ネットワークに接続されるシステムは，セキュリティデバイスと呼ばれる．NIDS(Network-based Intrusion Detection System) は，特に内部ネットワークにおけるネットワークトラフィックを監視するセキュリティデバイスである．

ミスユース型 NIDS は NIDS でも特に悪意のあるトラフィックに対して，シグネチャと呼ばれる特徴データを予め用意しておき，トラフィックがそれらに一致するかを比較するセキュリティデバイスである．シグネチャを用いるミスユース型は，ネットワーク利用者が適切にシグネチャを調整することで，正常なパケットの誤検知なしに既存の攻撃を検知できる．このため，特定のシグネチャが一致する活動に関して有効に利用できる．

ただし，シグネチャの登録されていない攻撃は検知できない．そのため，既存のシグネチャ型 IDS では未知の攻撃の検知が困難である．未知の攻撃に対して，honeycomb[24] や EarlyBird[25][26] といった研究をはじめ，シグネチャ型 IDS のためのシグネチャを動的に生成する機構の研究 [27] が活発である．これらの手法を用いて，従来のシグネチャ型 IDS では困難であった未知の攻撃検知が可能になり，ボット検知への応用にも利用できる可能性がある．しかし依然としてシグネチャを用いるという性質は変わらないため，亜種の十分な検知は難しい．

ミスユース型 NIDS の実装としては snort[28] が広く使われている．ミスユース型 NIDS で用いられているシグネチャの例を図 3.1に示す．例に示したシグネチャはボットがノードに感染した通信を検知するためのシグネチャであり，データのサイズと特定のバイナリ列が判定条件として設定されている．シグネチャを用いるミスユース型の検知手法は，シグネチャが登録されている活動に対しては有効であり，適切に調整されたシグネチャを用いることで正常なパケットの誤検知なしに既存の攻撃の検知を目的とする．このため，既知のマルウェアの感染や，定義されたシグネチャと同じ特徴をもつ活動に関しては有効に利用できる．

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
(msg: “BLEEDING-EDGE VIRUS Agobot/Phatbot Infection Successful”;
flow: established; dsize: 40;
content:”221Goodbye, have a good infection |3a 29 2e 0d 0a|”;
reference: url,www.lurhq.com/phatbot.html; classtype: trojan-activity; sid:
2000014; rev:2; )
```

図 3.1: snort 用に書かれたボット感染を検知するシグネチャの実例

3.4 ノードベースの対策

ノードにおけるボットへの対策手法では，各ノード上でソフトウェアの実行やネットワークの通信を監視し，ボットへの感染を予防あるいは検知する．ノード上における対応では，メモリやファイルシステムに対する不正なアクセス制御や，マルウェアによく含まれるシステムコールの特徴など，感染あるいは感染後の詳細なボットの特徴を用いた検知が可能である．

現在，多くのセキュリティベンダから一般的なインターネット利用者に対してセキュリティ対策を施すように喚起が促されている．ノードにおけるセキュリティ対策ソフトウェアは一般に**アンチウィルスソフト**と呼ばれる．アンチウィルスソフトの例として，symantec 社 [29] はから販売されているアンチウィルスソフト製品である Norton AntiVirus 2009[30] が挙げられる．基本的なアンチウィルスソフトの機能はシグネチャ型の検知である．ネットワークベースの対策においてもシグネチャ型の検知は用いられるが，ノードベースの対策ではネットワークを介して感染するマルウェアだけではなく，USB デバイスやフロッピーディスクを介して感染するマルウェアも検知の対象とする．もう 1 つの基本的な機能として，ファイアウォール機能が挙げられる．近年のアンチウィルスソフトで用いられているファイアウォール機能は，あらかじめ許可されたプロセスからの通信や指定されたポート番号の外部からの通信のみを許可する．それ以外の通信については全て破棄する．また，近年のアンチウィルスソフトではヒューリスティック・ビヘイビア型のマルウェア検知を実装している．ヒューリスティック・ビヘイビア型の検知機能はシステム資源に対する異常なアクセスや，本来，頻繁に発生しないディスク領域へのアクセスを利用してマルウェアらしい振る舞いを検知する．また，Norton AntiVirus 2009 においては特に AntiBot[31] と呼ばれるボットに特化した対策機能が導入されている．symantec 社によると，AntiBot はシグネチャ更新を必要としない検知機能であり，リアルタイムにボットの検出と駆除を可能とするノードにおけるヒューリスティック・ビヘイビア型の検知手法である．

近年のノードベースの対策は，これまでに述べたように多くの視点と対策でマルウェアの感染防止を試みる．しかし，ノードベースのおける検知は 2 つの点から十分なマ

ルウェア感染の防止ができない．まず１点は，攻撃コードやボットのプログラムを制作する攻撃者らは，よく知られたアンチウィルスソフトを保持している．そして，自らが作成したプログラムがアンチウィルスソフトで検知されないように工夫する．

もう１点はユーザの意識である．アンチウィルスソフトは高機能かつ多角的にノードのセキュリティを保護するが，アンチウィルスソフトの動作をノードの管理者に停止されると，一切の保護機能が無効となる．新しいソフトウェアの導入時などにおいて，導入するソフトウェアが正常に動作しない場合にはアンチウィルスソフトを停止させるように推奨する事例もある．

また，多くの場合，音声チャットシステムや画像配信システムを利用するためには，ファイアウォール機能を適切に設定する必要があるが，この設定には専門的な知識が求められるため，システムの利用のためにアンチウィルスソフトを無効化する事例もある．また，アンチウィルスソフトの無効化まで至らずとも，アンチウィルスソフトの不正動作を検知し，ユーザにその対応を求める実装が多く用いられている．しかし，先に述べたように多くの一般利用者は専門的な知識を要していないため，特に意識をせずに通信を許可してしまうことが考えられる．

前者の問題については解決が難しいとされているが対策が研究されている．だが，後者の問題については利用者のリテラシに関わる問題である．このため，ノードベースの検知によりボットの感染を低減させることは難しい．

第4章　通信データの収集

筆者らが運用しているボットに感染したノードが通信するトラフィックを調査するためのボット通信の収集システムについて述べる．

4.1　ボット通信収集に求められる課題

現在，マルウェア対策手法の研究や実現のために能動的に感染するマルウェア検体やWeb経由での感染のマルウェア検体を収集するための取り組みが行われている．より多くのボット検体とその活動通信を得るためには既存の機構に加えて２つの機能が必要となる．

4.1.1　自動的な収集機能

2008年10月度のサイバークリーンセンターの調査 [32] によると，既存の手法では検知できないマルウェアの亜種は月に千件以上発生しているとされている．第1章で述べたようにボットネットを用いた経済モデルが確立されており，今後もマルウェアの亜種が発生する件数は現在と同様あるいは増加すると考えられる．また，松木の調査 [33] で述べられているように，自らのプログラムを書き換えながら感染するポリモーフィック機能型のボットが発生している．このような状況から，通信データの収集対象となるマルウェアは多種多様に発生すると予想される．

さらに，多量のボットを実行するには多くの人件費がかかる．そのため，能動的な攻撃を試みるボットに関しては収集したボットを随時実行し，受動的な攻撃を試みるボットに関しては疑わしいコンテンツを随時取得できる必要がある．

4.1.2　外部への通信の制限

ボットに感染したノードが通信する内容を取得するためには実機あるいは実機に準じた環境下でボットを活動させる必要がある．ボットを活動させることで，管理サーバとの通信や，活動に関する情報など，ボットの活動を知る上で重要な情報を得られる．だが，第2.4節で述べたとおり，活動を開始したボットは外部に対する感染活動や他のインターネット利用者に対する迷惑メールの配送といった悪意を持つ活動を試行

し，攻撃者の活動を幇助してしまう．そのため，収集環境から外部に対する悪意のある通信を抑制は抑制し，それ以外の通信は許可する必要がある．

4.2 ハニーポット: 能動感染型ボットの取得

マルウェアの採取にはNepenthes[34]を用いる．Nepenthesは主にMicrosoft Windowsに含まれている脆弱性をエミュレートし，攻撃と攻撃元の情報を記録するハニーポットである．Nepenthesは脆弱性のエミュレート機能だけではなく，脆弱性を利用した結果結果ダウンロードされるマルウェアを取得する機能も有する．しかし，Nepenthesの出力するログファイルには研究に際して不十分な項目がある．本研究では，元のNepenthesに不足しているログの出力機能を付録 A の通り追加した．

Nepenthesでエミュレートできるのは，ダウンローダの実行と，その結果転送されるマルウェアを保存する段階までである．転送されたソフトウェアの実行とその結果生じる通信については別の環境で観測する必要がある．本研究では，自動化のために，x86仮想マシンであるVirtualBox 2.1[35]を利用した．実行プラットフォームとして，VirtualBox上に４つの仮想マシンを作成し，それぞれに対してMicrosoft Windows XP SP2を導入した．各仮想マシンはNepenthesで取得されたマルウェアを順次実行し，一定時間が経過した後に再起動する．マルウェアを実行後に再起動するのは，マルウェアの活動を発見するためである．

4.3 ボットクローラ: 受動感染型ボットの取得

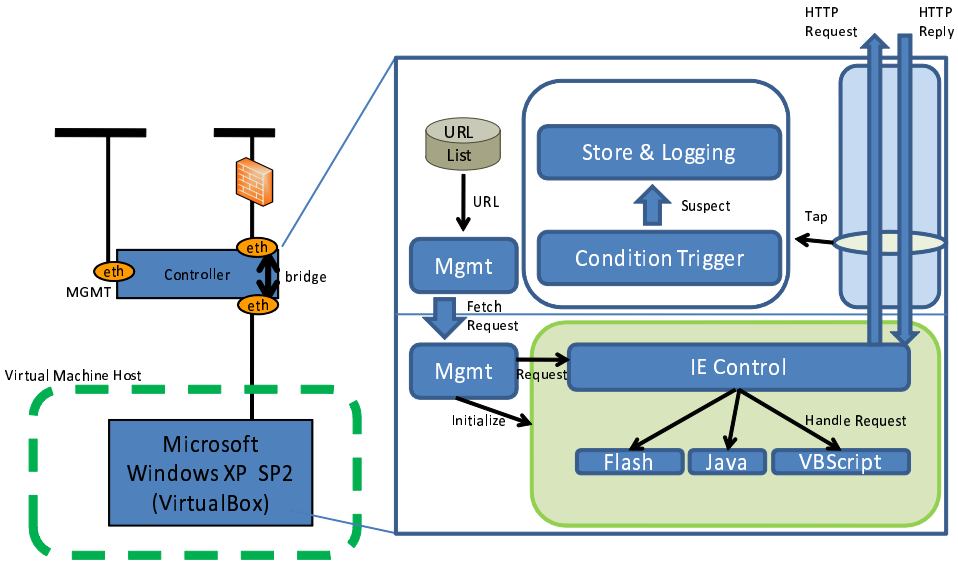

図 4.1: 受動感染型ボットの収集機構

図 4.1 に受動感染型ボットの収集機構の概要を示す.

Web 感染型マルウェアを取得するにあたり最も配慮すべき点は，アクセスするとマルウェアに感染する Web ページの URL である．攻撃者はサイトを運用する Web サーバ管理者による対策や，セキュリティベンダによる Web レピュテーションなどの対策に対して対策するため，常に新しいサイトを構築している．攻撃者は Web サーバのクラック，あるいはボットを Web サーバとして設定し，DDNS と連携することで常に新しい URL で運用されるサイトを構築する．第 2.4.3項で述べたように， 攻撃者は自らの作成した URL にユーザを誘導するため，無差別に用意したサイトの URL を含む迷惑メールを送信する．本研究では迷惑メールに含まれる URL として，2008 年 4 月から 2008 年 8 月に筆者のメールアドレスに到達した URL をリストとして用いた.

脆弱性を持ったコンテンツへのアクセスには第 4.2節で述べた能動感染型マルウェアの実行環境と同様に，VirtualBox 上と Microsoft Windows XP SP2 の環境を利用した．また，Web コンテンツへのアクセス後，20 分経過後再起動し，その後 20 分経過してから処理を終了する動作も同様である.

4.4 フィルタ: 外部に対する悪意のある通信の遮断

第 4.1.2節で述べたように，自動実行環境下で活動するボットの悪意ある通信は遮断されるべきである．本研究では次の 3 つについて通信の遮断が必要と考える.

- 感染活動
  感染活動は自動実行環境の中で最も留意しなければならない活動である．外部へのマルウェアの感染活動を許可するとマルウェアの拡大を助長する結果となり自動実行環境が加害者となる．現在，我々が把握している能動的な活動を試行するマルウェアは Microsoft Windows のよく知られた脆弱性を利用する．外部へのこれらの通信を禁止する必要がある.

- 迷惑メールの送信
  近年のマルウェアの中には感染と同時に外部の SMTP サーバにアクセスし，迷惑メールの送信を試みる事例が見受けられる．迷惑メールの一部はメールを介した感染活動である可能性がある．また，迷惑メールが感染活動を目的としない場合においても，他のインターネット利用者に対して迷惑行為を働くため，外部へのメールの送信は禁止する必要がある.

- DDoS 攻撃
  マルウェアは感染に成功すると自動的に，あるいは攻撃者からの命令を受けて外部への DDoS 攻撃を試行する事例が多く見受けられる．DDoS 攻撃は対象のネットワーク管理者によって容易に検知される活動であるため DDoS 攻撃を十するボットは減少傾向にあるが，たびたび見受けられる活動である．DDoS 攻撃も他

のネットワーク利用者に対して大きな迷惑行為となる活動であるため，これらの通信についても制限をする必要がある．

感染活動と DDoS 攻撃については内部から外部に対する TCP ポート番号 137,138,139,445,1433,1434 への通信を禁止する．また，迷惑メールの送信についてはメール送信に用いられる SMTP の TCP ポート番号 25 の外部への通信を禁止する．

第5章　通信のモデル化

本章ではボットに感染したノードにおける通信の類似性について述べる．また，効果的なボット検知のためには通信の類似性を利用した通信のモデル化が有用であることを示す．

5.1　亜種と通信の類似性

マルウェアの隠蔽化技術は近年になって急速に成長している．インターネットが広く普及していない時代に流行したコンピュータウィルス (Cascade や Cookie Monster) やインターネット普及初期に流行したワーム (Happy99 や Blaster) といったマルウェアに対しては，ミスユース型検知手法が有効であった．これらのマルウェアは毎回同じ通信内容で他のノードに感染を試行し，自己複製を繰り返していたため，特定の情報を用いたミスユース型検知手法が効果的であった．しかし，近年では攻撃時の通信を難読化して感染するマルウェアや，ダウンローダで送り込むプログラムのバイナリ列を毎回変化させるマルウェアが出現している．この様子を図 5.1 に示す．図中では，３台のノードが存在し，$Host1$ が $MalwareA$ に感染している状態である．$Host1$ が２台のノード $Host2$ と $Host3$ に感染を試みる際に用いる通信は異なる．図中ではこの例を $Host1$ から $Host2$ に対して $CodeA$ の通信内容で感染を試みるが，他のノード $Host3$ に対しては $CodeA'$ の通信内容で感染を試みている．また，その攻撃が成功した際に $Host2$ と $Host3$ に送信されるマルウェアは，$MalwareA'$ と $MalwareA''$ であり，元の $MalwareA$ とは異なる内容を持つプログラムである．具体的には第 2.5.4節で述べたようにパッカーの回数を変更して，異なるバイナリ列とする手法が挙げられる．

第 3章で述べたように，ミスユース型検知手法に代表されるボットの**特定の情報**を用いた検知は困難である．現在，この問題を解決する手法として水谷らの研究 [36] や BotHunter[37] では，ボットの通信類似性に着目した検知手法を提案している．これらの研究においては，マルウェアの動作によって発生する通信を特にボットの動作と関連づけて未知のボット亜種を検知する．

実際のボット感染の通信を例に挙げて，マルウェアにおけるボット通信の類似性について述べる．2008 年 11 月に筆者らが運用するハニーポットにおいて，収集された２種類の検体を実行した際に観測された通信について解析した．２つの検体をボット A とボット B とし，それぞれの検体における通信の内容を表 5.1 と表 5.2 にそれぞれ示す．各行は通信に含まれるフローを示す．

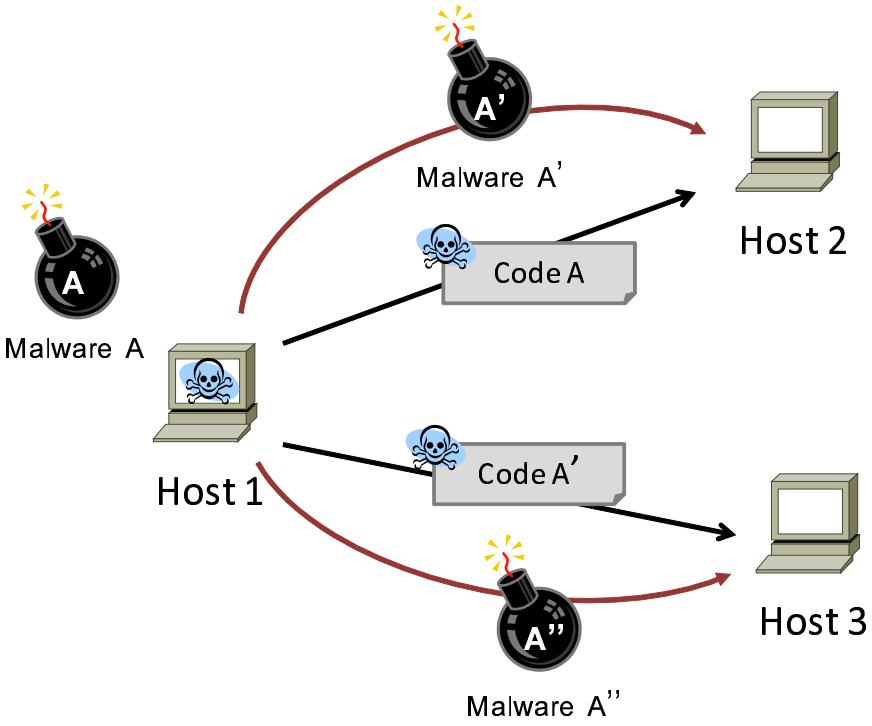

図 5.1: 攻撃パターンを変化させて感染する自己変形型マルウェアの例

表 5.1: 能動感染型ボット A における通信

番号	タイムスタンプ	通信先	通信方向	プロトコル	SrcPort	DstPort
$a1$	0.000000	218.x.x.x	*incoming*	*TCP*	2035	135
$a2$	1.095493	218.x.x.x	*outgoing*	*TFTP*	–	69
$a3$	51.532777	192.168.0.254	*outgoing*	*DNS*	–	53
$a4$	51.572196	72.x.x.x	*outgoing*	*TCP*	1033	6667

番号は解説に利用するために筆者がつけた通信を識別するための名称である．

タイムスタンプはそれぞれのボットの通信が最初に観測されてから，その通信が開始されるまでの経過時間を秒で示す．

通信先はノードに対して通信先となるノードの IP アドレスである．本論文中ではノードの特定を防ぐため，通信先がグローバル IP アドレスである場合は第 2 オクテット以降を x.x.x と伏せて表記した．

通信方向は通信が発生したノードの方向を示す．*outgoing* は，ネットワーク内ノードから発生した通信を示し，*incoming* は，インターネット上のノードから発生した通信を示す．

プロトコルは通信のプロトコルを *tcpdump*[38] で得られたプロトコル名を示す．*tcpdump*

表 5.2: 能動感染型ボット B における通信

番号	タイムスタンプ	通信先	通信方向	プロトコル	SrcPort	DstPort
$b1$	0.000000	$59.x.x.x$	$incoming$	TCP	1209	445
$b2$	0.885616	$59.x.x.x$	$outgoing$	TCP	1035	15228
$b3$	242.387435	192.168.0.254	$outgoing$	DNS	–	53
$b4$	242.425992	$24.x.x.x$	$outgoing$	TCP	1039	3000

ではポート番号から通信の内容が特定出来る場合は，アプリケーションプロトコルの名称が得られる．ポート番号からプロトコルが特定できない場合は，それまでに確実に特定できる通信のプロトコル名が得られる．

SrcPort と **DstPort** はそれぞれ通信が開始されたポート番号と通信先のポート番号を示す．

ボット A とボット B によって発生したトラフィックは一見すると DNS のパケットを除き，全く異なった通信に見える．しかし，これらのトラフィックをボットの活動と照らし合わせると，通信が発生した原因について類似性が見られる．まず，$a1$ は RPC の脆弱性を利用した攻撃であり，$b1$ は $LSASS$ の脆弱性を利用した攻撃である．それぞれ利用している脆弱性は異なるが，ノードに対して不正なコードの実行を試みる動作は共通している．次に，$a2$ は $TFTP$ を用いたマルウェアの転送であり，$b2$ はポート番号を変更した FTP プロトコルを利用したマルウェアの転送であった．転送に利用しているプロトコルは異なるが，共にマルウェアを転送するための通信である．$a3$ と $b3$ では DNS によって異なるノード名から IP アドレスを解決している．$a4$ は IRC の $Well-Known$ ポートを利用した管理接続であり，$b4$ もポート番号が変更されているが通信を解析すると IRC プロトコルを利用した通信であった．このようにボット A とボット B のトラフィックはそれぞれ，攻撃 $\Rightarrow$ 転送 $\Rightarrow$ 管理というボットネットの動作に従って発生している．

このような共通性はマルウェアの開発過程によるものであると考えられる．マルウェアの亜種は agobot や SDbot など 2000 年前半に開発されたソフトウェアに改良を加えて開発されてきたといわれている [7]．動作の目的に共通性が見られたとしても具体的な動作が異れば，第 3章で示したように既存の対策では有効性が低い．固有のシグネチャを利用するミスユース型 NIDS やアンチウィルスソフトは，マルウェア本体や通信プロトコル，通信に利用するデータにわずかな変更があるだけで検知ができなくなってしまう．そのため，マルウェア開発者は頻繁に亜種を作成することで，既存の対策を回避している．検知回避のために亜種を作成する場合は，大幅に機能を変更する必要はない．従って，あらかじめオブジェクト化された機能を組み合わせたり，部分的にデータを変更することで亜種を大量に生成していると考えられる．これらの状況から大部分のボットの攻撃手段や活動内容は類似していると仮説が立てられる．

5.2 通信のモデル化を利用した検知

本研究はマルウェアの活動における類似性に着目し，マルウェアの活動によって発生する通信をモデル化することによって，マルウェア感染の汎用的な検知を目指す．通信のモデル化とは，マルウェア特有の通信を抽象化し、それらの状態遷移を定義することである．第 5.1節で示したようにマルウェアには通信方法や通信内容は異なるが，共通した目的の動作がある．これを活動の目的毎に分類した集合を P_n とする．またマルウェアは連続して異なる通信を実行するため，マルウェアに感染したノードの通信は状態遷移として捉えることができる．よって，共通した目的の動作の集合 P_n の遷移によってモデル M_x を表すことができる．第 5.1節で示した 2 つのマルウェアの通信を利用して具体的なモデルの例を示す．

ボット A の例

ボット A が RPC の脆弱性を利用した攻撃の後，$TFTP$ でマルウェアをダウンロードし，TCP ポート番号 6667 番を用いた IRC により攻撃者と通信した．

ボット B の例

ボット B は，攻撃に $LSASS$ の脆弱性を利用した攻撃の後，通常使われるポートとは異なる FTP を用いてマルウェアをダウンロードし，TCP ポート番号を 3000 番に変更した IRC 通信を行っていた．

いずれのボットもボットネットの動作に伴う通信を発生させている．これらの動作は 1) 攻撃，2) 転送，3) 管理サーバへの接続という目的として表すことができる．それぞれを P_1，P_2，P_3 とすると，表 5.1，表 5.2の通信はともに $M_a = P_1 \rightarrow P_2 \rightarrow P_3$ というモデルで表せる．

P_n で表されるボットに関連した動作 (S_n) は，複数種類の通信の特徴を含む．例えば P_3 とした管理サーバへの接続では特定の IRC コマンドを含む通信だけではなく，長期の TCP セッションである点や ASCII 文字のみで構成されている TCP セッションも管理サーバとの通信を発見するための手がかりとなる．本手法ではボットの通信を発見する場合，特定の文字列や特定の通信方法だけではなく，多様な要素を判断条件として利用する．マルウェア開発のオブジェクト化が進むことで，既知のマルウェアの亜種に感染したボットの通信は類似した通信内容の組み合わせで表現できると考えられる．そのため、現在のボット通信を適切にモデル化することで、将来発生しうる未知のマルウェアの亜種の通信を検知できると期待される．

5.3 マルウェアの通信モデル化の例

実際のモデルの例を図 5.2に示す．マルウェアの特徴的な共通性を用いると，通信の暗号化や，一部のバイナリ列を書き換えるといった攻撃者による活動の隠蔽化に対して有効な検知ができる．図では能動感染型攻撃 (P_1)，受動感染型攻撃 (P_1')，転送 (P_2)，管理ネットワークへの接続 (P_3) がそれぞれ示されており，ここから $M_a = P_1 \rightarrow P_2 \rightarrow P_3$ と $M_b = P_1' \rightarrow P_2 \rightarrow P_3$ の 2 種類のモデルを示している．M_a は能動感染型の攻撃に

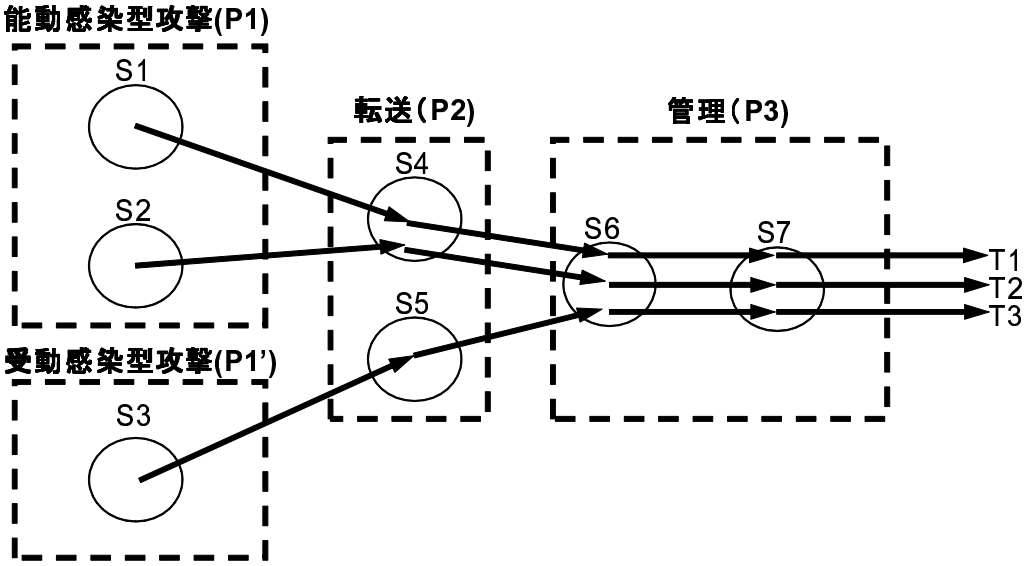

図 5.2: モデルの例

よって感染したボット，M_b は受動感染型の攻撃によって感染したボットの動作モデルである．P_n における各動作は S_n として表されている．例えば S_1 は LSASS の脆弱性を利用した攻撃，S_2 は DCE/RPC の脆弱性を利用した攻撃というように具体的な通信内容は異なるが，能動感染型の攻撃として P_1 に属している．T_n は各ボットの通信を表している．T_1 は $S_1 \rightarrow S_4 \rightarrow S_6 \rightarrow S_7$ という通信を発生させており，これは M_a の通信モデルに該当する．T_2 の能動感染型の攻撃は S_2 であり T_1 とは異なる通信をしているが，これも M_a の通信モデルに該当している．このように，個別の通信の詳細が異なったとしても，モデルにそった通信をしていればボットの活動と認識することができる．一方，T_3 は受動感染型の攻撃によってマルウェアに感染しているが，T_1,T_2 とは異なる M_b のモデルに該当している．

本手法を実現するにあたり，ボットの特徴となる通信 S_n のデータ数が重要な要素となる．一般的な通信とは異なるボットの特徴を抽出するために，本研究では第 4章で示したボット通信データの収集機構を利用している．従来の手法では検知対象となるマルウェアの種類毎にシグネチャを作成する必要があったため，全てのマルウェアの情報を把握する必要があった．しかし，本手法では部分的なマルウェアの情報だけであっても，各動作の特徴を抽出しモデルとしてまとめている．これによって，組合せによって作成されたマルウェアや部分的に動作を変更したマルウェアであれば，シグネチャをもたないマルウェアの亜種でも検知できる可能性が高いと言える．

第6章　評価

本章では通信データの収集によって作成したマルウェアの通信モデルを用いて，効果的にボットの検知ができることを示す．

6.1　評価環境

評価対象とする通信情報は第4章で述べた環境により収集する．表6.1に今回利用した能動感染型マルウェアの収集環境について示す．まず，今回ハニーポットを設置した

表 6.1: 評価に利用した能動感染型マルウェアの収集環境

ネットワークの種別	学術ネットワーク
ネットワークの規模	ClassC ネットワーク 12 個分相当
収集期間	2008/10/30 から 2008/12/29
モデル化に利用した収集期間	2008/10/30 から 2008/11/20

ネットワークは学術ネットワークに割り当てられているアドレス空間を用いた．また，このネットワークは ClassC ネットワークを 12 個分のアドレスを持つ．今回の評価で用いたハニーポットから得られた能動感染型マルウェアの情報は，2008 年 10 月 30 日から 2008 年 12 月 29 日に得られた通信情報である．モデル化にあたっては，期間中の 2008 年 11 月 20 日までの情報を利用した．評価にあたり，2008 年 10 月 30 日から 2008 年 11 月 20 日までの通信データをモデル化に用いたこの通信情報を Th とする．Th には期間中に検知された 80 件の異なるバイナリ列をもつマルウェアの通信情報が含まれる．また，2008 年 11 月 21 日から 2008 年 12 月 29 日までの通信データの通信情報を Tu とする．Tu は Th と同様の環境で収集されたマルウェアの通信情報であるが，モデル化に用いた通信の後に収集された通信情報であるため，未知のマルウェアに関する通信情報を含むことが考えられる．Tu には期間中に検知された 42 件の何らかの通信が発生したマルウェアの通信情報が含まれる．

また，ウェブクローラを用いた受動感染型マルウェアの通信情報については 2008 年 8 月に筆者のメールボックスにスパムとして判定されたメールから有効な Web の URL を利用した．各 URL に対して 2008 年 8 月にアクセスし，ウェブクローラから得られ

た情報からマルウェアの動作が確認できた通信情報を利用した．これらの通信情報Twには期間中に実行した 17 件の Web サイトとの通信が含まれる．

6.2 評価対象

図 6.1 に今回の評価で用いたモデルを遷移図の形で示す．

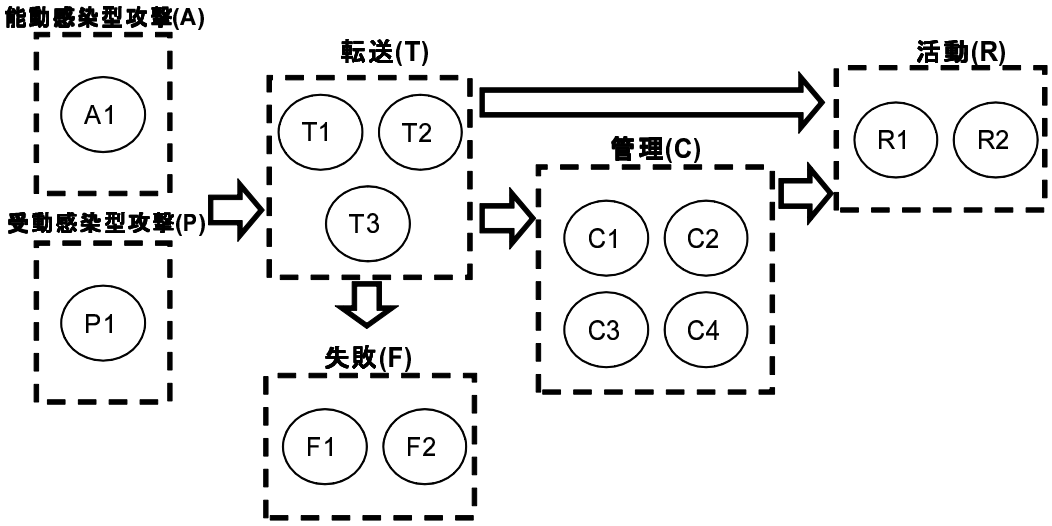

図 6.1: 評価で用いたモデル

6.2.1 モデルの解説

図 6.1 に示した図においてそれぞれのモデルが示す状態の遷移と意味を表 6.2 に示す．本モデルにおいて，能動感染型攻撃，受動感染型攻撃，転送，管理，活動，通信の失敗についての段階をそれぞれA, P, T, C, R, Fで示した．今回のモデルでは各段階の状態は 1 つから 4 つで表される．

6.2.2 モデルに含まれる状態の解説

表 6.2 に各段階における状態の遷移条件を示す．

攻撃の通信の特徴は能動感染型 (Active-Attack) の感染活動と受動感染型 (Passive-Attack) の感染活動のそれぞれを$A1$と$P1$で示す．$A1$は能動感染型の Windows のよく知られた脆弱性を利用した攻撃を示す．今回の評価において本状態は期間中に観測されたすべての能動感染型マルウェアの動作を示していることを確認した．$P1$は受動感染型の通信を示す．

表 6.2: 評価モデルの解説

モデル	段階の遷移	解説
$M1$	$A \rightarrow T \rightarrow C \rightarrow R$	能動的な感染により管理サーバとの接続と活動を行った通信モデル
$M2$	$A \rightarrow T \rightarrow C$	能動的な感染により管理サーバと接続を行ったが活動が観測されなかった通信モデル
$M3$	$A \rightarrow T \rightarrow R$	能動的な感染にマルウェアを転送した後，管理サーバとの接続が観測されず活動した通信モデル
$M4$	$A \rightarrow T \rightarrow F$	能動的な感染にマルウェアを転送したが管理サーバとの接続に失敗した通信モデル
$M5$	$P \rightarrow T \rightarrow C \rightarrow R$	受動的な感染により管理サーバとの接続と活動を行った通信モデル
$M6$	$P \rightarrow T \rightarrow C$	受動的な感染により管理サーバと接続を行ったが活動が観測されなかった通信モデル
$M7$	$P \rightarrow T$	受動的的な感染にマルウェアを転送したが管理サーバとの通信が発生しなかった通信モデル

転送 (Transfer) の通信の特徴は $T1$ と $T2$ で示される．$T1$ は，転送サーバから実行形式のファイルが転送される通信を示す．これには，Windows で実行可能なプログラム冒頭に含まれる特徴あるバイナリ列を用いる．$T2$ は，転送サーバからファイルの転送が発生している通信を示す．2006 年の筆者の調査 [39] では転送サーバからマルウェアを転送する際には FTP[40] や TFTP[41] が多く用いられていると述べられているため，この特徴を利用する．$T3$ は，転送サーバから HTTP を用いて gzip で圧縮された形式のファイルが転送される通信を示す．

管理 (Communication) の通信の特徴は $C1$ から $C4$ で示される．$C1$ は，古典的な IRC プロトコルあるいは IRC プロトコルに基づいた通信方法を用いる管理セッションの特徴を示す．$C2$ は，ネットワーク内部のボットから，管理サーバへ定期的に命令を取得する管理セッションの特徴を示す．$C3$ は，管理サーバとの通信にアスキー文字列を用いる管理セッションの特徴を示す．2005 年の Honeynet Project の報告 [6] によれば，ボットが管理サーバと通信する管理プロトコルとして，IRC プロトコルあるいは IRC プロトコルを元とするプロトコルが主であった．これらの通信ではいずれもアスキー文字の含有率が高い．$C4$ は，継続的な管理サーバとの通信を示す．

転送や管理を試行したボットがそれらの活動に失敗 (Fail) した通信は $F1$ と $F2$ で示される．$F1$ は，管理サーバのネットワーク管理者などによって対策されたボットの管理活動を示す．$F2$ は，ボットがノード管理者やネットワーク管理者によって対策をされるなど，接続性が失われた管理サーバへの接続試行を示す．

ボットとなったノードが命令に従って行動を実行 (Run) した通信は $R1$ と $R2$ で示さ

表 6.3: 評価で用いた状態の解説

状態	遷移条件	解説
$A1$	TCP ポート番号 137,138,139,445,1433 と 1434 の外部からの通信	よく知られた攻撃ポートに対してのアクセス
$P1$	TCP ポート番号 80 の内部からの通信	内部ノードからの Web アクセス
$T1$	exe ファイルの転送	マルウェア本体の転送通信
$T2$	FTP あるいは TFTP の通信	既存のプロトコルを用いたマルウェア本体の転送通信
$T3$	gzip 符号化された HTTP 通信	gzip で圧縮されたマルウェア本体の転送通信
$C1$	TCP ポート番号 6667 の内部からの通信	古典的な IRC ベースの管理通信
$C2$	単一のノードに対して同一ポートの通信を 3 分以内に繰り返し行っている	管理サーバへの定期的なアクセス
$C3$	データグラムに含まれる ASCII 文字の割合が 90%以上である	ASCII 文字列を利用した管理通信
$C4$	1 台のノードに対して 300 秒以上継続している通信	常時管理サーバからの命令を受け付ける管理通信
$F1$	30 秒以内に 5 回以上の DNS 要求を失敗している	管理サーバと接続しようとして失敗したボットの活動
$F2$	1 台の同じノードに対して 1 分以内に 10 件以上の SYN 要求を送信	管理サーバに対して接続を試行したが成功しなかったボットの活動
$R1$	10 秒以内に異なるノードに対して SMTP 接続を試行した	スパムメールの配信
$R2$	1 秒間に異なるノードに対して 10 件以上の SYN 要求を送信	他のノードに対するスキャンや攻撃活動の通信

れる．$R1$ は，ネットワーク内部のボットからインターネット上に迷惑メール配送の活動セッション，あるいは SMTP を用いて管理サーバに接続し命令を取得する管理セッ

ションの特徴を示す．本モデルの作成に用いたトラフィックデータには，外部への通信制限により SMTP の通信を許可していない．そのため，SMTP 通信を開始しようという通信は取得できるものの，該当セッション内に含まれる情報は取得できないため，具体的な活動内容の特定はできなかった．$R2$ は，外部に対してスキャンや攻撃活動を行う活動を示す．

6.3 評価結果

80 件の能動感染型マルウェアの通信情報を含むトラフィック Th，42 件の能動感染型マルウェアの通信情報を含むトラフィック Tu と 17 件の受動感染型マルウェアの通信情報を含むトラフィック Tw が，表 6.2 で示したそれぞれのモデルに対してどれだけ一致したかについて表 6.4 に示す．また，それぞれのトラフィックが，表 6.3 で示したそれぞれの状態に対してどれだけ一致したかについて表 6.5 に示す．各状態についてそれぞれの項目は “一致したトラフィックの件数/トラフィックの総件数” である．また，表 6.5 の括弧内の数値は，総件数に対する一致したトラフィックの百分率 (小数点第 3 位を四捨五入) を示した．また，各モデルは，$n < m$ である M_n と M_m に一致した場合，M_n として数え，M_m としては数えない．

表 6.4: 各モデルに一致したイベント数

状態名	Th	Tu	Tw
M1($A \to T \to C \to R$)	28	13	-
M2($A \to T \to C$)	40	1	-
M3($A \to T \to F$)	12	11	-
Ma($A \to T \to R$)	0	15	-
M4($P \to T \to C \to R$)	-	-	4
M5($P \to T \to C$)	-	-	12
M6($P \to T$)	-	-	1
該当なし	0	2	0
合計	80	42	17

6.3.1 各モデルに一致したイベント数

Th について着目すると，全ての通信において，マルウェアの転送に成功している．マルウェアの転送後，12 件については，管理サーバへの接続に失敗した．残る 68 件は管理サーバへの疎通に成功し，そのうち 28 件はボットとしての活動を試行した．全体の半数に当たる 40 件については管理サーバに接続したが活動の通信がみられなかった．

表 6.5: 各状態に一致したイベント数

状態名	Th	Tu	Tw
A1	80/80(100.00%)	42/42(100.00%)	0/17(0.00%)
P1	0/80(0.00%)	0/42(0.00%)	17/17(100.00%)
T1	80/80(100.00%)	42/42(100.00%)	8/17(47.06%)
T2	13/80(16.25%)	2/42(4.76%)	0/17(0.00%)
T3	0/0(0.00%)	0/42(0.00%)	9/17(52.94%)
C1	2/80(2.50%)	3/42(7.14%)	0/17(0.00%)
C2	12/80(15.00%)	2/42(4.76%)	13/17(76.47%)
C3	45/80(56.25%)	19/42(45.24%)	14/17(82.35%)
C4	37/80(46.25%)	21/42(50.00%)	1/17(5.88%)
F1	10/80(12.50%)	10/42(23.81%)	0/17(0.00%)
F2	2/80(2.50%)	2/42(4.76%)	0/17(0.00%)
R1	19/80(23.75%)	3/42(7.14%)	3/17(17.65%)
R2	11/80(13.75%)	35/42(83.33%)	1/17(5.88%)

Tu について着目すると，42 件の通信情報のうち 40 件が定義したモデルに当てはまった．マルウェアの転送後，11 件については，管理サーバへの接続に失敗した．残りの 14 件は管理サーバへの疎通に成功し，そのうち 13 件はボットとしての活動を試行し，他の 1 件については管理サーバに接続したが活動の通信がみられなかった．また，転送に成功した後，管理サーバとの接続を試行する通信が検知されず，外部への攻撃を試行するモデルが 15 件存在した．

Tw について着目すると，全ての通信において，マルウェアの転送に成功している．マルウェアの転送後 4 件のマルウェアは，管理サーバとの通信に成功した後に，迷惑メールの送信あるいはインターネット上のノードに対して DDoS 攻撃を試行した．転送に成功したうち別の 12 件は管理サーバとの通信が観測されたが，特に活動を行わなかった．また，残る 1 件について，実行形式ファイルの転送は確認できたが，管理サーバへの接続の試行や，マルウェアらしいの試行は発見されなかった．

Th と Tw のうちでモデルに該当しない通信はなかった．これにより，全通信はいずれかのモデルを満たした．Tu に含まれる 42 件の通信情報のうち 2 件が定義したモデルに当てはまらなかった．この 2 件については，第 6.4節で詳しく述べる．

6.3.2 各状態に一致したイベント数

攻撃の状態に着目すると，$A1$ と $P1$ はそれぞれの感染手法に拠る状態と定義したため，それぞれの通信は各感染手法の状態を満たした．

転送の状態に着目すると，Th においては暗号化や圧縮された転送が存在しなかったため，すべての通信が $T1$ を満たした．また，一部の検体では FTP や TFTP と特定出来る通信 $T2$ が観測された．Tu については Th と同様に暗号化や圧縮された転送が存在しなかったため，すべての通信が $T1$ を満たし，一部の通信が $T2$ として観測された．Tw においては約半数の通信がマルウェアを gzip 圧縮された形式で転送したため，通信の内容による実行形式の検知 $T1$ を満たしていない．残りの転送は gzip を用いた $T3$ の条件を満たしており，$T2$ の通信は確認できなかった．Tw のうち 1 件は JavaScript により難読化された HTTP 通信 $P1$ を満たし，実行形式のファイル転送が観測されたが，その後，管理と思われる通信は一切行われなかった．

管理の状態に着目すると，Th において一般的な IRC 通信の TCP ポート番号 6667 による通信 $C1$ は 2 件検出され，約半数の通信において $C3$ と $C4$ を満たした．また 12 件の $C2$ が観測された．Tu はおおよそ Th と同様の傾向が見られた．Tw においては多くの通信が $C2$ と $C3$ を満たした．継続的な通信である $C4$ は 1 件の通信が満たした．

活動について着目すると，全てのトラフィックにおいて $R1$ と $R2$ を満たす通信がいくつか観測された．Tu において約 8 割の大量の SYN 送信を伴う通信を行っている．

活動に失敗した通信に着目すると，Th については 10 件の DNS 要求の失敗 $F1$ が観測された．この 10 件のうち 9 件は同一のドメイン名に対する名前解決要求であった．また，$F2$ については Th で 2 件観測された．Tu はおおよそ Th と同様の傾向が見られ，Tw では活動に失敗した通信は発見されなかった．

6.4 既存研究との比較

本研究では，モデルの作成に利用した後にハニーポットで得られた情報 Tu を用いて，提案手法が未知のマルウェアの検知に有効であるかを比較する．比較対象として，第 3.4節で述べた symantec 社の Norton AntiVirus2009(NAV) を用いた．NAV は現在最も広く一般的なインターネット利用者に利用されているアンチウィルスソフトである．また，NAV はボットに特化したアンチボットの機能を持つ．アンチボットの機能は，シグネチャの頻繁な更新を必要とせずに，ボットの感染や動作を検知できる．本論文における提案手法はネットワークベースの検知手法であるのに対し，NAV はノードベースの検知である．NAV はネットワークの通信情報に加えて，システム資源の監視など多くの情報を得られる．広く用いられているアンチウィルスソフトである点とボットに特化した検知手法である点から，NAV と同等あるいはそれ以上の検知を実現できれば，未知のマルウェアに対して十分な検知が可能であるといえる．

NAV により，マルウェアが検知できるかの評価は NAV をインストールしたマルウェアの実行環境を構築し実現した．この環境は，マルウェアの実行環境で利用した仮想マシンを元に，NAV をインストールした．それぞれのマルウェアについて，この仮想マシン上で実行を試行し，NAV によってマルウェア感染の検知に成功したというログが出力された場合，NAV で検知が成功したと判断する．

この評価結果を表 6.6に示す．全体の 42 件のうち約 7 割にあたる 31 件が両方の手法

表 6.6: 未知のマルウェアを含む通信情報 (Tu) に対する検知比較結果

	本手法で検知成功	本手法で検知失敗	合計
NAV で検知成功	31	2	33
NAV で検知失敗	9	0	9
合計	40	2	42

で検知できた．また，本手法でボットの動作を検知できたが，NAV で検知できなかった通信が 9 件存在した．これらのマルウェアは NAV がインストールされた環境下で実行を試行しても，NAV に何ら検知イベントとして発見されないマルウェア (あるいは，何らかの活動が見受けられると警告を発するが悪意のあるマルウェアと断定されないマルウェア) である．

今回のモデルを用いた評価では，提案手法を用いた検知が失敗し，NAV で検知が成功した通信が 2 件存在した．それぞれの通信の特徴を分析した結果を次に示す．

1. W32.Sasser.D

本手法で検知が失敗した 1 つめのマルウェアは W32.Sasser の亜種である．このマルウェアの通信には，転送終了後に，多量の ICMP の ECHO Request メッセージをインターネット上に送信するスキャン活動が含まれている．今回，モデル化に利用した通信情報には多量の ICMP を試行するマルウェアが含まれておらず，攻撃活動の検知に失敗した．

2. W32.Gobot.A

本手法で検知が失敗したもう 1 つのマルウェアは W32.Gobot の亜種であった．このマルウェアは，転送終了後に，管理サーバとの接続試行とみられる 4 つの DNS パケットを送信し，その解決に失敗した．今回のモデルの定義では $F2$(複数の DNS の解決に失敗) に当てはまるべき活動であるが，“30 秒以内に 5 回以上の DNS 要求を失敗” という条件に当てはまらず，検知に失敗した．

また，Tu 内に，本手法と NAV 両方で検知できなかったマルウェアは含まれていなかった．

6.5 考察

本結果では $C4$ の長期のセッションに着目した通信だけではなく，$C2$ のように類似した通信を繰り返し続けるセッションも観測された．これは，短期のセッションを定期的に通信する管理手法が能動感染型のマルウェアでも頻繁に用いられている．2007 年頃から増加している Web を介した受動感染型のマルウェアの大多数の管理通信は，このような定期的かつ短期的な通信を主流としている．管理通信が HTTP を多く用いる傾向は，ボットがノードの管理者が自ら HTTP の通信を要求しているように振る舞い，

活動の隠蔽化を目的とした活動であると考えられる．しかし，本手法ではこれらの差異が検知結果に影響しないことを評価によって示した．

今回評価に用いた通信データのうち，通信のモデル化に利用した通信情報である 97 件のマルウェアの活動が通信モデルから検知出来ることを示した．また，未知のマルウェアを含む通信情報についても 42 件中 40 件の検知のマルウェアを検知でき，検知できなかったマルウェアが 2 件存在した．これらを検知するためには，1 件については多量の ICMP 送信に関する攻撃の特徴を定義する，もう 1 件についてはより多くのマルウェア通信を解析し条件となる数値を適切に調整するなど，モデル改善の必要性が見られる．しかし，モデルを改善することで，これらのマルウエア通信の検知は可能である．

本章では提案手法が既存のボット通信を十分にモデル化できていることを示した．また，既存のボット通信を利用したモデルは未知のボット通信を十分に検知でき，既存研究において困難とされていた未知のボット亜種の検知に有効であると示した．

第7章　結論

本章では本研究をまとめ，本研究分野が取り組むべき今後の課題について示す．

7.1　総括

既存のセキュリティ対策手法の多くはシグネチャを必要とする．シグネチャを必要とする検知手法では常に亜種の出現するマルウェアの対策は困難である．また，ボットネットに着目した調査やDNSを用いた対策手法は，ボットの規模性や管理サーバの冗長化といった問題を抱えている．これらの既存の手法は，世界中に拡散して多くの亜種の出現するボットネットに対して十分な成果が期待できない．

この問題に対して本研究では，ボット通信をモデル化し，既知のボットを元としたボット亜種の効率的な検知を実現した．また，本研究を用いることで，ネットワーク管理者はボットネットの専門知識を持たなくとも，確実に検知したネットワーク内のボットを検知できる．これにより，既存の手法では難しかったボットへの対応が可能となり，インターネット上のボットによる脅威の低減が可能となる．

7.2　今後の課題

本研究ではインターネット上の脅威を低減するための手法について取り組んだ．しかし，攻撃者は常に脅威の対策を回避する手法の開発に注力しているため，マルウェアの検知の分野においてはいくつかの大きな課題がある．

7.2.1　複数の対策手法の連携

ボット通信の多様化はセキュリティインシデント検知において避けられない課題であり，今後も多くのボットが発生すると考えられる．また，検知手法にはそれぞれ第3章で述べた通りの特徴がある．インターネット上からマルウェアによる被害を低減するためには，インターネット規模における管理サーバの発見や，ノードにおける多様な管理視点に基づいた検知手法など複数の視点から取り組む必要がある．

7.2.2 全く新しいマルウェアへの対応

今回は実験データを用いて収集されたボットの通信の特徴を用いて，正しくボットが検知されることを示した．しかし，第1章で述べたとおり，ボットは経済モデルが成り立っていることから，新しい通信モデルを用いたボットネットの開発が進んでいくと考えられる．その際には，明らかに悪意のある通信を確実に検知できる方向を考え，検知できない活動を効率的にモデル化していく必要がある．

7.2.3 実ネットワークでの運用と評価

本論文中では，マルウェアのモデルがボットの検知に有用であることを示した．しかし，本研究を含む通信のモデル化や，第3章で挙げた DNS を利用したボットネットの検知については，正常なトラフィックの誤検知問題など解決すべき課題が残る．近年では Norton AntiBot などマルウェアのふるまいを抽象化した検知手法が製品として登場しており，実ネットワークにおける運用と評価が期待される．

7.2.4 環境に依存する脅威への対応

Zou らの報告 [42] によれば，一部のボットは外部への接続が制限された環境では正常な動作を行わないといわれている．そのため，第 4.4節で述べたように通信を軽減している環境下では実環境のボット通信を完全に再現しているとはいえない．鵜飼らの取り組み [43] は特定の条件下でしか活動しないボットの条件などを判別する静的解析手法が研究されている．今後，特定環境下でしか実行できないボットが増加する可能性を考慮し，静的解析と連携したボット通信の取得方法が期待される．

謝辞

本論文の執筆にあたり，本当に多くの方々にお世話になりました．この場をお借りして，お礼を述べさせていただきたいと思います．

まず，本研究を進めるにあたり，ご指導をいただきました慶應義塾大学 環境情報学部教授 村井純博士に感謝いたします．大変ご多忙な中，本研究のテーマや研究の進め方について，非常に貴重な意見を頂きました．

同学部教授 中村修博士に感謝いたします．私は氏から丁寧に研究の本質を考えるようご指導いただきました．

慶應義塾大学 政策・メディア研究科教授 武田圭史博士に感謝いたします．氏は私の研究を親身に考えてくださり，方向性や執筆に迷った際には大きな助けを私に与えてくださいました．

学部からの研究について常にご指導いただき，様々な議論を交わしあった慶應義塾大学 政策・メディア研究科 水谷正慶氏と白畑真氏に感謝いたします．お二人は私が研究室に入った当初から私の研究についての議論や様々な講義のために多くの時間を割いてくださいました．氏らと共に実験環境の構築や結果の考察に取り組むことで，セキュリティという研究分野に一層楽しく取り組めました．

また，かねてより研究を進める上でモービル広域ネットワークプロジェクトの先生方から多くの助言などを頂いてきました．楠本博之博士，Rodney D. Van Meter III 博士，鈴木茂哉氏，土本康生博士，重近範行博士，吉藤英明博士，朝枝仁博士，植原啓介博士，斉藤賢爾博士，三次仁博博士，稲葉達也氏，中根雅文氏に感謝します．

本論文執筆にあたっては慶應義塾大学徳田・村井・楠本・中村・高汐・重近・バンミーター・植原・三次・中澤合同研究プロジェクトの諸氏らからさまざまな励ましと助言，手助けをいただきました．

岡田耕司氏には本研究をはじめとして研究室内での生活における様々な課題について相談に乗っていただきました．氏は常に私の研究生活を励ましてくださいました．松谷健史氏の持つハードウェアに対する情熱と知識には強く刺激を受け，私の興味の分野を広げることができました．また松谷氏は本論文執筆中における多くの夕飯をサポートしてくださいました．お二人のサポートは私の研究生活に欠かせないものでした．本当にありがとうございました．

研究室の同期であり共に執筆に取り組んだ空閑洋平氏の新しい技術やツールに関する情報アンテナの広さは私に刺激を与え，研究生活を豊かな環境にしてくれました．特

に，氏の考案したトランプゲーム *charlatan* は論文執筆中に非常に良い気分転換の機会となりました．同様に執筆を共にした奥村祐介氏は常に場の空気を和ませてくれ，時にゲームのライバルとして，非常に充実した時間を与えてくれました．本多倫夫氏とは学部１年次から共に研究活動に取り組んできました．あの頃に苦労しながらも共に学んだ輪講の様子は昨日のように覚えています．みんな，ありがとう．

石原知洋氏，海崎良氏，堀場勝広氏と工藤紀篤氏は多くのネットワーク設計・運用の場と人脈を築く機会を私に与えてくださいました．これらの経験はネットワークを基盤とする私の研究活動において，非常に有意義なものでした．南政樹氏と小原泰弘博士には，私が研究室に入った時から多くのことをご指導いただきました．氏らは私にプログラミングやネットワーク・サーバ運用の技術からそれらの心構え，研究室での生活まで多くのことを教えてくださいました．氏らのご指導なくして，ここまで有意義な研究生活は過ごせませんでした．ありがとうございました．

研究室で活動を共にした神谷尚保氏，町田啓太氏，六田佳祐氏，波多野敏明氏，永山翔太氏，上原雄貴氏，黒宮佑介氏，遠峰隆史氏，川喜田佑介博士，江村桂吾氏，佐藤龍氏，佐藤泰介氏，三部剛義氏，久松剛氏，三島和宏氏，松園和久氏，中村友一氏，Achmad Basuki 氏，Mohamad Dikshie Fauzie 氏，中村遼氏，中里恵女史，小椋康平氏，尾崎隆亮氏に感謝いたします．特に六田佳祐氏と波多野敏明氏は本研究の手助けだけにとどまらず，研究室内での様々な取り組みを快く引き受けてくれました．また，永山翔太氏と上原雄貴氏は研究生活に笑顔を与えてくれ，論文執筆の面でも大きな手助けをくれました．小椋康平氏から伝えていただいたゲームの文化は私の心に大きなゆとりと新しい楽しみをもたらしてくれました．また，私は石原知洋氏と黒宮佑介氏が作成してくれた私の本論文執筆を心強くサポートするポスターを忘れません．そして，私は本論文の執筆中に私を癒し続けてくれた，水谷氏のペットであるでんすけの顔を忘れません．皆さん，ありがとうございました．

多くの友人と家族に対して感謝いたします．

新倉弥幸女史とは学位論文のテーマや執筆ついて互いに語り合いました．研究分野の違う女史との会話は非常に興味深く，私に刺激を与えてくれました．

竹尾明子女史は私をいつも見守り，励ましつづけ，女史自身が多忙であった時も私を支え続けてくれました．心から感謝いたします．本当にありがとう．

最後に私の健康を気遣い，学部時代から影ながら研究を大いに支えてくれた父 千明，母 艶子と妹 香に感謝します．

皆様の御指導と御助力がなければ本論文は成し得ず，私の学生生活もこれほどまでに充実したものにはなりませんでした．本当にありがとうございました．

2009 年 2 月
金井 瑛

参考文献

[1] Nikkei Business Publications, Inc. ヤフーがクラッカの攻撃を受けて 3 時間機能停止、イーベイ、バイ・ドット・コムなども相次いで被害 , February 2000.

[2] Swa Frantzen. Clickbot, May 2006. http://isc.sans.org/diary.php?storyid=1334.

[3] Tony Carothers. Large botnet in the netherlands taken down, Oct 2005. http://isc.sans.org/diary.php?storyid=742.

[4] Trend Micro. ボットネットの脅威とソリューション：フィッシング, Oct 2006. `http://jp.trendmicro.com/imperia/md/content/jp/threat/specificsolutions/trendphishingwhitepaper.pdf`.

[5] IPA. ボット感染予防推進事業の開始について, Feb 2008. `http://www.ipa.go.jp/security/isg/bot.html`.

[6] Paul Bacher, Thorsten Holz, Markus Kotter, Georg Wicherski. Know your Enemy: Tracking Botnets -Using honeynets to learn more about Bots-, March 2005.

[7] Barford, Paul and Yegneswaran, Vinod. An Inside Look at Botnets. *Advances in Information Security*, Springer 2007.

[8] J. Reynolds, J. Postel. Request for Comments: 1700: ASSIGNED NUMBERS, October 1994.

[9] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. RFC 1928: SOCKS Protocol Version 5, March.

[10] Nikkei Business Publications, Inc. Nikkeibp: Telecom-isac japan が「ボットネット」の実験結果を報告, April 2006.

[11] Nikkei Business Publications, Inc. Nikkeibp:［network 調査隊］「ボットネット」の正体を探る, January 2006.

[12] 日本国総務省. 平成１７年「通信利用動向調査」の結果, May 2006.

[13] CERT Advisory CA-2003-20 W32/Blaster worm.
http://www.cert.org/advisories/CA-2003-20.html, Aug 2003.

[14] CERT Incident Note IN-2001-09 “Code Red II: Another worm Exploiting Buffer Overflow In IIS Indexing Service DLL”.
http://www.cert.org/incident_notes/IN-2001-09.html, Aug 2001.

[15] ClamAV and Clam AntiVirus. Clam antivirus, 2007. `http://www.clamav.net/`.

[16] Cisco System. Ironport c660 -大企業 isp 向けメールセキュリティアプライアンス, 2007. `http://www.ironport.com/jp/products/ironport_c660.html`.

[17] Wietse Venema. Postfix.

[18] Inc. Sendmail. sendmail.

[19] Google Inc. Google search. `http://www.google.com`, 2008.

[20] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. The Ghost In The Browser Analysis of Web-based Malware. In *Proceedings of the First USENIX Workshop on Hot Topics in Understanding Botnets*, 2007.

[21] SURFnet. Welcome to the SURFnet website. http://www.surfnet.nl/info/en.

[22] Antoine Schonewille, Dirk-Jan van Helmond. The Domain Name Service as an IDS, Feb 2006.

[23] J. Oikarinen, D. Reed. RFC 1459: Internet Relay Chat Protocol, May 1993.

[24] Christian Kreibich and Jon Crowcroft. Honeycomb - Creating Intrusion Detection Signatures Using Honeypots. In *Proceedings of the Second Workshop on Hot Topics in Networks (Hotnets II)*, Boston, Nov 2003.

[25] Sumeet Singh, Cristian Estan, George Varghese and Stefan Savage. The EarlyBird System for Real-time Detection of Unknown Worms. August 2003.

[26] Sumeet Singh, Cristian Estan, George Varghese and Stefan Savage. Automated Worm Fingerprinting. December 2004.

[27] 金井 瑛, 水谷 正慶, 白畑 真, 南 政樹, 村井 純. IDS と連携した高速に伝播するワームのシグネチャ自動生成機構の設計と実装. **第 *13* 回マルチメディア通信と分散処理ワークショップ**, November 2005.

[28] Martin Roesch. SNORT-LIGHTWEIGHT INTRUSION DETECTION FOR NETWORKS. *USENIX LISA ‘ 99 Conference*, 1999.

[29] Symantec Corporation. Symantec WWW page. http://www.symantec.com/.

[30] Symantec. Norton antivirus 2009. `http://www.symantec.com/norton/antivirus`, 2008.

[31] Symantec. Norton antibot. `http://www.symantec.com/norton/antibot`, 2008.

[32] サイバークリーンセンター. 2008 年 10 月度 サイバークリーンセンター活動実績, Oct 2008. `https://www.ccc.go.jp/report/200810/0810monthly.html`.

[33] 松木 隆宏. 時系列分析による連鎖感染の可視化と検体種別の推測. **マルウェア対策研究人材育成ワークショップ** *2008* **論文集**, 2008.

[34] Nepenthes Development Team. Nepenthes - finest collection - Web Page. http://nepenthes.mwcollect.org/.

[35] Sun Microsystems. Virtualbox, 2008. `http://www.virtualbox.org/`.

[36] 水谷 正慶, 武田 圭史, and 村井 純. 通信の状態遷移に着目したボット活動の調査. **マルウェア対策研究人材育成ワークショップ** *2008* **論文集**, 2008.

[37] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong and Wenke Lee. BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. *USENIX Security Symposium*, 2007.

[38] tcpdump.

[39] 金井 瑛. フロー順序に着目したボット検知手法の提案と検証. **慶應義塾大学 環境情報学部 卒業論文**, 2006.

[40] J. Postel and J. K. Reynolds. Rfc 959, 1985. `http://tools.ietf.org/html/rfc959`.

[41] K. Sollins. THE TFTP PROTOCOL (REVISION 2), July 1992.

[42] Cliff C. Zou and Ryan Cunningham. Honeypot-aware advanced botnet construction and maintenance. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 199–208, June 2006.

[43] 鵜飼 裕司, 小林 偉昭, and 中野 学. 脆弱性を利用した標的型攻撃のための解析ツール. **マルウェア対策研究人材育成ワークショップ** *2008* **論文集**, 2008.

付 録A　通信データの再構成

A.1　目的

本論文では，能動感染型ボットの活動によって生じる通信を取得するために，ハニーポットソフトウェアNepenthesを用いた．Nepenthesは本論中で紹介したとおり，多くの脆弱性エミュレート機能を備え，ダウンロードの実行が可能なハニーポットである．Nepenthesは仮想マシンや実機を用いる高インタラクションハニーポットとは異なり，脆弱性の処理と脆弱性によって発生するマルウェアの転送をエミュレートし，その後の通信には関与しない．すなわち，Nepenthesの通信を監視するとマルウェアの活動のうち攻撃と転送についての情報を得られる．しかし，その後の管理と活動については有益な情報を得られない．本章では，Nepenthesでマルウェアを取得する際の通信と，取得されたマルウェアを実行環境で実行した際の通信を結合する工夫について述べる．

A.2　能動型感染ボットの通信取得機構

ハニーポットはボットネットの研究を進める上で有効なツールである．運用コストや法的な問題から低インタラクション型のボットネットは研究に用いやすい．しかし，一般的な低インタラクション型の環境から得られる情報は，セッション型ボット検知機構に対して十分な情報を与えない．

一般的な低インタラクション型ハニーポットを用いたボットトラフィック収集の環境を図A.1に示す．図中では，IPアドレスIPaを持つマシン上で低インタラクション型ハニーポットが稼働している．ハニーポットの稼働しているマシンの通信はすべてトラフィックミラーポイントMPaでミラーリングされ，蓄積される．MPaのトラフィックダンプデータにはハニーポットが対応したすべての攻撃に対してのトラフィックが蓄積される．ハニーポットで蓄積したマルウェアは手動でIPアドレスIPbを持つ実行マシンにコピーする．法的な問題から実行マシンからのトラフィックは管理者の設定したポリシーによって，フィルタアウトされる．また，実行マシンの通信はトラフィックミラーポイントMPbによってミラーリングされ，蓄積される．MPbのトラフィックダンプデータは筆者のモデルを用いたボット検知機構などのマルウェアの通信を利用してボットを検知する機構には有効である．対して，MPaのトラフィックダンプデータ

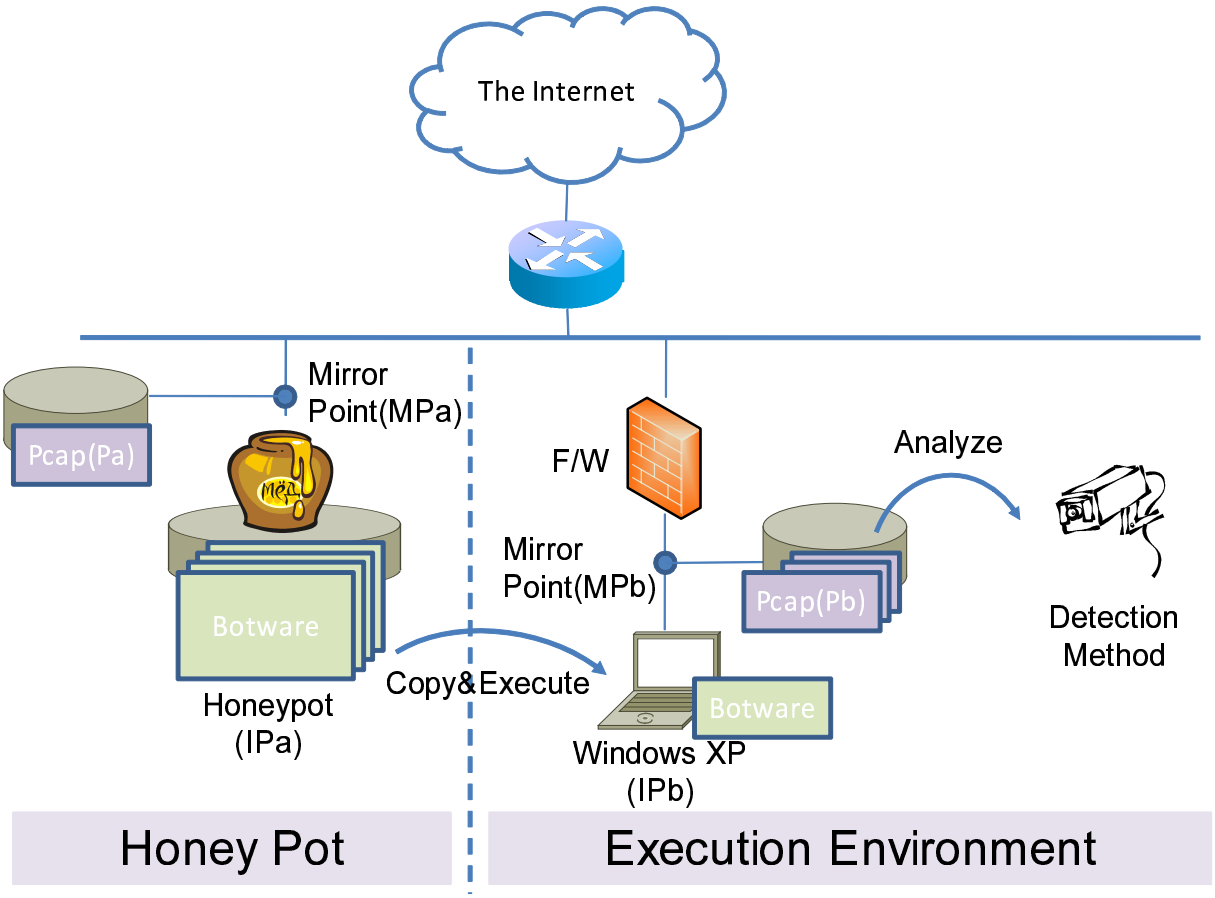

図 A.1: マルウェアトラフィック収集のための一般的なハニーポット環境

にはマルウェアのダウンロードまでのトラフィックのみが含まれるため，筆者の機構などには有効な情報とならない．

A.3 問題点

第 A.2節ではハニーポットの有用性と，ハニーポットと実行マシンを用いたマルウェアの通信情報収集環境について述べた．実際のマルウェア通信を再現するためには MPa の通信と MPb の通信を 1 つに結合する必要がある．

2 つの PCAP ファイルのパケットを結合する既存の実装として *tcpmerge* が存在する．*tcpmerge* は 2 つの PCAP ファイル内のパケットの時間情報を保持したまま，パケットを結合する．しかし，結合に際して，**PCAP ファイルの持つ情報の差**の問題がある．また，問題を解決しトラフィックを結合したのみでは **IP アドレスの差**と**時間情報の差**の 2 つの問題が発生する．これら 3 つの問題は，通信を用いたネットワークイ

ンシデント検知機構の評価に大きな影響を及ぼす.

A.3.1 PCAP ファイルの持つ情報の差

ハニーポットのトラフィック MPa はすべてのマルウェアに関するトラフィックを保持する．しかし，実行環境のトラフィック MPb は，マルウェアごとに分割されたトラフィックデータを保持する．そのため，マルウェア毎のトラフィックを得るには，ハニーポットトラフィックからマルウェア毎のトラフィックを抜き出す必要がある．この問題を図 A.2に示す.

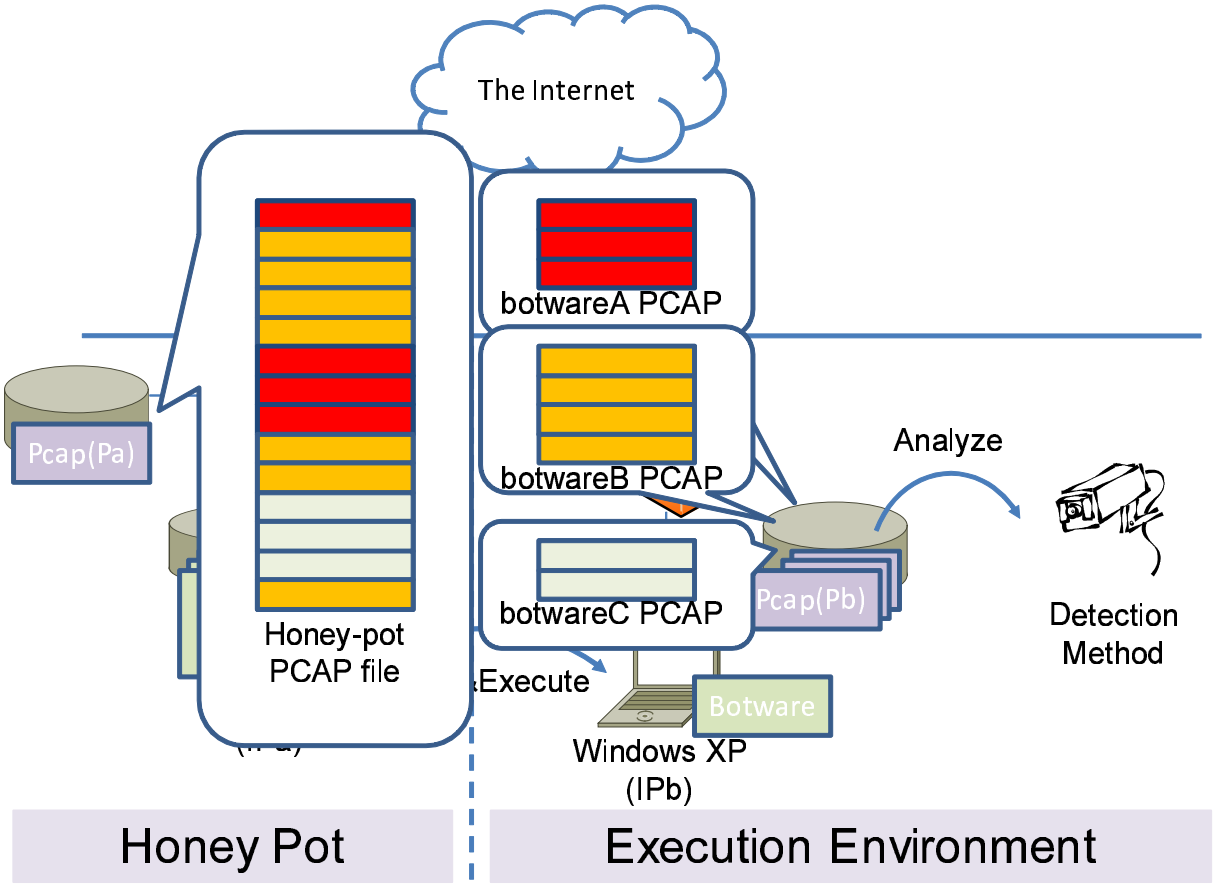

図 A.2: 実環境との差

A.3.2 IP アドレスの差

ハニーポットマシンと実行環境マシンは保持する IP アドレスが異なるので，IP アドレスを用いて通信ノードを特定すると，実際とは異なったマシンが通信しているように観測される．図 A.1の例では，ハニーポットトラフィックでは IPa が，実行環境トラフィックでは IPb がそれぞれ外部と通信をしているトラフィックとなる．本論文で述べた種類のネットワークベースの検知機構では，内部ノードのフローの順序を利用して検知するが，IPa と IPb の行う別の通信として扱われるため，正しい検知が実現できない．

A.3.3 時間情報の差

実行環境トラフィックは，マルウェアを収集した直後に収集されていないので，実際の動作と時間情報に差が出る．この様子を図 A.3に示す．ただし，図中の用語を表 A.1

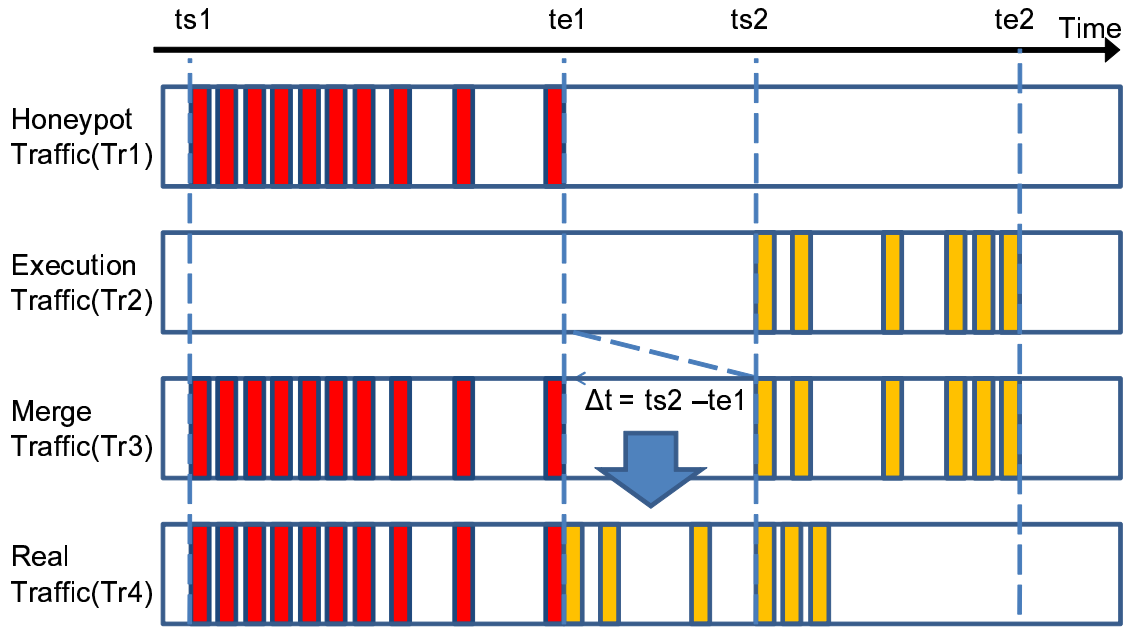

図 A.3: 実時間と pcap に含まれる時間の差分

のように定義する．Δt は式 A.1 のように表わされる．

$$\Delta t = ts2 - te1 \tag{A.1}$$

実際のトラフィックでは一般的に，マルウェアの転送終了直後にマルウェアが実行されるため，$te1$ と $ts2$ の差は十分に小さい．しかし，一般的な実行環境においては，マルウェアのダウンロード時間とマルウェアの実行に大きな時間差が生じるため，$Tr2$ の全トラフィックは実際に比べて Δt だけ遅れた時間をもつパケットとなる．

表 A.1: タイムラインで用いられる記号の解説

項目名	内容
ts1	ハニーポットトラフィックの開始時刻
te1	ハニーポットトラフィックの終了時刻
ts2	実行環境トラフィックの開始時刻
te2	実行環境トラフィックの終了時刻
Tr1	ハニーポットトラフィック
Tr2	実行環境トラフィック
Tr3	時間情報を保持したまま結合したトラフィック
Tr3	実際のトラフィック

本論文で述べたモデル化による検知機構を含む多くのトラフィック監視するセキュリティ機構では，フロー間の間隔が長いとそれらを連続したフローとして扱わない．そこで，$Tr2$ は Δt だけ時間を早くして $Tr1$ と結合し，実際のトラフィックに近づける必要がある．

A.4 実装

A.4.1 Nepenthes へのパッチ

本節では，低インタラクションハニーポットである Nepenthes に対して加えた変更について述べる．

Nepenthes の構成モジュール

Nepenthes は C++で書かれたハニーポットである．Nepenthes のボット収集に関する機能は大きく 5 つのモジュールに分かれている．各モジュール間の全体像を図 A.4に示す．図に示したモジュール以外にも nepenthes はログの出力機構や，イベントハンドリングモジュールなどを含む．

Socket Module
本モジュールは Nepenthes 起動時に読み込んだ Vulnerable モジュールへの通信を受け付ける Socket を listen する．また，Download モジュールからのダウンロード要請を処理する．Socket モジュールとは Msg クラスによりデータがやり取りされている．

Vulnerable Module
本モジュールは socket モジュールと Msg クラスによりデータをやりとりし，脆弱性を

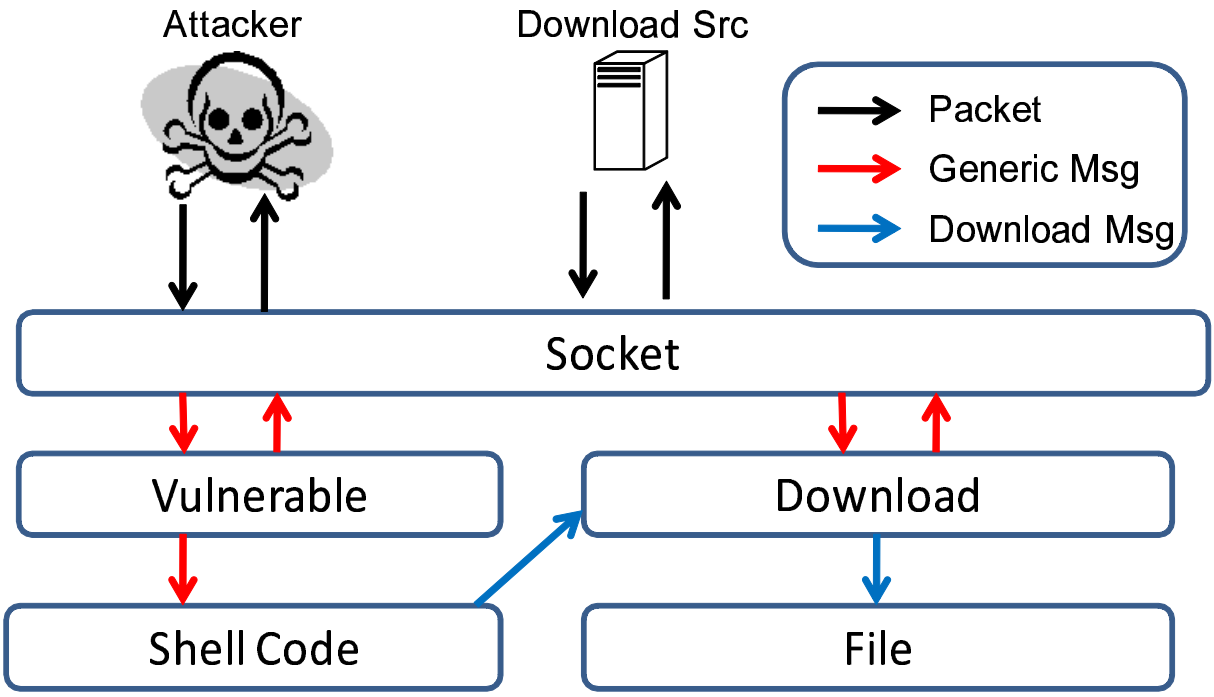

図 A.4: Nepenthes における実装の差分

エミュレートする．多くの脆弱性では，シェルコードを実行するために，受け取ったデータを Msg クラスで ShellCode モジュールに伝える．

Shell Code Module

本モジュールは主に，送り込まれたシェルコードの実行をエミュレートする．多くのエミュレートの結果，Donwload モジュールに対して，Download クラスによってダウンロードを要求する．

Download Module

本モジュールは，ダウンロード要求を処理する．ShellCode モジュールからのダウンロード要求を受け付け，Msg クラスによって Socket モジュールと通信する．ダウンロードが終了すると，Download クラスによって，File クラスへデータを通達する．

File Module

本モジュールは Download モジュールからの Download クラスのデータを受け取り，ダウンロードが終了したデータをファイルに保存する．

Nepenthes へのパッチ

各モジュールに対してイベント発生時固有の識別子を生成・継承するように Nepenthes を変更した．この結果，図 A.5 のようにログが出力される．

```
(12102007 22:21:36 info handler dia) Time: 1192195296,
Info: KNOWN DCOM(2) ATTACK RECEIVED, UniqueId: 1,
Local: 203.178.143.29:135, Remote: 203.174.215.156:1258
12102007 info mgr submit Time: 1192195331

(12102007 22:21:36 info down handler module) Time: 1192195296,
Info: LINK(1) DOWNLOAD START, UniqueId: 1,
Local: 203.178.143.29:33286, Remote: 203.174.215.156:54791

(12102007 22:21:36 info handler dia) Time: 1192195296,
Info: BLINK DOWNLOAD START, UniqueId: 1,
Local: 203.178.143.29:33286, Remote: 203.174.215.156:54791

(12102007 22:22:11 info mgr submit) Time: 1192195331,
Info: SUBMITED, UniqueId: 1,
FileName: 2aa59ba4251795deda72738d1c67be7c,
FileType: MS-DOS executable PE for MS Windows (GUI) Intel 80386 32-bit
```

図 A.5: パッチを適応した Nepenthes により出力されるログの例

A.4.2 PCAP Address Replacer

PCAP Address Replacer は C 言語で記述されたパケット内部の IP アドレスと MAC アドレスを書き換えるコマンドラインソフトウェアである．パケットに含まれるコマンドラインから指定したアドレスを置き換え，また，IP チェックサムおよび，L4 のチェックサムを再計算する．IP は IPv4 および IPv6 をサポートする．また，L4 プロトコルは TCP と UDP をサポートする．

PCAP Address Replacer は PCAP IP Address Replacer と PCAP MAC Address Replacer に分かれており，それぞれ図 A.6 と図 A.7 に示したコマンドラインで置換する．

A.4.3 PCAP Appender

PCAP Appender は C 言語で記述された 2 つの PCAP ファイル内のパケットの時間を調整し，それぞれの PCAP ファイルを結合するソフトウェアである．本ソフトウェアは，図 A.8 に示したコマンドラインで結合を実行する．

本ソフトウェアは第 A.3.3節で示したように，infile1 のトラフィックを $Tr1$，infile2 の

pcap_ipreplacer -i*infile1* -o*outfile* -b*before-ip-address* -a*after-ip-address*
-i: Input PCAP file :入力 PCAP ファイル
-o: Output PCAP file: 出力 PCAP ファイル
-b: Replace IP address: 置換対象の IP アドレス (ipv4/ipv6)
-s: Replace with IP address: 置換した後の IP アドレス (ipv4/ipv6)

図 A.6: pcap_ipreplacer の利用方法

pcap_macreplacer -i*infile1* -o*outfile* -b*before-mac-address* -a*after-mac-address*
-i: Input PCAP file :入力 PCAP ファイル
-o: Output PCAP file: 出力 PCAP ファイル
-b: Replace MAC address: 置換対象の MAC アドレス
-s: Replace with MAC address: 置換した後の MAC アドレス
MAC アドレスは:あるいは-あるいは. で区切られた 6 フィールドあるいは 3 フィールドの文字列

図 A.7: pcap_macreplacer の利用方法

トラフィックを $Tr2$ として扱い，$Tr2$ の各パケットの時間軸を $-\Delta t$ ずらすことで，$Tr4$ に近いトラフィックを生成して，outfile に書き出す.

A.5 考察

本機構はハニーポットトラフィックと実行トラフィックを合成することで，実際のトラフィックに似たトラフィックを生成する機構である．表 A.2に，筆者が提案した手法あるいは一般的なネットワークベースの検知手法にたいして，実際のトラフィックインシデントの検知に影響が生じるかを示した.

IP Address

IP アドレスは，本手法によって置換されるため，実際のトラフィックと変わらない.

Protocol Type

L4 のプロトコルタイプは，一切の影響を受けないため，実際のトラフィックと変わらない.

Packet Length

筆者の提案手法では利用しない項目である.

ネットワークスタックの問題により，完全に実際のトラフィックとは一致しない.

Start Time

フローの開始時刻は最初の攻撃のパケットであるため，実際のトラフィックと変わら

pcap_appender -1*infile1* -2*infile2* -o*outfile*
-1: Merge base PCAP: 結合する
-2: Merge PCAP file to *infile1*: *infile1* に結合する PCAP
-o: Output PCAP file: 結合した結果を出力するファイル名

図 A.8: pcap_appender の利用方法

表 A.2: 通信合成による検知手法への影響

	提案手法への影響	一般的な影響	影響
IP Address	-	-	
Protocol Type	-	-	
Packet Length	UNUSE	o	Network-Stack issue
Start Time	-	o	Network-Stack issue
End Time	-	o	Network-Stack issue
pps	-	o	Network-Stack issue
bps	-	o	Network-Stack issue
port number	-	-	Ephemeral port issue
Timeout	perhap	perhap	We can solve this problem
DNS PACKET	-	-	
L7 payload	UNUSE	-	
TCP FLAG	UNUSE	-	
WindowSize pattren	UNUSE	o	Network Stack issue
PacketSize pattren	UNUSE	o	Network Stack issue

ない.

End Time

ネットワークスタックの問題により，完全に実際のトラフィックとは一致しない．今回の提案手法では無視できる程度と考える.

pps/bps

ネットワークスタックの問題により，完全に実際のトラフィックとは一致しない．今回の提案手法では無視できる程度と考える.

port number

ポートは一切の影響を受けないため，実際のトラフィックとは変わらない．ただし，ハニーポットが外部のノードに接続する際は，OS の実装によって，Ephemeral Port Number が異なる可能性をある.

Timeout

ネットワークスタックの問題により，フロー間の間隔が若干長くなる可能性はあるが，今回の提案手法では無視できる程度と考える．

DNS Packet

DNS パケットには特に影響がないため，実際のトラフィックと変わらない．

L7 Payload

今回の提案手法では利用しない項目である．

アプリケーション層のペイロードには影響がないため，実際のトラフィックと変わらない．ただし，ペイロード内にハニーポットあるいは実行環境マシンの IP アドレスを含み，検知手法がその項目を重要視する場合は影響がある．kanai の提案手法では無視できる．

TCP FLAG

kanai の提案手法では利用しない項目である．

TCP フラグには特に影響がないため，実際のトラフィックと変わらない．

WindowSize/PacketSize Pattern

kanai の提案手法では利用しない項目である．

ネットワークスタックの問題により，完全に実際のトラフィックとは一致しない．