平成16年度 修士論文

トラフィック観測に基づいたファイアウォールルール制御の自動化手法

電気通信大学 大学院情報システム学研究科

情報システム設計学専攻

0250041 仁平 幸子

指導教官 多田 好克 助教授
星 守 教授
田野 俊一 教授

提出日 平成16年8月19日

目 次

図 目 次

表 目 次

第 1 章

はじめに

近年、無線 LAN や Dynamic Host Configuration Protocol(以後 DHCP と表記)などによって構成される可変的なネットワークが普及し、一般的なものとなりつつある。これにより、誰でも自由かつ簡単に組織内部にホストを接続できるというメリットが生まれた。しかし同時に、内部の接続構成が管理者にとって把握困難になるというデメリットも発生している。

これまで組織内部のネットワークは管理者が把握したホストのみ固定の IP アドレスで設置され、管理者権限でインストールされているアプリケーションのみ使用を許可されていた。そのため、内部ネットワークにはある一定の安全が保たれていた。しかし現在、可変的ネットワークが一般的になり、自由に私物のノートパソコンなどを持ち込めることで、内部ネットワークは信頼できるという前提条件が崩れて来ている。

このような可変的ネットワークでは、ネットワークを介して伝播するウイルスやワームが深刻な問題となっている。ひとたび感染ホストが内部ネットワークに接続されると他のコンピュータにまでも感染を広げ、外部に向かって攻撃活動を開始する。内部ネットワークの利用者自身が攻撃者になっているという意識を持ち合わせていないため、ウイルス、ワーム感染ホストの特定を行うのも管理者の仕事の一部となっている。管理者はこれまでのように外部からの攻撃・侵入だけでなく、内部から攻撃者を出さないことにおいても細心の注意を必要とされる。

このように、近年のネットワーク利用形態の変化は、ネットワーク管理者にとっ

て管理のための負担を増加させている。ネットワークの規模が大きく、可変的であればあるほどネットワーク構成の把握や、ウイルス感染ホストの特定が困難になる。

この増大するネットワーク管理者の負担を削減するために、夜間や不在時に管理者に代わって一時対応を行うシステムが必須である。しかし現状では、自動対応の判断基準となる検知システムの精度自体がまだ低い。それに加え、誤った検知の際に、接続の切断など極端な対応を行ってしまった場合、逆にその自動対応がサービス妨害となりうる危険性も秘めている。そのため、より柔軟な判断と対応を行う必要がある。

本研究では、平常時のトラフィック観測結果の統計を取り、それを元にウイルス、ワームの伝播活動の可能性がある異常な通信を発見するシステムの設計および実装を行った。また、ウイルス、ワームの可能性がある異常な通信を検知した際、ファイアウォールのルールを自動的に変更し、該当通信に関してのみ通信制限を行い、他通信への影響を最小限に抑える機能を加えた。

平常時のトラフィック観測は組織内部から外部への通信を対象とし、ゲートウェイ上で行う。観測を継続して行うことにより、内部ネットワークについてホストごとに特徴的な接続試行回数の統計情報を作成する。

本システムにより、これまでネットワーク管理者が手動で行っていたログ調査や、異常な通信が認められた際に行っていたファイアウォールのルール変更が自動化され、管理者の負担が軽減される。自動対応のためには、通常の通信と異なるウイルスの特徴を把握する必要がある。そのため、数千台のホストを抱える既存ネットワークにおける平常時のモニタリングを行った。また、その結果と比較するために閉鎖的な実験ネットワーク環境で実際にウイルスに感染しているホストを用意し、ウイルス感染活動のモニタリングを行った。それらの結果から、短時間での接続試行回数が非常に多いというウイルスの感染活動のもっとも顕著な特徴を導き出した。また、多量の接続試行を行っているホストの通信をゲートウェイマシン上で

制限することにより、外部への影響を最小限に抑えることができることを実証し、これまでのウイルス拡散による基幹ネットワークの輻輳を回避できる可能性について述べた。

本論文は 次のような構成となっている。第 2 章では、可変的ネットワーク普及とそれに伴う管理コストの増加、内部からの脅威について紹介し、本研究の目的を述べる。第 3 章では、侵入検知システム (Intrusion Detection System: 以後 IDS と表記) などの既存システムの紹介と問題点、組織内部からの攻撃防止に関する関連研究について述べる。第 4 章では、本システムで規制すべき通信を定義し、実際のウイルス、ワームを用いた実験結果と実在ネットワークでの利用傾向との比較を行いウイルス、ワーム伝播特徴の抽出を行った。第 5 章では既存システムの問題点と第 4 章で得られたウイルス、ワーム伝播の特徴を元に、本システムの設計方針と設計目標について述べる。第 6 章でシステム実装の詳細と動作、利用方法について述べる。第 7 章では、実験環境での本システムを用いた自動対応実験を行った結果と、本システムの今後の展望について述べる。最後に第 8 章でまとめとする。

第 2 章

背景と目的

2.1 背景

無線 LAN や DHCP などの可変的なネットワークが一般的なものとなるにつれて、これまで『内部からの通信は安全である』というネットワークセキュリティの前提条件が崩壊しつつある。これにより管理者は外部からの攻撃・侵入への対策だけでなく、組織内部からの脅威についても目を向けなくてはならなくなった。

しかし、可変的ネットワークにおいて管理者がネットワーク構成を把握するのは、内部ホストの接続が固定的であった頃に比べ非常に難しくなった。日々、新しいホストが繋がり、別なサブネットへ移動することもあるため、その変化はネットワーク全体を統括すべき管理者に見えにくいものとなってきている。

また Network Address Translation(以後 NAT と表記) 技術により、グローバルアドレスを持たなくても、内部に広大なネットワーク空間を持つことが可能になり、対外接続のゲートウェイとなるホストの下には、さらに各部署などのゲートウェイが子、孫状に設置され、数多くのホストが接続されるようになった。計算機自体の低価格化もこれに加わり、組織に常設された計算機以外にも、個人所有のノートパソコンなどが接続される機会も増えた。ネットワークに接続されるホストの増加により、ゲートウェイなどで取得されている通信のログサイズも膨大なものとなってきている。そのため管理者はログの示す状況の把握が困難になってしまい、ネットワークに発生した異常の特定、対策までに総じて時間がかかり、ネット

ワークを使用する他の業務にも支障が出る。

こういった管理の困難に加え、近年ネットワーク経由で様々な感染方法を駆使して広がり続けるウイルス、ワームの危険性がある。これは組織内部ネットワークを利用する人間が意図しないうちに、自らが攻撃者になってしまう可能性をはらんでいる。

2003 年 1 月 25 日に発見された Slammer ワーム [1] の感染流行時に見られたようなネットワーク基幹全体での輻輳発生は、組織の対外的な信用を失墜させるだけでなく、全インターネット利用者のサービス不能状態を引き起こす結果にも繋がる。Slammer は、大量のネットワークパケットを生成し、サーバやルータをオーバーロードさせるため、 20% 近くものパケットロスを発生させた。同日には Slammer により 13 台あるインターネット・ルートネームサーバのうち 5 台がダウンした [2]。そして全世界では、少なくとも 75,000 台ものマシンが感染し、金融期間の ATM が使用不可能になるなどの深刻な問題が発生した [3]。ウイルスの大規模、高速拡散はもはや理論上の脅威ではなく現実のものとなってきている。

内部ネットワークに関する管理者の日常的業務を図 2.1 に示した。もし内部ネットワークにウイルス、ワームが発生した場合、下記の手順で対応を行わなくてはならない。

1. IDS からのアラートメール受信
2. IDS のログ確認
3. ファイアウォールのログ確認
4. ファイアウォールのルール設定変更
5. 該当通信を行っていたホストを内部ネットワークから切断
6. 該当通信を行っていた一般ユーザへの注意喚起

さらに、ユーザが自分で対応する知識を持たなければ、対策・ウイルス駆除まで行う必要も出てくる。感染ホストが多数あれば、同じ工程を何度も繰り返すことになる。

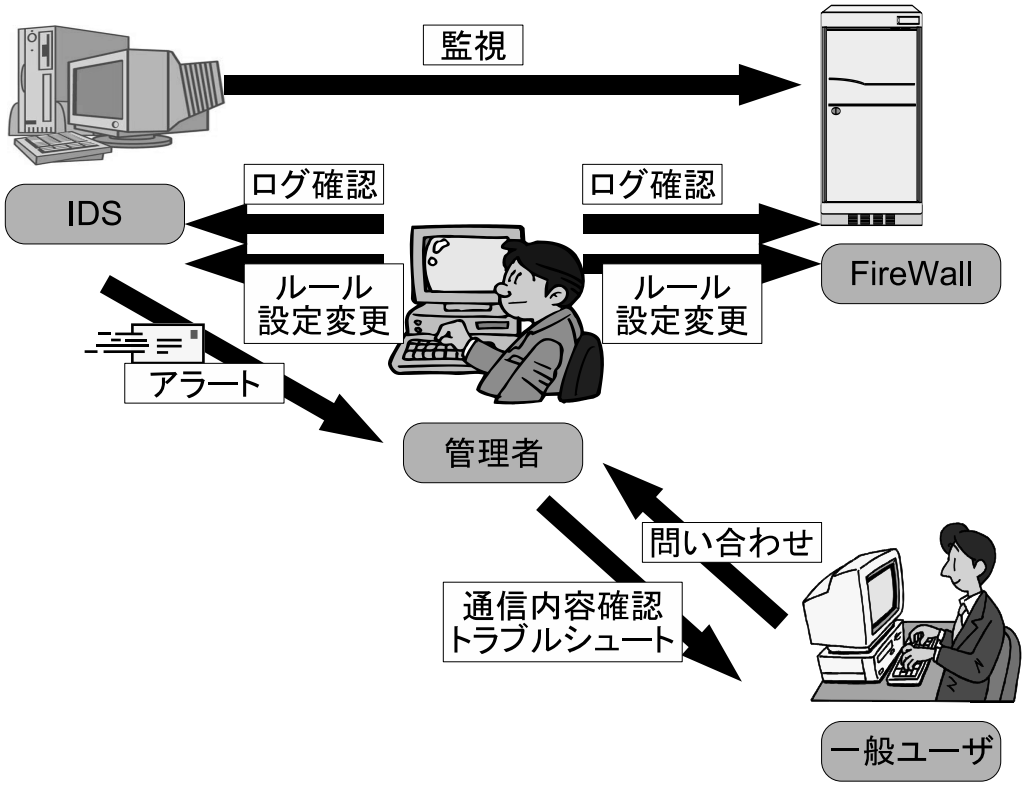

図 2.1: 管理者の業務

2.2 研究の目的

本研究の一番の目的は、ネットワークを運用していく人間の負担を取り除くことである。

特に組織ネットワークの内部から次々と発生するウイルス、ワームの脅威は管理者やユーザの貴重な時間を無駄に消費するだけでなく、攻撃トラフィックでネットワーク資源を浪費する。その事態を回避すべく、ウイルス、ワームの被害をでき

る限り管理者の手を煩わすことなく自動対応で制限すべきである。

そのためには現在のネットワーク管理で何が負担の原因、遠因となっているのかを明確にする必要がある。

本研究を開始するにあたり、実際の通信関連企業のネットワーク管理で問題になっている点を調査した。その結果を以下に列挙する。

- 人的制限
  - 管理者の管理能力不足
  - 管理者の業務時間の制限
- 物理的制限
  - ログ保存のためのストレージの制限
  - 計算機資源の制限
- 技術的制限
  - IDS の検知精度の限界

上記の問題を改善するようなシステムを構築することが本研究の第一目標である。それを実現するため、以下のような要件を考えた。

- 人的制限への対応
  - 簡単な導入・管理方法
  - ログ処理の自動化
  - ログ統計処理結果にもとづく自動対応
- 物理的制限への対応
  - ログの簡素化によるストレージ節約

- 必要最低限の動作

- 技術的制限への対応
  - ウイルス、ワームの動作に着目した検知システム
  - 穏やかな通信規制の実施

これらの要件は第 5 章で述べる設計においても、重要な基本方針となる。これらの人的・物理的・技術的な制限を排除し、管理者にとって導入・運用が行いやすい、負担の少ないシステム構築を目指す。

第 3 章

既存システムと関連研究

3.1 既存システム

現在、組織内部からのウイルス、ワームなどの脅威を検知するため用いられている既存システムがある。この章では既存システムについて概要を説明する。それらの既存システムは単体で使われることもあるが、複数組み合わせて、各メリットとデメリットを補う形で運用される場合が多い。

ここではよく用いられる既存システムとして、 侵入検知システム (Intrusion Detection System :以後 IDS と呼ぶ）とファイアウォールについて、それぞれ分けて説明を行う。

3.1.1 IDS

IDS とは

IDS は侵入検知システムの略で、ネットワーク上に流れている不正パケットを検知し、管理者に警告を出すシステムである。ネットワーク上の不正パケットには、ウイルス、ワームも含むことが多い。いくつかの方式や種類があるが、現在のところそれぞれに運用・導入を困難にするような短所も見られる。次では IDS の分類とそれぞれの問題点を示した。

IDS の種類と問題点

IDS は設置箇所で二種類に分類することができ、それぞれについて検知手法でさらに二種類に分類することができる。IDS は現在ホスト型とネットワーク型の二種類に分類されている。

ホスト型 IDS は監視対象ホストにインストールされ、自分のホストに流れてきたパケットやシステムファイル、ログサイズ、パーミッション、改ざん、ユーザの操作履歴などを監視することができる。ホスト型 IDS の場合、監視対象となるホストに直接インストールされるため本来そのホストが提供するサービス以外に余計な負担をかける恐れがある。

ネットワーク型 IDS はネットワーク上を流れるパケットを収集し、不正パケットを検知する。ネットワーク型 IDS はサービスを提供するホストではなく、監視対象ホストの途中経路やネットワークのゲートウェイとなる箇所に設置され、パケットをモニタリングし、異常を検知する。そのため、監視対象ホストマシンのリソースを消費することはなく、監視対象ホストは自己の提供するサービスのみに専念できる。

検知方法は Signature-Based と Anomaly-Based の二種類があるが、それぞれに導入・運用を困難なものにする問題点が存在する。

Signature-Based IDS はシグネチャ(ルールパターン) を持ち、収集したパケットと照合してマッチしたものを攻撃と認識する。ウイルス、ワームについてのシグネチャも個別に用意されている。しかし Sinature-Based IDS はシグネチャが古ければ検知できなかったり、あるいはシグネチャのチューニングを行わないと、関係のない警告でログが膨大な量になってしまうこともある。またシグネチャの数自体も膨大であるため、管理が難しい。シグネチャの意味を管理者が理解できないとネットワークに合せたチューニングも行うことができない。

Anomaly-Based IDS は通常のパケットやユーザの普段の操作をプロファイルとして記憶しておき、それから外れた操作やパケットを攻撃と認識する。さらに

Anomaly-Based IDS は特定のシグネチャを持たないで、パケットやユーザのふるまいの記録 (プロファイル) によって判断するため、誤検知率が非常に高い。

それぞれの欠点を克服し、利点を活かすために Signature-Based IDS と Anomaly-Based IDS の特徴を融合させたハイブリット型の IDS も存在する。しかし IDS 全般の問題として、運用自体が難しく導入・運用を行っていくにあたって、管理者の知識と根気が必要となる。知識不足な管理者や怠惰な管理者が導入した場合まったく意味をなさないシステムとなってしまったり、逆に負担が増加する可能性が高い。

3.1.2 ファイアウォール

ファイアウォールとは

ファイアウォールとは、ネットワーク上のパケットを判別し、設定されたルールをもとに、許可・拒絶を行うシステムである。ヘッダの情報のみを監視し、許可・拒絶を行うパケットフィルタ型と、パケットのペイロードまで監視し許可・拒絶を行うプロキシ型の二種類が主流である。ファイアウォールは通常、組織ネットワークの出入口に設置され、外部からの攻撃を防御する目的のために用いられてきた。内部からの通信を規制するためにも用られるようになり、厳しいセキュリティポリシーを持っている組織の場合は、内部から必要最低限の通信しか許可されない。

ファイアウォールの現状

ファイアウォールはネットワーク上で通信を制限するために重要な役割を果している。近年では IDS と併用して用いられたり、あるいは分散ファイアウォールとして、ネットワーク全体に配置され、異常な通信を通信元から遮断するような利用方法の研究も行われている [4]。しかし、近年可変的なネットワークの普及により、IP アドレスやネットワークアドレスは管理者が割り振るものではなく、DHCP などで自動的に取得される機会が増加している。そのため、IP アドレスやネット

ワークアドレス単位での制限方法を取るファイアウォールでは、内部に接続されたホストからの通信を規制することが困難になってきている。

ファイアウォールの限界

内部からの通信に対しファイアウォールで厳しく規制をかけている組織がある一方で、内部からの通信に規制をかけない組織もある。前者の場合、ネットワーク管理者は組織内部からどのような通信が出ているかに気を配り、同時に業務で必要とされる通信が遮断されていないかについても考える必要がある。厳しい制限をかけたファイアウォールによって、本来業務に使用する通信が遮断されるなどの逆サービス妨害が発生している可能性が考えられるためである。また、後者の場合でも外部に迷惑行為となるような通信が出ていないかについて注意を払わなくてはならない。

また、内部からの通信を規制する際に、これまでは特定ポート、特定通信元、もしくは通信元ネットワーク、通信先などで規制をかけることが多かったが、 DHCP や 無線 LAN による可変的なネットワークが普及しホストが移動することや、別なアドレスを取り直したりすることが多くなった。これにより、サブネット単位での規制は無意味なものとなって来ている。

ルールの変更・削除は管理者が手動でコマンドを与えることにより行われるが、ルールが膨大になるほど人的な管理が難しくなってくる。不要になったルールも削除し忘れることがある。

3.2 関連研究

ウイルス感染ホストなどから行われる分散サービス妨害 (Distributed Denial of Service: 以後 DDoS と呼ぶ) 攻撃に対する自動防御、対策は近年盛んに研究されている分野である。しかし、自動対策のトリガー役である侵入検知システムの検

知精度に限界があることや、度を過ぎた自動対応はそれ自体がサービス妨害になりうるため、実装・運用可能なシステムとして組織ネットワークに積極的に取り入れるまでには至っていない。

本研究を行うにあたり、 DDoS 攻撃防止技術、通信元における DDoS 攻撃防止対策、流量制限技術などについてかかれた論文のサーベイを行った。以下でサーベイを行った論文において提案、実装された DDoS 防御システムの内容について述べる。

3.2.1 Pushback Router

Pushback Router[5] とは、 DDoS 攻撃防止のために、流量制限技術を利用したルータ上で動作するシステムである。カーネルからドロップしているパケットについて調査することで、輻輳を発生させている通信を割り出し、その通信を抑制する命令を次々と上流ルータに伝えていく。IP アドレスを偽装していても、通信先を元に検知するため、実際の通信元まで遡って制限をかけることが可能である。同様の研究に NTT が開発した Moving Firewall [6] が挙げられる。

3.2.2 D-WARD

D-WARD[7] は、設置ネットワークから外部への攻撃を、正当な通信に対するダメージを最小限に抑えつつ抑制することを目的とした Source-end の DDoS 防御システムである。Source-end での検知は検知の確実性が低いことや、過度な自動対応方法は逆にサービス妨害になることもあり Source-end での対応の難しさに繋がっている。そのためこの D-WARD がとる攻撃対策は、ファイアウォールのルールによるフィルタリングではなく帯域制御を利用している。

3.2.3 GrIDS

GrIDS[8] は、active graph というネットワーク接続を示すグラフを用いるワーム検知に重点を置いた IDS である。 active graph を接続ノード間でやりとりし、あらかじめ定義されたパターンをそのグラフから見つけることにより、ワームを検知する。

また、 GrIDS の問題点であるノード間での検知データ交換のオーバーヘッドや、パケットのペイロードが検知情報に含まれないなどを考慮したシステムが設計・実装されている [9]。これはウイルス、ワームの伝播方法に着目した、接続状況を元にした異常検知である。このシステムは Signature-based NIDS のようにシグネチャ (パターンファイル) を最初から提供しない。疑わしさの測定基準のみを用意し、実際の接続履歴から、ワームの動作パターンを抽出し、検知されたパターンが悪意ある通信である可能性を計算する。通常の接続と比較して、ウイルス、ワームは下記のような特徴を持つと述べられている。

接続パターンの類似性

同じ動作が繰り返される。繰り返し同じプロトコルで様々な接続先に接続要求を出し続けるなど。

接続パターンの因果関係

ある動作を受けたホストが他のホストに対し、全く同じ動作を行う。ホストが感染行動を開始する前には、必ず他のホストから同じ感染行動を受けているためである。

無意味な接続

ウイルスやワームは、ランダムに作成したアドレスや、自分と同じアドレス空間に対し順に接続を試行し、攻撃をしかける。それにより存在しないホストや存

在しないサービスに対する接続が急激に増加する。

3.2.4 Self-securing Network Interface

Self-securing Network Interface [10] は、監視対象ホストとは完全に独立した別マシン上で、監視対象ホストの状態を追跡・異常検知を行うシステムである。監視側ホストのカーネルを改良し Network Interface (NI) カーネル Siphon を用意した。Siphon は動作マシンの NIC に届くフレームのコピーを取得し、バッファ後パケットフィルタエンジンに渡す。パケットフィルタルールにマッチしなかった場合、即座にフォワードを行う。監視プログラムが異常を検知すると Siphon に該当接続を制限する命令が送られ、 SYN パケットが抑制される。監視プログラム (スキャナ) は複数用意され、あらかじめ想定されたいくつかの攻撃を監視するようになっており、攻撃の種類によってどのスキャナに通されるかは変化する。

本研究は上記の論文から次のようなアイディアを受けた。

- ウイルスの接続パターンを捉える検知方法の設計
- 帯域制御などを用いた穏やかな規制方法
- 異常な通信の発生元で対策を行うことの有効性

ウイルスの接続パターンが大きなヒントになることを知り、第 4 章でのウイルスを実際に用いた観測実験を行うきっかけとなった。また異常な通信が発生した場合に通信元で対策を行う方が効果的であることが分かった。さらに、通信元で制限を行う場合には、誤検知の可能性も踏まえ完全に遮断するより帯域制御などで緩やかに制限した方が良い事が分かった。

これらのアイディアをもとにより良いシステムを考えるため、本章で取り上げた研究論文に共通する問題点を下記のように考えた。

- 管理者が細かい検知基準の設定や、検知のための様々な重み付け設定を行う必要がある。これらの作業は、ある程度管理者が仕組みを理解することや、運用する能力があることを前提条件としている。検知基準が実際にそぐわない場合、誤検知を多量に発生させてしまう恐れもある。

- カーネルに手を入れたり、各ノード間での情報交換が必要になったりなど規模が大きくなり、導入実現の難易度が高くなる。

こういった問題点をクリアしない限り、管理の工数は増加し、システムは管理者にとって負担を増加させるだけである。第5章の設計では、本システムがいかに簡単さ、明解さを重視したかをポイントとして述べていく。

第 4 章

規制すべき内部ネットワークからの通信

4.1 規制をかけるべき通信

本システムは組織内部からインターネットに対して行われる下記のような通信を対象として規制することを考えている。これらの通信を対応せず放置することはインターネットに対し不要な負荷をかけ、他のユーザに迷惑をかける結果となる。

- ウイルス、ワームによる攻撃パケットの送出
- ウェブページ自動取得ロボットなどによる過度な接続要求パケットの送出
- 設定ミスや誤ったプログラムによる不要なパケットの送出

本章では、上記のうちウイルス、ワームの挙動について詳細に調査を行い、どのような方法で検知・規制が行えるかについてまとめた。

4.2 ウイルスの動作パターン

ウイルスの動作パターンについては、関連研究の章で挙げたように次のようなポイントがある。

- 接続パターンの類似性
- 接続パターンの因果関係

- 無意味な接続

しかし、本研究で着目したのは、もっとも単純で顕著に現れるウイルスの動作である。その特徴は短時間での接続試行回数が非常に多いということである。

ウイルスは大規模拡散を目的としたプログラムであり、拡散のための接続試行を非常に早いスピードで繰り返すよう作成されていることが多い。そのため、帯域の輻輳を発生させるなど、感染以外の二次被害の方が深刻なものとなっている。

研究室内に設置した実験環境で、実際の W32.Nimda.E@mm ウイルス [11] を使用したモニタリングを行った。W32.Nimda.E@mm は 2001 年 9 月 18 日 (米国時間) に最初に発見された W32.Nimda.A@mm のバグを修正したものであり、一部の改良によって、ウイルス検知ソフトの検知を逃れるようになっている。W32.Nimda.A@mm の発見から約 1 ヵ月後に見られるようになった。

この W32.Nimda.E@mm は 10 日おきに発症するメール送信ルーチンが有名であるが、それ以外に無防備な IIS サーバを狙う HTTP(80/tcp) での通信や TFTP(69/udp) での自己ファイル転送など、多彩な感染方法を持ち合わせている。

4.3 ウイルス伝播実験

システム設計の予備実験として、プロトタイプのトラフィックモニタリングプログラムを作成し、二つの環境でトラフィックモニタリング実験を行った。このモニタリングプログラムは C 言語で作成した約 600 行のプログラムである。これは本システムの一部であり、接続試行回数 (SYN フラグ付パケット) についてモニタリングを行い、その結果をログに残していく。

このモニタリングプログラムによって、ウイルスの動作パターンと内部ホスト数千台をかかえる大規模ネットワークでのトラフィックの比較を行うことができ、もっとも単純なウイルス判定の指標を導き出すことができた。

4.3.1 実験環境について

ここでは W32.Nimda.E@mm を利用したモニタリング実験について述べる。実験環境は図 4.1 の通りである。

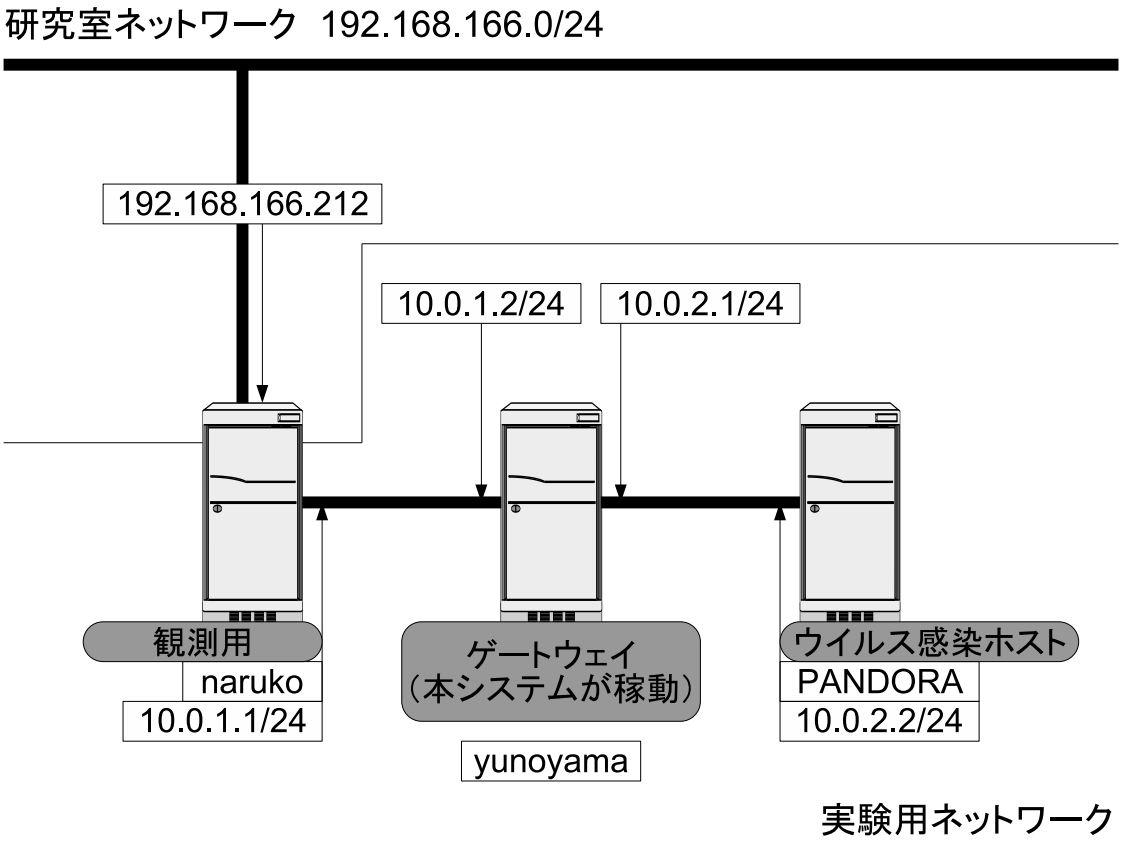

図 4.1: 実験ネットワーク環境

外部への影響を遮断するために、研究室内ネットワークとは切り離した別ネットワークを用意した。観測用ホストのスペックは表 4.1 のとおりである。ゲートウェイマシンのスペックは表 4.2、ウイルス感染ホストのスペックは 4.3 に示した。

観測用ホスト (naruko)

ウイルス攻撃対象ホスト兼研究室ネットワークから実験環境にログインするための踏台サーバである。

表 4.1: 観測用ホスト (naruko) のスペック

OS	FreeBSD 5.2.1-RELEASE
CPU	Intel Pentium III (800MHz)
NIC	Intel 82559 Pro/100 Ethernet
NIC	RealTek 8139 10/100BaseTX

ゲートウェイ (yunoyama)

本システム稼働マシンであり、ファイアウォールによる通信の制限もここで行われる。

表 4.2: ゲートウェイ (yunoyama) のスペック

OS	FreeBSD 4.8-RELEASE
CPU	Pentium II (266MHz)
NIC	Intel 21143 10/100BaseTX
NIC	Intel 21143 10/100BaseTX

ウイルス感染ホスト (PANDORA)

ウイルス感染ホスト役であり VMWare のスナップショット機能を用いてウイルス感染状態からの復帰を容易にした。

表 4.3: ウイルス感染ホスト (PANDORA) のスペック

ホスト OS	Windows XP Professional version 2002
ゲスト OS	Windows XP Professional version 2002 on VMware Workstation 4
CPU	Intel Pentium 4 3.00GHz
NIC	Intel PRO/1000 Network Connection 10/100baseTX

4.3.2 Nimda ワームの感染先探索特性

図 4.1 の実験ネットワークにおいて感染ホストの役割を勤める PANDORA は W32.Nimda.E@mm ワームに感染している。今回 80/tcp(HTTP) の攻撃が広く用いられることに着目して 80/tcp についてのみ観測を行ったが、他に 69/udp の TFTP などの通信や Code Red の残したバックドアを用いた侵入などを試みる。感染方法については図 4.2 にまとめた。

その中で、本システムによる観測に用いたものは 80/tcp 経由での攻撃である。脆弱性のある IIS サーバの探索、攻撃や Code Red II のバックドア利用など、複数の攻撃に使われるプロトコルであり、本システムでの予備実験の際にも、観測された数が一時間で 70000 回以上と、最も多く観察され、本システムでの自動検知、自動対応実験に適していると考えられたためである。

Nimda は攻撃の標的となる IP アドレスを、動作しているホストのアドレスをもとに作成する。この実験環境では PANDORA の IP アドレスは 10.0.2.2 であるため、攻撃の 50% は 10.0.*.* に対し行われる。 25% は 10.*.*.* であり、残り 25% は完全にランダムに生成されたアドレスに対して攻撃を行う。

そのため、組織内部では同じネットワークに属するコンピュータに感染をさせて、同じ攻撃を行うホストを増やしつつ、外部に対し基幹ネットワークの輻輳を引き起こすような過剰なトラフィックを流すこととなる。

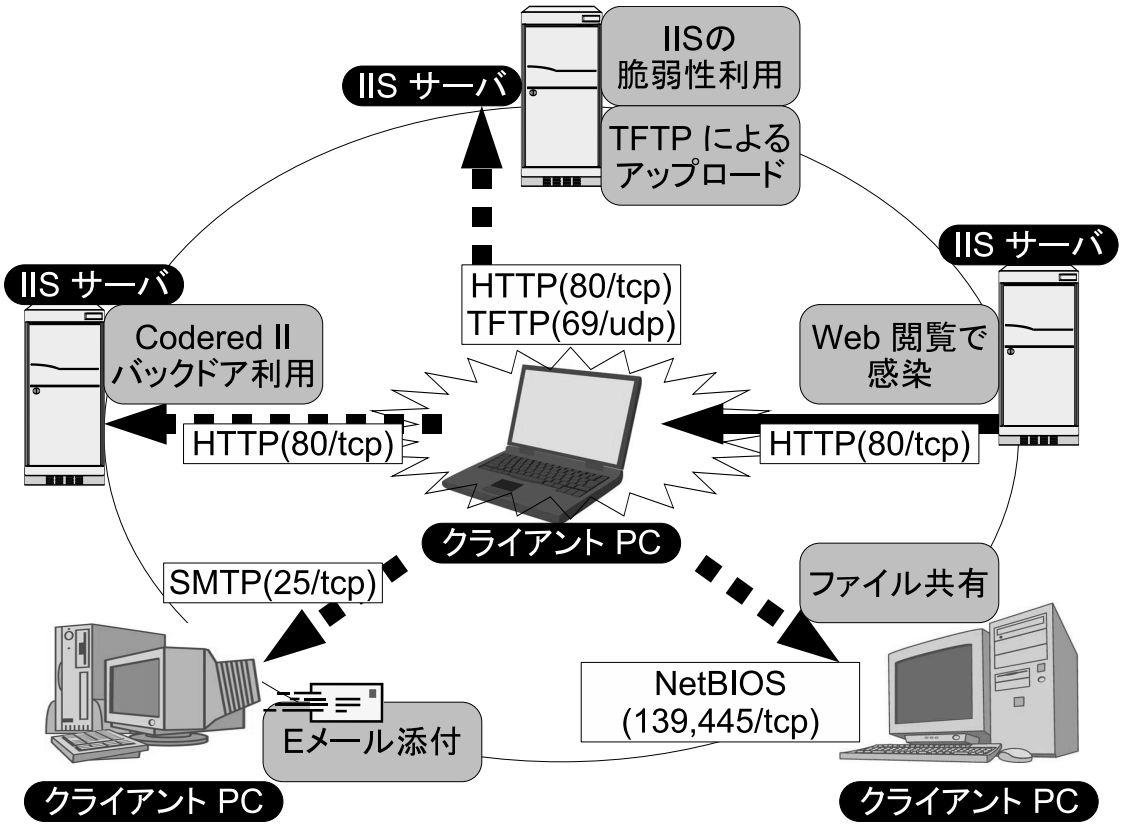

図 4.2: Nimda ワームの感染方法

4.3.3 実在の組織ネットワークにおけるトラフィックモニタリング

研究室内でのウイルス感染実験の比較対象として用いるデータを集めるために、1,000 台を越える内部ホストを抱え、クラス B のアドレス空間を持つ、実在の大規模ネットワークでのモニタリングを行った。

このモニタリングは研究室内部の実験環境と同様に、本システムの元となったモニタリングツールを利用して行われた。対象組織の対外接続部に設置されたタッピングハブに表 4.4 のモニタリングマシンを設置した。このネットワークのトラフィックは非常に多量となることが予想されたため、マシンスペックが十分なものを用意した。

表 4.4: モニタリングマシンのスペック

OS	FreeBSD 5.2.1-RELEASE
CPU	Intel Pentium 4 CPU 3.00GHz
NIC	Intel 21143 10/100BaseTX
NIC	Intel PRO/1000 Network Connection 10/100baseTX

モニタリング実施期間は 2004/07/20 17:32:04 から 2004/08/12 00:42:44 までの約 3 週間程度で、その間に接続が確認されたホストは 1723 台だった。これらのホストは単独でグローバルアドレスを用い繋げられているものもある一方、グローバルアドレスを持った一台のゲートウェイで NAT し、内部にローカルなアドレスを持つホストを多数抱える場合もあると考えられる。

このネットワークでの利用ポートの割合を上位 10 件抜き出し、図 4.3 のようなグラフを作成した。80/tcp での通信が 86.4% を占め、続いて 25/tcp が 9.5% であった。さらに 5 位までの利用ポートと割合は 443/tcp が 1.8 % , 110/tcp が 0.9%, 21/tcp が 0.3% であった。このグラフから、このネットワークの利用傾向として

HTTP によるウェブページの閲覧と、メールの送信・受信、FTP によるファイル取得がメインであることがわかる。 80/tcp についてはウェブページの閲覧以外にも、メッセンジャーソフトを始めとする多くのソフトウェアで用いられているため、このように最も多く見られたと考えられる。あるいは、よく用いられるポートであるからこそ、ファイアウォールで許可されている可能性が高く、それを利用するアプリケーションが増加したとも言える。

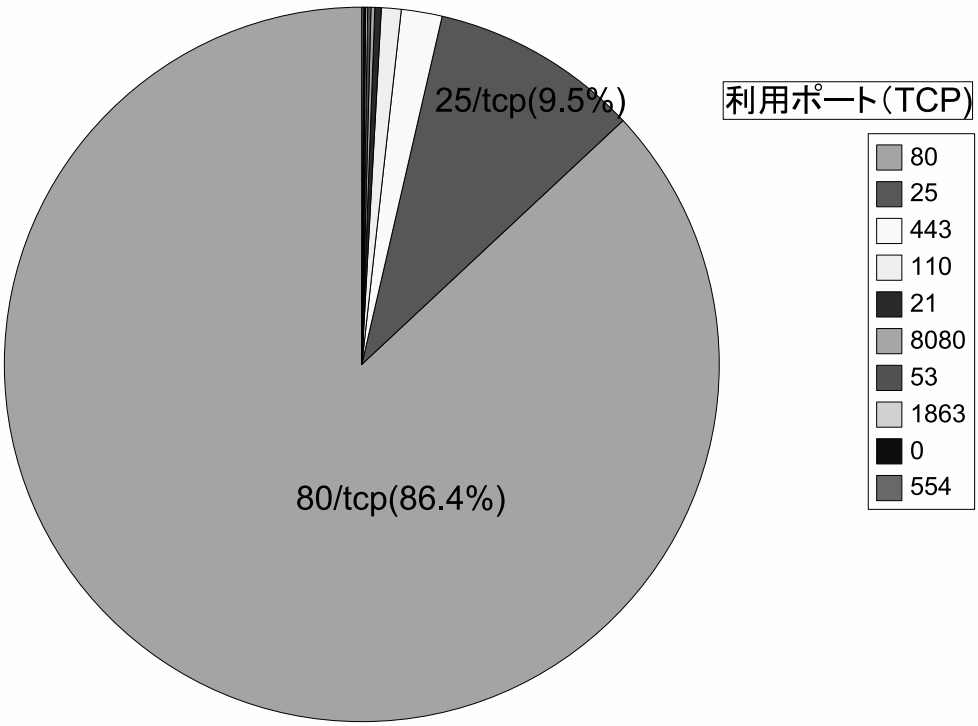

図 4.3: 実在ネットワークにおけるポート別利用割合

この結果から分かるように 80/tcp, 25/tcp のような必須のポートは、正当なソフトウェアだけに限らず、ウイルス、ワームにとっても利用しやすいものである。

次に、このネットワークの内部からインターネットへの HTTP(80/tcp) での接続試行回数を把握するために、最も HTTP(80/tcp) での通信が多かったプロキシサーバに着目した。このプロキシサーバからの通信をグラフ化したものを図4.4に

示す。

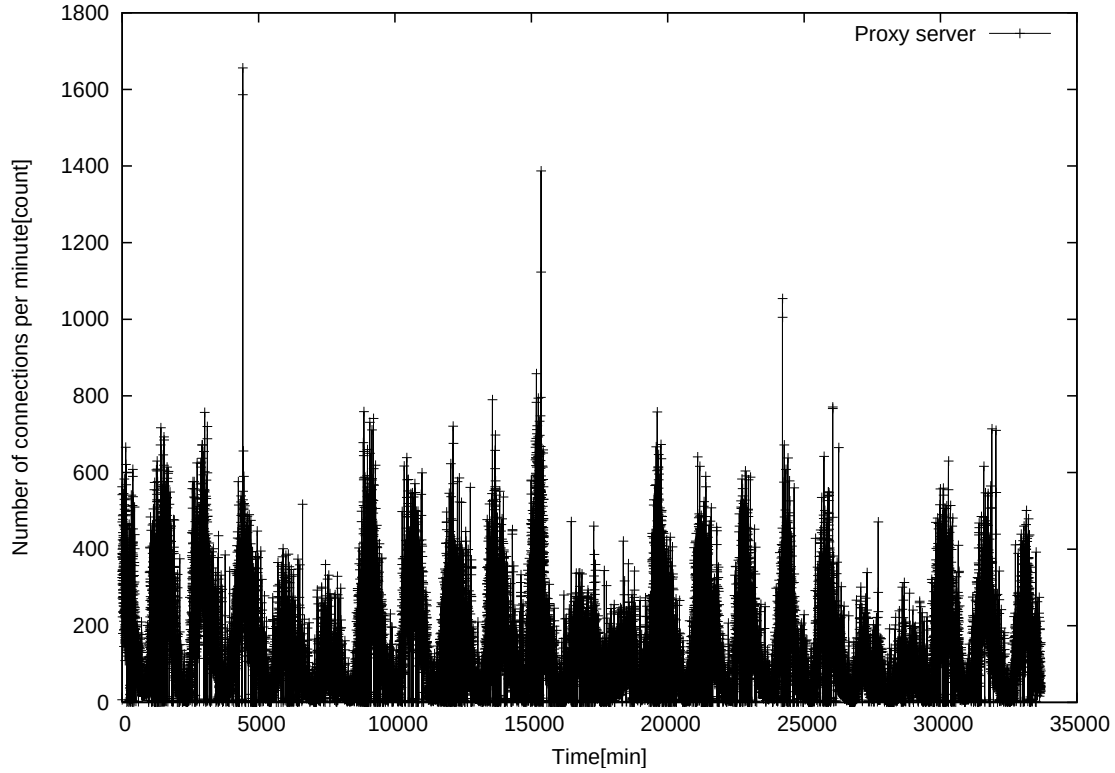

図 4.4: HTTP プロキシの接続試行回数

図 4.4 は 35,000 分 (約 23 日分) のログをグラフ化したものである。ログの取得開始曜日は火曜日からで、 3 週間の利用状況が見られる。ある程度のバースト的な通信の増加が見られるものの、曜日ごと、一日ごとの利用状況は一定している様子がわかる。内部に千台以上のホストを内包するこのネットワークの HTTP プロキシでも、外部への接続試行は一分間に 1000 回を越えることは少ない。ウェブの自動取得ツールなどを使わずに、人間がページの内容を理解しながら、次へのリンクを探し接続していく場合には、頻繁な接続要求は発生しないためであると考えられる。

この結果を踏まえた上で、次に研究室内ネットワークにおけるワーム・ウイルスの接続試行パターンの観測結果との比較を行った。

4.3.4 観測結果

研究室内ネットワークは図 4.1 にて示した通り、ファイアウォールホスト (yunoyama) の内側には、クロスケーブルで直結されたウイルス感染ホスト (PANDORA) が一台存在するのみで、操作をなにも行っていない。しかし、このウイルス感染ホスト (PANDORA) は W32.Nimda.E@mm ウイルスに感染しており、図 4.5 に示すような多量の通信を発生させている。この結果はファイアウォールホスト (yunoyama) の内側 インターフェイスにて観測したものである。

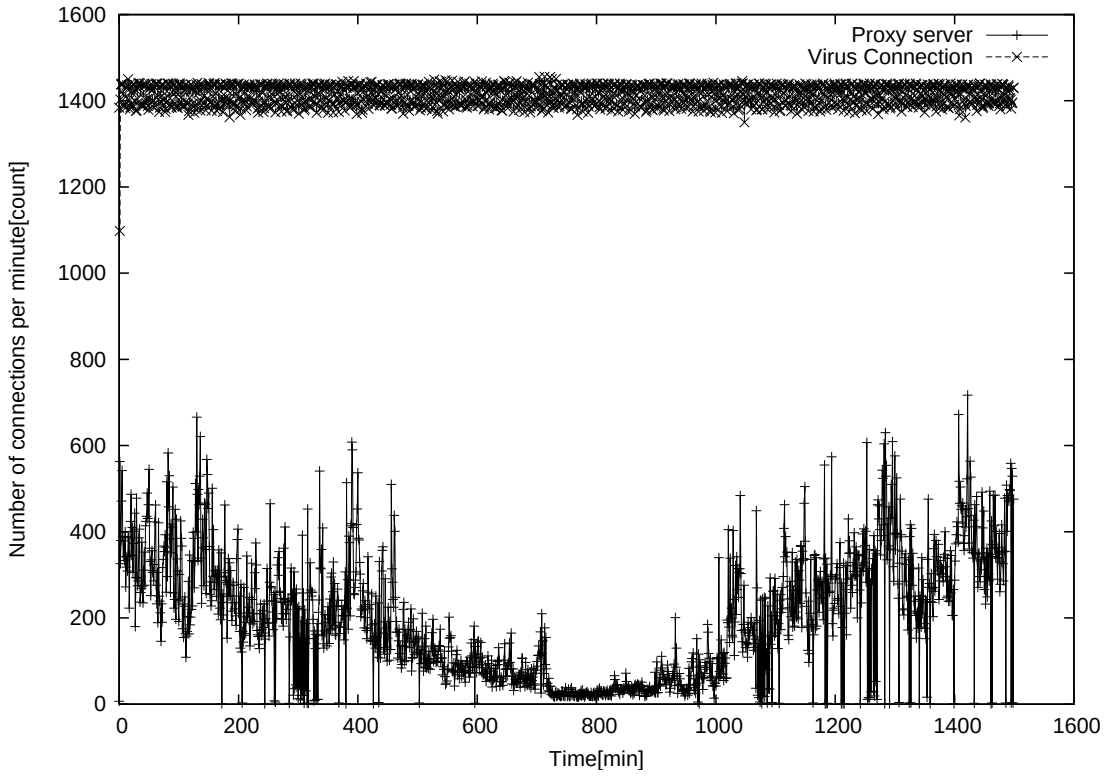

図 4.5: 実在 HTTP プロキシと Nimda の接続試行数比較

実在ネットワークの HTTP プロキシで観測されたものと、研究室内の実験ネットワークで観測された 1500 分のログを抽出し、図 4.5 のような比較表示を行った。実在ネットワークにおいて HTTP プロキシは、複数台のクライアントが利用しており、接続試行も各マシンから出たものである。 HTTP プロキシがキャッシュを

保持していて、それをクライアントに返す事も考えられるが、画像など以外の新しいコンテンツは外部に取得に向かうはずである。それに対し、実験環境ではたった一台のホスト PANDORA からこれだけの接続試行が、モニタリング期間中、常に発生していた。この比較から、 感染ホスト (PANDORA) が明らかに HTTP プロキシを利用している他のホストとは違う振舞いをしているということが分かる。この異常な接続試行の回数を検知に用いることは、非常に簡素だが強力な指標になると考えられる。人がウェブの閲覧を行うだけで、これだけの通信を出すことはまず不可能であると考えられる。

この比較実験からも、ウイルス、ワームは短時間での広域拡散を第一の目的としていることが分かる。もしも、ウイルス、ワームが日常的によく用いられる 80/tcp などのポートを利用した場合でも、通常のホストが送出するよりはるかに大量の SYN パケットを送出している事が分かった。

これらの結果から、このような多量の SYN パケット送出を阻むことで、ネットワーク全体でのウイルスの強力な影響力を封じることができると考えた。

もしも仮にこの通信がウイルスでなく、 Wget のようなウェブページ自動取得ツールだったとしても、再帰的取得でなければ多量の接続要求を出す事はない。もし仮にミラーリングを行っていたとしても、一分間で 300 回程度で、図 4.5 ほどの通信は出さない。これは Wget が一つのコネクションで、相手サーバからの返答を待ち、ファイルのダウンロードが完了するまで、次の接続試行を行わないためであると考えられる。

また、もし Wget を一つのホストで複数立ち上げる、あるいは独自プログラムを使用するなどして、ウイルス、ワームのように図 4.5 ほどの SYN パケットを送出した場合は規制されるべきである。これだけの通信を意図して出すということは、同じ組織ネットワークを利用する他のユーザに影響を与える事を全く考慮しておらず、ある意味 DoS 攻撃であると考えられるからである。インターネットはすべてのユーザの共有資源であり、それを一方的に独占するような方法をとることは、

それ自体が他のユーザに対する迷惑行為である。

4.4 通信抑制実験

IPFW の SYN 抑制ルールによって、ウイルス、ワーム感染ホストからの SYN パケット送出を制限できることを示すため、研究室内に設けた実験環境で自動対応実験を行った。

4.4.1 実験環境

実験環境は図 4.6 に示すように、ファイアウォールホスト (yunoyama) の内側インターフェイス (図 4.6: 観測点 1) と、観測ホスト (naruko) の内側インターフェイス (図 4.6: 観測点 2) の 2 箇所を観測点とした。

また IPFW での制御は、図 4.6 の制御点で示されるようにファイアウォールホスト (yunoyama) の外側インターフェイスで行った。本システムはファイアウォールホスト (yunoyama) 上で稼働し、観測点 1 での閾値を越える接続試行増加をトリガーとして観測ホスト (naruko) 側へ流れるトラフィックを制御する。

4.4.2 接続試行制限実験

実験ネットワークにおいて、 IPFW の SYN 抑制コマンドを用いウイルス感染ホスト (PANDORA) からの接続試行を制限した。外部に出ていく SYN パケットの数量を減少させた様子を実験の際に取得したログを用いて説明する。

感染ホスト (PANDORA) からの 80/tcp での接続試行回数を図 4.6 の観測点 1 において観測した結果、一分間で約 1400 回前後の SYN パケット送出されていることが観測された。

個人差や、閲覧するページの内容にもよるが、ネットショッピングなどの画像読み込みが多いサイトを、内容を読み飛ばしつつ次々と連続して閲覧した場合でも

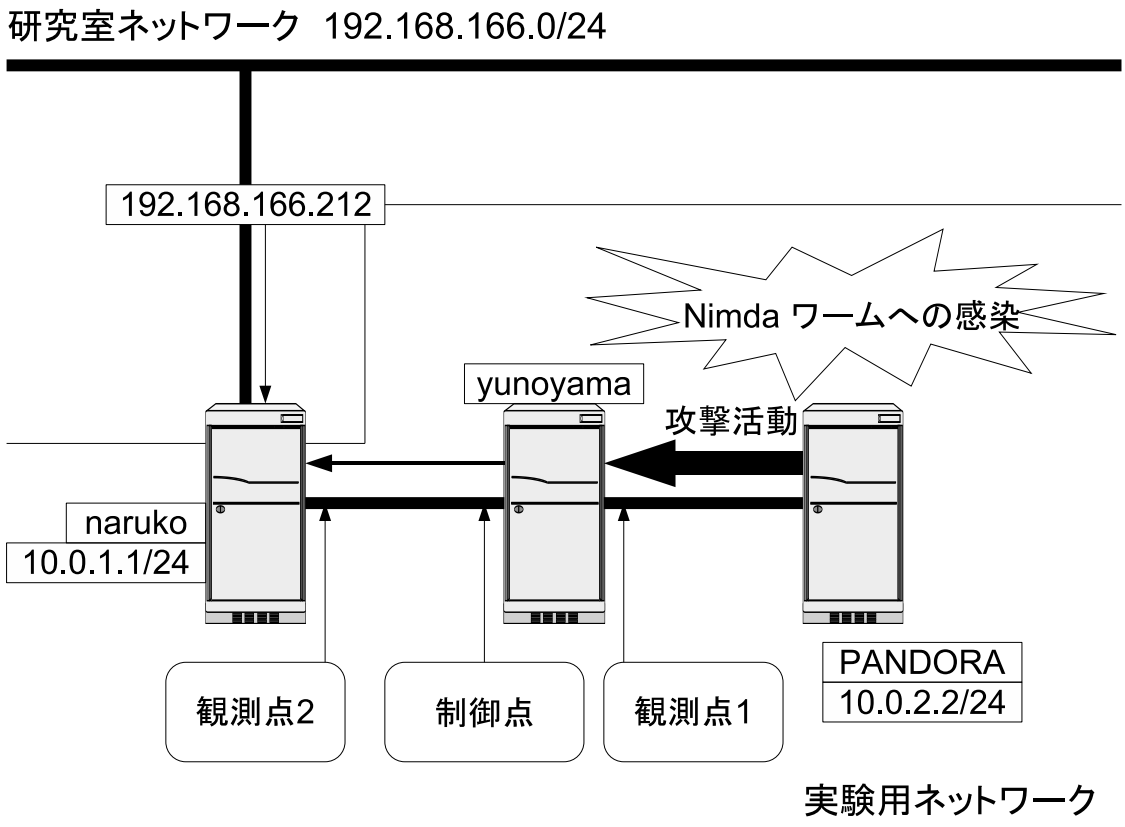
研究室ネットワーク 192.168.166.0/24
192.168.166.212
Nimda ワームへの感染
yunoyama
攻撃活動
naruko
10.0.1.1/24
PANDORA
10.0.2.2/24
観測点2
制御点
観測点1
実験用ネットワーク

図 4.6: 観測実験環境

一分間で 200 回前後の SYN パケット送出しか観測されなかった。wget のような自動ウェブページ取得ツールでも 1 分間に 300 回程度で、 一つのウェブサイトを再帰的に取得するような設定でない限り、その通信が継続することはない。それと比較すると、ウイルス拡散には非常に大量の SYN パケットが出ていることが分かる。

感染ホスト (PANDORA) はこの観測を行っている間、ほぼ 24 時間休止することなく接続試行を行っていた。

制限をかけずに観測点 1 と観測点 2 で取得したログを照らし会わせてみるとプログラム開始時刻に多少のずれがあるため、数に多少の違いはあるものの、表 4.5 のようにほぼ同数の接続試行が見られた。

表 4.5: 制限前の各観測点での観測結果

観測単位 [分]	観測点 1 での接続試行回数 [回]	観測点 2 での接続試行回数 [回]
1	1411	1406
2	1391	1358
3	1369	1378

4.4.3 実験結果

次にファイアウォールホスト（yunoyama）において一度に許可される接続試行回数を 1 に制限するルールを追加した場合、ログに表 4.6 のような変化が見られた。

観測点 1 では 1438 回/分 の 80/tcp での接続試行が制限前と同様記録されているが、観測点 2 では 80/tcp は 3 回しか記録されていない。

現在の設定では、同時に行える接続試行の回数を 1 に限定している。そのため、

表 4.6: 各観測点での観測結果 (制限後)

観測単位 [分]	観測点 1 での接続試行回数 [回]	観測点 2 での接続試行回数 [回]
1	1387	990
2	1425	3
3	1438	3

一つのセッションがタイムアウト待ちをしている間、他のセッションが全く通らなくなり、上記のような結果が観測できたと考えられる。

本章の実験で外部への SYN パケットを制限することで、ウイルス、ワームを始めとする異常な接続試行を抑え、インターネット全体への影響を軽減できることが分かった。

次の第 5 章ではこの方法を用い、簡単な設定で動く自動対応システムの設計について議論する。

第 5 章

設計

5.1 システム設計に向けた問題点の整理

第2章のまとめとして、従来のネットワーク管理には人的・物理的・技術的な制限があり、管理の負担を軽減させるためには、下記の要件を備えたシステムが必要であることを述べた。

- 人的制限への対応
  - 簡単な導入・管理方法
  - ログ処理の自動化
  - ログ統計処理結果にもとづく自動対応
- 物理的制限への対応
  - ログの簡素化によるストレージ節約
  - 必要最低限の動作
- 技術的制限への対応
  - ウイルス、ワームの動作に着目した検知システム
  - 穏やかな通信規制の実施

本章では、上に挙げたような要件にもとづき、それらを満たすシステムの実現方法を議論する。

5.2 人的制限への対応

5.2.1 簡単な導入・管理方法

本システムは組織ネットワークのゲートウェイにて動作することを前提としている。内部の各ゲートウェイにて動作させることも可能であるが最低限の導入コストを考えた場合、組織ネットワークの対外接続ゲートウェイにて動作させるのが最も効果的であると考えられる。

5.2.2 ログ処理の自動化

第 2 章で述べたように、異常発生後に管理者が必ず行う工程として ファイアウォールのログ確認がある。

しかしファイアウォールのログには、多くの通信が記録されておりその中から、問題のある通信を探しだして対応することは管理者の負担であるばかりでなく、対応にかかる時間も増加し、結果として被害を拡大する可能性がある。

第 4 章で行った予備実験の結果から、ウイルス、ワームやそれに準ずる通信には、多くの SYN パケット を発生させるという特徴が得られた。それを基準としてログの処理をプログラムで自動的に行い、多量の SYN パケット を送出しようとするホストを探し出すことが可能である。

5.2.3 ログ統計処理結果にもとづく自動対応

第 2 章で述べたように、ファイアウォールのログ確認を行った後で、該当する異常な通信をインターネットに送出するのを規制する必要がある。その際には必ず

ファイアウォールの設定変更が行われる。ログの処理を自動で行い、異常な通信を発生させている該当ホストが発見された場合、そのホストからの通信を制限するための自動対応も共に行うことで、さらに管理者の対応工数を減らすことができる。さらに夜間などの管理者不在時に、代理で一時的な対応を行うことができる。これにより管理者が対応できるまでの間、ウイルス、ワームなどのパケット送出を抑えることが可能である。

5.3 物理的制限への対応

5.3.1 ログの簡素化によるストレージ節約

本システムが用いるログは、自動対応に必要な最低限の情報のみを必要とし、すべての通信について取得する必要はない。具体的には TCP のパケットで SYN フラグの付いているものだけを対象にすれば良い。さらに、本システムが接続数の増加を把握するためには、時系列に観測されたパケットを記録するよりも、接続単位時間毎に区切ってパケット数をカウントした方が、増加傾向を把握しやすい。そのため、本システムではパケットのペイロードについての記録は行わず、単位時間ごと、ホストごとに TCP の SYN フラグ付きパケットを記録することとした。これにより、パケットのヘッダからペイロードまで全てを時系列に記録していくよりも、ログ保存のために用いられるストレージを節約することができる。

5.3.2 必要最低限の動作

本システムが導入・動作するゲートウェイホストは十分なマシンパワーがあるものと考えられる。しかし他にも IDS や tcpdump などのモニタリングツールなどが動く可能性もある。そのため本システムはそれらとともに動作しても、他の動作に影響を与えない、なるべくリソースを節約した、簡単な造りを目指す。そのた

めに、第4章で行った予備実験の結果を元に、本システムは異常な通信を検知して対応するまでに必要最低限の動作のみを行うものとする。

5.4 技術的制限への対応

5.4.1 ウイルス、ワームの動作に着目した検知システム

第4章で行った予備実験で、ウイルス、ワームやその他の制限すべき通信に共通の特徴を得ることができた。これまでの Signature-Based IDS のように、シグネチャ（ルールパターン）とパケットとの照合を行う場合、シグネチャに存在しないウイルス、ワームは検知できないという問題点があった。しかし本システムはウイルス、ワームあるいはそれに準ずる通信が、ある一定時間において通常の通信をはるかに上回ることを対応の根拠としている。そのため未知のウイルス、ワームであったとしても、高速な大規模拡散を試みている限り、本システムで検知可能である。

5.4.2 穏やかな通信規制の実施

本システムはウイルス、ワームの動作に着目した検知システムではあるが、分類的には Anomaly-Based NIDS に近い。そのため万が一、誤検知でウイルスではなかった場合の通信を完全に遮断してしまった場合、それが逆の DoS となる可能性がある。そのため、SYN パケット に対して緩やかな通信規制を行うことで他の通信への影響を最小限に抑えながらも、その許可される範囲内で通信を続けることを許可することができる。

5.5 システム概要

図 5.1 にシステム概念図を示す。

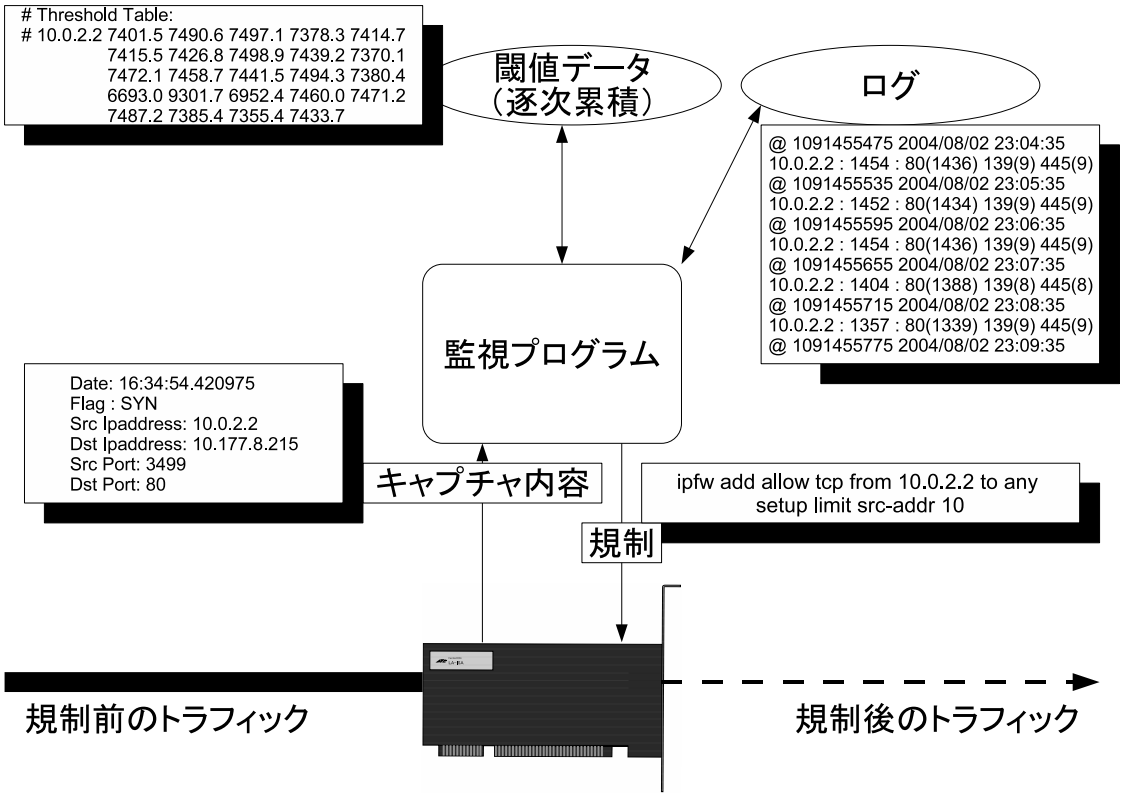
Threshold Table:
10.0.2.2 7401.5 7490.6 7497.1 7378.3 7414.7
7415.5 7426.8 7498.9 7439.2 7370.1
7472.1 7458.7 7441.5 7494.3 7380.4
6693.0 9301.7 6952.4 7460.0 7471.2
7487.2 7385.4 7355.4 7433.7
閾値データ
（逐次累積）
ログ
@ 1091455475 2004/08/02 23:04:35
10.0.2.2 : 1454 : 80(1436) 139(9) 445(9)
@ 1091455535 2004/08/02 23:05:35
10.0.2.2 : 1452 : 80(1434) 139(9) 445(9)
@ 1091455595 2004/08/02 23:06:35
10.0.2.2 : 1454 : 80(1436) 139(9) 445(9)
@ 1091455655 2004/08/02 23:07:35
10.0.2.2 : 1404 : 80(1388) 139(8) 445(8)
@ 1091455715 2004/08/02 23:08:35
10.0.2.2 : 1357 : 80(1339) 139(9) 445(9)
@ 1091455775 2004/08/02 23:09:35
監視プログラム
Date: 16:34:54.420975
Flag : SYN
Src Ipaddress: 10.0.2.2
Dst Ipaddress: 10.177.8.215
Src Port: 3499
Dst Port: 80
キャプチャ内容
ipfw add allow tcp from 10.0.2.2 to any
setup limit src-addr 10
規制
規制前のトラフィック
規制後のトラフィック

図 5.1: システム概念図

5.5.1 監視プログラム

監視プログラムは、 pcap ライブラリ [12] を用い、ネットワークインターフェイスから入力されるパケットをキャプチャしている。本システムにおけるウイルス検知は、接続試行フラグ (SYN フラグ) のついているパケットのカウントで行う。タイムスタンプ、接続元 IP アドレスや接続先ポートなどの情報を設定されている単位時間ごとに集計していき、ログファイルに出力する。また、そのログファイルを一定間隔で読み取り、内部に閾値データを保持して、逐次累積更新していく。この読み取り間隔は変更可能な値である。

5.5.2 ログ

@ マークで始まる部分がタイムスタンプである。図 5.1 の例では一分間隔で出力している。@ の次の行が通信内容の記録である。左から順に、通信元 IP アドレス、 全体の接続試行回数、利用されたポート (通信回数) となっている。監視中に出現した新しいホストの追加や、カウントしているホストの通信回数の更新のため、このログファイルに対して頻繁にアクセスが発生する。本システムはすべてのログを残さないものの、各ホストの通信動向を逐次更新して記録していく。そのため、ホスト数の多いネットワークに接続される場合、ログのリストからホストを探索する際に、順に n 個の要素からなるリストを探索する方法では $O(n)$ の計算量がかかってしまう。そのため、本システムでは二分木を用いてこのログのリストを作成し、リスト探索の際にかかる計算量を $O(log_2 n)$ に抑えた。

5.5.3 閾値データ

閾値データは常にプログラム内部で保持し、シグナルで停止された時に# で開始するコメント行として、ログファイルに書き出される。システムが再度立ち上がる時には、ログを再度読んで計算しなおすため、# で始まるコメント行は、あく

まで管理者が把握するために出力されるだけである。

本システムを利用する組織によって、現在の通信回数を重視する場合と、過去の通信回数を重視する場合が異なることを考慮し、閾値算出には、それぞれ現在の値と過去の値に重みをつけられるようにした。過去閾値と現在の閾値にそれぞれ重み付けを行っても、最終的に得られる値は一定でなくてはならない。そのため過去の重み付け値と現在の重み付け値を足し合わせた時にちょうど 1.0 になるようにした。

接続試行回数の閾値算出は下記の計算式で行われる。

$$threshold = oldthreshold * pastfactor + currentcount * (1.0 - pastfactor)$$

各要素についての説明は下記の通りである。

oldthreshold

過去の閾値

pastfactor

過去の閾値の値をどのぐらい考慮するかの重みづけ

currentcount

新たに集計された一定時間の通信数

pastfactor は 0.0 から 1.0 の間で可変であるが、デフォルト値は 0.6 と設定している。 0.0 の場合過去の閾値を全く考慮に入れないことになる。また、 1.0 の場合は閾値を更新せず、現在の値は無視されることになる。

5.5.4 通信規制

実際の通信規制は FreeBSD などに最初から導入されている IPFW(ただし利用にはカーネルモジュールとカーネルの再構築が必要) を用いて行う。本システムで

は接続試行回数を対象としており、ウイルスのような短時間で大量の接続試行を繰り返す通信に対し、 一度に送出できる SYN パケットの数を限定する方法で対策を行う。

5.5.5 IPFW によるウイルス、ワーム伝播の抑制効果

前セクションで述べたような異常な通信の抑制手段として FreeBSD に最初から搭載されている IPFW の SYN 抑制ルールを利用することを考えた。

IPFW の SYN 抑制ルールは、 FreeBSD の マニュアルページ IPFW(8) でも確認することができる。このルールは特定のソースあるいはネットワークから一度に発信できる SYN パケットの数を制限するもので、下記のような記述で実行される。

```
#ipfw add allow tcp from srcIP to any setup limit src-addr 10
```

図 5.2: IPFW の SYN 抑制ルール追加コマンド

この例では *srcIP* から外部ホストすべてに対して一度に発信できる SYN パケットの上限値を 10 としている。この srcIP はネットワークアドレスでも指定可能であるが、本システムは個別のホストに対し抑制を行うため、ここではネットワークアドレス指定を用いない。

本システムではこのコマンドをウイルス、ワーム拡散防止のための手段として、自動対応の際に使用する。

第 6 章

システムの実装

6.1 システム実装の概要

本システムは約 1050 行の C のプログラムである。第 4 章でのモニタリング実験に使用したモニタリングプログラムをもとに作成した。元のモニタリングプログラムに、一分間のログと閾値との比較機能を追加し、閾値を越えていた場合には IPFW のルールを発行する機能を追加した。

6.2 実装方法の選定

本システムの設計にあたり、どのような方法で実装を行うかの検討を行った。候補としては次のような方法を考えた。

1. 既存モニタリングプログラムに手を加え、必要な出力を得る方法
2. 既存モニタリングプログラムの出力を ruby で処理する方法
3. Pcap ライブラリを使用した必要最低限のキャプチャプログラムに、必要な出力を得る部分を追加する方法

1 の方法では、既存モニタリングプログラムの理解が必要である。また、既存モニタリングプログラムはさまざまなプラットフォームで動くように、 OS 別のコー

ドが混在している。そのため理解しにくい部分もあり、限られた時間で実装を行っていくことは難しいと考えた。

2 の方法では、実際に試作してみたが、既存モニタリングプログラムの CPU 使用量が多いことや、さらにその上で ruby の処理をすることで、リアルタイムの結果を得る事が難しいと考えられた。また ruby で文字列処理を行う際に、既存モニタリングプログラムからの出力で必要な部分はごくわずかなため、すべての出力を受ける事は無駄が多いと考えた。

そのため本システムでは 3 の、Pcap プログラムをもとに、本システムで必要な SYN パケットや接続元アドレスなどを抽出する部分を、適宜追加する方法を取った。

6.3 システム詳細

本システムは下記の 5 つのパートに分割できる。

- 接続試行回数のロギング
- ホスト別時間別の接続試行回数の平均値算出
- 累積ログの読み込み
- 異常検知
- 検知結果に基づく自動対応

それぞれの動作、役割について次で詳細に述べる。

6.3.1 接続試行回数のロギング

まず pcap ライブラリを用いてパケットをキャプチャしていく。その中で TCP のパケットでかつ SYN フラグのついた、接続試行をしているパケットについてロ

ギングを行う。必要な情報は、接続元の IP アドレス、使用プロトコル、タイムスタンプのみである。一定時間に区切って該当するタイムスタンプ分のパケット情報を読みだしていき、そこから集計した結果を一定時間ごとにログファイルに出力する。ログファイルはプログラムが読んでも、人間が読んでも分かりやすい図 6.1 のようなテキストファイル形式とした。

```
@ 1092150829 2004/08/11 00:13:49 10.0.2.2 : 144 * 1406 : 80(1388) 139(9) 445(9)
@ 1092151069 2004/08/11 00:17:49 10.0.2.2 : 144 * 1455 : 80(1437) 139(9) 445(9)
```

図 6.1: ログファイルの一部抜粋

図 6.1 の @ はタイムスタンプ部分である。最初に UNIX 時間 (1970 年 1 月 1 日 00:00:00 UTC からの経過時間を秒単位で表現) を記録し、その後に通常の日付と時刻が記録される。タイムスタンプの次の部分からがその時間帯に記録された通信である。

この図 6.1 では 10.0.2.2 が通信を行っていたホストの IP アドレスである。次の 144 という値は制限値である。これは記録されたログから算出した閾値に、検知のための重み（デフォルト値は 1.2) をかけたものである。

* の次に来ている 1438 が実際の接続試行回数であり、『:』(セミコロン）以降の 80(1420) 139(9) 445(9) という値はそれぞれ、接続試行に利用されたポート番号と、その内訳数となっている。10 ポート以上が利用されていた場合には、発見した順に表示していき、10 以降を 0 として分類するようになる。

6.3.2 累積ログの読み込み

閾値の算出には、出力したログファイルを用いる。プログラム起動時に、該当するログファイルがあれば読み込み、それまでの値を計算して、その続きから新たに集計結果を得て、閾値を更新する。接続試行回数の閾値算出は設計の章でも述べたとおり、下記の計算式で行われる。

$$threshold = oldthreshold * pastfactor + currentcount * (1.0 - pastfactor)$$

プログラムが停止した際は、その計算結果をログファイルに出力する。

ただし、これはコメント記号 『#』で始まり、プログラムの動作自体には関係しない。

他にも図 6.2 のようにプログラムの開始・終了や IPFW のルール追加コマンドを発行した際の標準出力、標準エラー出力などがログファイルにコメント記号付きで出力される。

```
# Started at:
# Threshold Table:
# 10.0.2.2 29 : 4320.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 602.4 0.0 0.0 0.0
# 14825> 10000 allow tcp from 10.0.2.2 to any limit src-addr 1 setup
# Stopped at:
```

図 6.2: ログ中のコメント記号記述

また、図 6.2 以外の出力として 図 6.3 に示すような『!』で始まるログがある。これはファイアウォールのルールに変化があった事を意味している。このルール変更のログはカウントされ、プログラム停止時に出力されるログに回数を表示するのに用いられる。(図 6.3 最下行)

6.3.3 異常検知

異常検知は、これまでの累積ログから作成した閾値データに重み付けした数値を上回った際に行われる。その値は *limitfactor* という変数で設定可能である。今回の実験環境においては、ごく些細な通信回数の変化でも対応できるようにデフォ

```
! limit 10.0.2.2 10000
! unlimit 10.0.2.2 10000
# 10.0.2.2 27 : 12000.0 1920.0 .... 1920.0 1920.0 1920.4
```

図 6.3: 『!』 で始まるログ

ルト値を 1.2 としている。しかしウイルスの場合は、通常の利用傾向をはるかに上回る接続試行が発生するため、より高い値に設定しておいた方が、誤検知の可能性は減少する。

異常検知された時の接続試行回数を、閾値データの計算に含めると閾値データが高くなり、誤検知を引き起こす恐れがあるため、異常検知された場合には、*limitfactor* をかけた、接続試行回数以上のカウントは含めない。図 6.4 に示すようにその値が上限値に設定されることとなる。

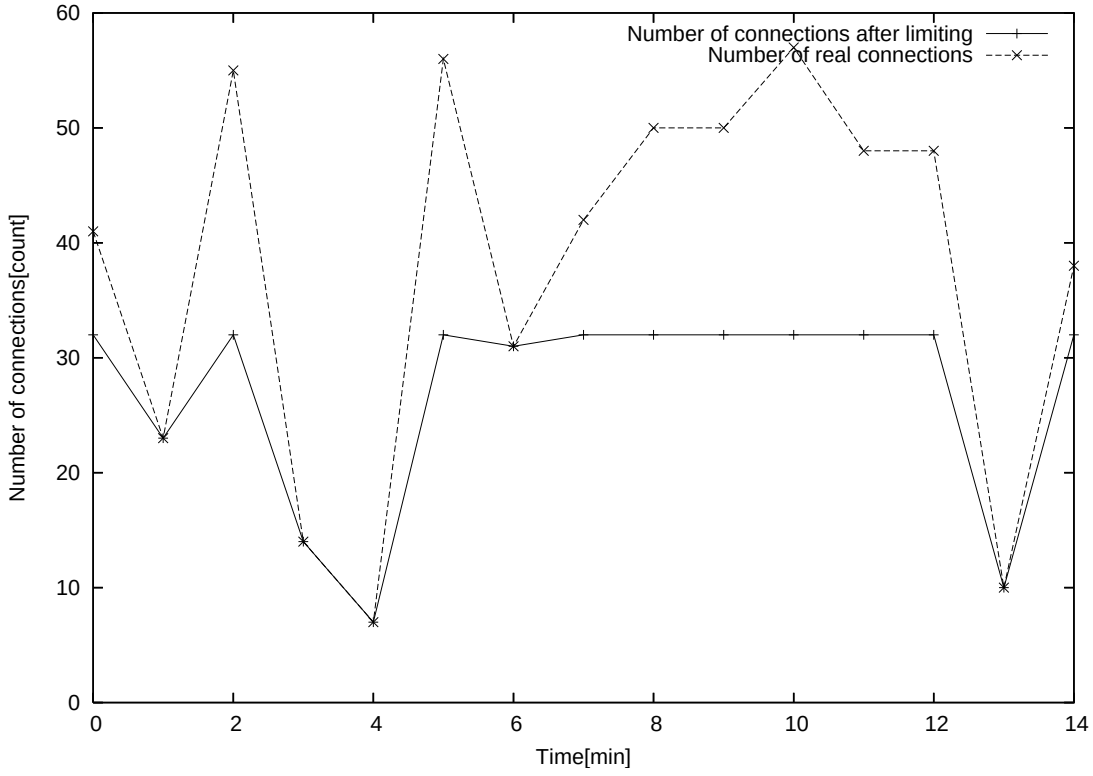

図 6.4: 実際の接続試行回数と閾値以下に制限されたログの比較

6.3.4 検知結果に基づく自動対応

本システムが以上を検知し自動対応を行う際には、 FreeBSD に最初から導入されているパケットフィルタ型ファイアウォールである IPFW を使用する。

ファイアウォールには、通信の利用ポートなどから判別してルールにマッチングするパケットフィルタ型と、通信のペイロードまで判別してルールにマッチングさせるプロキシ型がある。プロキシ型の場合、通信の詳細を判別するため、マシンパワーが必要となり、そのファイアウォールがネットワークのボトルネックになる可能性がある。

本システムでは、通信のペイロードまでを対象としていないため、軽くて扱いが簡単なパケットフィルタ型の IPFW が最適であると考え、これを本システムの一部として利用した。

本システムは異常を検知した際に、 IPFW で接続試行を制限するコマンドを発行する。

6.4 動作環境について

現在の実装ではトラフィック制御を行うために IPFW の制御ルールを用いている。したがって開発は FreeBSD のみを対象にしている。しかし他の環境に移植することは容易である。

本システムのプログラムは TCP の SYN パケットのみを対象としてモニタリング、集計し、他のパケットキャプチャツールのようにすべてのパケットは検査しない。そのため高いマシンパワーを必要としない。実験環境の yunoyama のような古いマシンでも、リソースを消耗することなく正常に稼働できる。ただしトラフィックの量によっては、より良い NIC やそのドライバを必要とする事も考えられる。

6.5 pcap ライブラリ

本システムのパケットキャプチャ部分の実装には pcap ライブラリ [12] を用いた。

pcap ライブラリは Van jacobson 氏らの開発したライブラリで、パケットのキャプチャに用いられる。 Unix を始めとする様々なプラットフォームで動作する。また Win32 システム用のライブラリである Winpcap も存在する。

この pcap ライブラリを利用する事で、簡単にパケットキャプチャが行えるだけでなく、広いプラットフォームで利用可能なため、本システムの移植性を高めることができる。

6.6 本システムの起動

6.6.1 起動コマンドと利用権限

本システムは FreeBSD 上で起動する。起動のためには図 6.5 のようなコマンドを入力する。tcpdump などのパケットキャプチャプログラムと同様、自分宛のパケットだけでなくネットワーク上を流れるパケットを無差別に取得する promiscuous モードで動かす必要がある。promiscuous モードで動作させるためには root 権限が必要とされる。

```
prototype [options] <network interface> <logfilename>
```

図 6.5: 本システムの起動コマンド

コマンドライン行末に & を付け足して、バックグラウンドのプロセスとして動かすことも可能である。本システム停止時にはコマンドライン実行の場合 Ctrl + C で終了する。バックグラウンドのプロセスとして動かしている場合には該当プロセスにシグナルを送り終了させる。

指定するログファイルは起動時にロックされる。そのため本システムを一つ起動すると、そのプログラムが終了するまで、同じログファイルを参照した本システムを起動する事はできない。

6.6.2 本システムのコマンドライン引数

本システムのコマンドライン引数は下記のようになる。

```
-s              only dump the threashold table; ignore network interface
-p factor       past factor
-l              enable the limit control
-m factor       limit factor
-i n            watch interval (second)
-a n            statistics interval (second)
-t n            limit time (second)
-S addr/mask    source address/mask; mask is optional
-x n            minimum of ipfw's rule numbers
-y n            maximum of ipfw's rule numbers
-r 'addr threshold'
```

各オプションについて説明する。

-s

これは閾値の計算のために、ログの読み込みだけを行うモードで動作させる場合に用いる。ネットワークインターフェイスから入って来る新たなパケットは無視される。

-p

閾値の計算を行う際に、過去の値にかけられる重みを手動で設定したい場合に用いる。デフォルト値の 0.6 であれば指定する必要はない。

-l

IPFW のルールを用いた制限を用いる場合にのみ用いる。観測を行うだけで良い場合には必要ない。

-m factor

異常と検知する際に閾値にかける重み。デフォルトでは 1.2 であるが 1.0 以上の値に変更可能。

-i n

ログ取得の間隔を秒単位で指定できる。デフォルト値は 60 秒。

-a n

ログ読み込みの間隔を秒単位で指定できる。デフォルト値は 3600 秒 (一時間)。

-t n

制限解除までの時間を秒単位で指定できる。デフォルト値は 3600 秒 (一時間)。

-S addr/mask

接続元アドレスとして認識するアドレスを指定。外部ネットワークからのパケットをカウントに含めないために、設定する。マスク指定で接続元ネットワーク指定も可能。-S 192.168.0.1/255.255.255.0 のように指定する。

-x n

利用できる IPFW ルール番号の最小値。 IPFW のルールは下記のように、各ルールに番号が振られており、ルールの削除の際には番号を指定する。

```
00100   6790   9798578 allow ip from any to any via lo0
00200      0         0 deny ip from any to 127.0.0.0/8
00300      0         0 deny ip from 127.0.0.0/8 to any
```

```
65000 584292 268910416 allow ip from any to any
65535      0          0 deny ip from any to any
```

既に用いられている既存ルールと番号がぶつからないように、本システムで用いる事のできるルール番号の範囲を指定する。デフォルトでは 10000 - 19999 の間を用いる。

-y n

利用できる IPFW ルール番号の最大値

-r 'addr threshold'

閾値をログの統計から用いるのではなく、管理者が手動で設定指定したい場合に用いる。 -r 'addr @ threshold' とすると、すべての単位時間に対して同じ閾値を設定できる。

6.7 ログ処理

ログファイルはコマンドラインでファイル名を指定する。本システムのプログラムがコマンドライン引数で指定したログファイルにアクセスするのは次の 3 つの場合である。

リスト探索

まず、パケットを取得するたびに、このログファイル中にあるホスト情報のリストを探索する必要がある。通信していることが確認されたホスト数が多ければ多い程、このリストは長大なものとなるため、探索には二分木を用いた。

ホスト登録

リスト探索を行い、そこにない新しいホストであれば、リストに追加を行う。また、リストにあった場合は、接続試行が確認された分だけカウントを増やしていく。

ログ読み込み

起動後一旦停止し、再開する場合、その時点までの閾値の計算を再度行うためにログの読み込みが発生する。

6.8 自動対応

本システムは異常を検知した際に IPFW の接続試行回数制限を用いて、該当する通信の外部への接続試行を制限する自動対応を行う。

6.8.1 発行コマンド例

実際に用いられるコマンドは図 6.6 の通りである。

```
ipfw add allow tcp from srcIP to any setup limit src-addr Num
```

図 6.6: 本システムが発行する IPFW 制御コマンド

これは *IP address* からすべてのアドレスについての TCP 接続で一度にオープンできるコネクションの数を *Num* に限定するというものである。この *IP address* には本システムが検知した、多量の接続試行を行っているホストの IP アドレスが入る。*Num* には同時接続の上限値を入れる。

第 7 章

評価と考察

7.1 本システムの検証実験

本システムが通信の増加を検知し、自動的に通信の抑制を行えることを示すため、研究室内に設けた実験環境で自動対応実験を行った。

7.1.1 実験環境

実験環境は図 7.1 に示すように、ファイアウォールホスト (yunoyama) の内側インターフェイス (図 7.1: 観測点 1) と、観測ホスト (naruko) の内側インターフェイス (図 7.1: 観測点 2) の 2 箇所を観測点とした。

また IPFW での制御は、図 7.1 の制御点で示されるようにファイアウォールホスト (yunoyama) の外側インターフェイスで行った。本システムはファイアウォールホスト (yunoyama) 上で稼働し、観測点 1 での閾値を越える接続試行増加をトリガーとして観測ホスト (naruko) 側へ流れるトラフィックを制御する。

7.1.2 接続試行の制限

IPFW で制限を行うゲートウェイホストである yunoyama 上で本システムを接続試行制限機能つきで動作させた。動作開始のコマンドには、第 6 章で説明したコマンドライン引数のうち、下記のものを指定した。

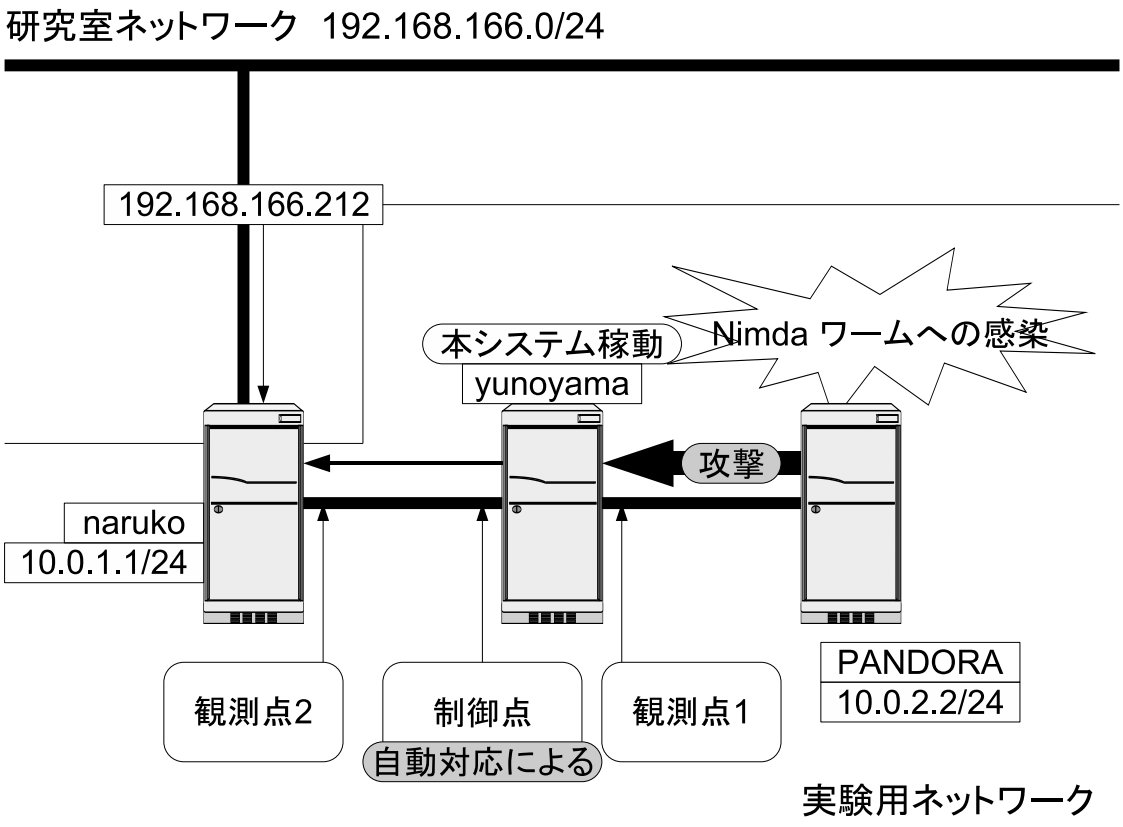
研究室ネットワーク 192.168.166.0/24
192.168.166.212
本システム稼動
Nimda ワームへの感染
yunoyama
攻撃
naruko
10.0.1.1/24
PANDORA
10.0.2.2/24
観測点2
制御点
観測点1
自動対応による
実験用ネットワーク

図 7.1: 観測実験環境

1. -S 10.0.2.2

2. -r '10.0.2.2 @ 12000.0'

観測対象とするパケットをウイルス感染ホスト (PANDORA：10.0.2.2) に限定し、さらにその閾値を手動で 12000 回に設定した。これは画像の多いウェブサイト閲覧を行った際に発生する約 200 回/分の接続試行回数をもとに、その値に単位時間を一時間として 60 をかけたものである。

ウイルス感染ホスト (PANDORA：10.0.2.2) からは、ほぼ 1400 回/分 の 80/tcp での接続試行通信が行われ続けている。この実験結果は図 7.2 のようになった。

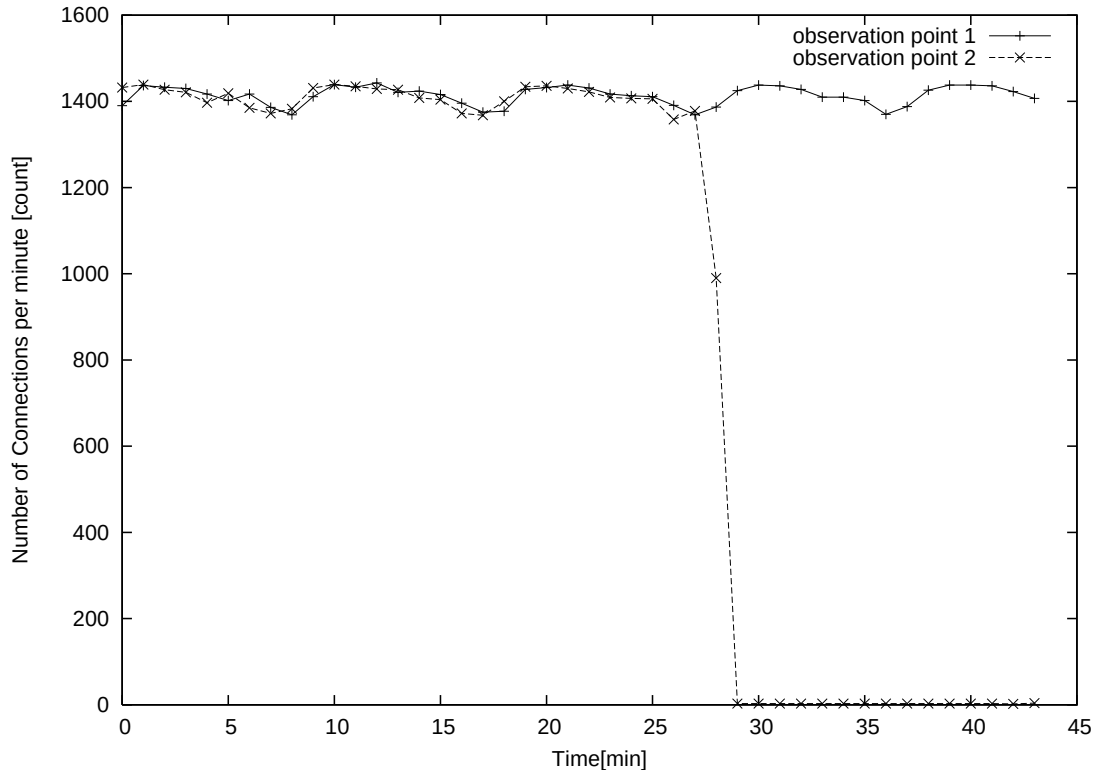

図 7.2: IPFW による接続試行制限の有無と接続試行回数の比較

IPFW ルールが適用された後の通信が到着する naruko の内側では接続試行制限がなされていない yunoyama 内側に比べ図 7.2 に見られるように、大幅な抑制が行われていることが分かる。これを行うことにより、外部に対してウイルスが無制

限に流出することを抑制する事が可能である。

7.1.3 実験結果についての考察

観測が開始してから 27 分ごろに本システムを稼働させた。図 7.2 より、システム稼働から約 1 分後に接続の制限が行われていることが分かる。検知から対応まで約一分弱かかっているため、ウイルスの稼働開始が、本システムの稼働開始と同じ場合、約 1 分間の通信がそのまま外部に出ていくこととなる。

しかし、実際人間の管理者が検知システムの連絡をメールなどで受けてからログ調査を開始し、さらにファイアウォールルールで適切に制限するまでには、1 分以上かかる。さらに夜間で管理者が連絡を受けられない場合や管理者が不在の時には、対処が行われるまで更に多くの時間を必要とする。

また Signature-based IDS を利用して、自動対処するスクリプトも存在する。しかし Signature-based IDS の検知に基づくため、そのパターンファイルにマッチしない新たなワームやウイルスの攻撃には対処が行えない。また、禁止のルールを追加するタイプのスクリプトであるため、検知が誤検知だった場合には、正常な通信を突然禁止してしまうことになる。

本システムは、それらの複合的な対処方法に比べ、ウイルスの行動パターンに特化し、拡散スピードを数十分の一に抑えることで、ウイルスやワームの一番の目的である短時間での大規模拡散を阻止することが可能である。

完全な抑止を一部のネットワークで行うより、すべてのネットワークで本システムのように緩やかな検知と制御を行う方が、ネットワーク全体を埋め尽くすようなウイルス蔓延、輻輳発生による二次被害を防ぐことが可能であると考えられる。

7.2 本システムの対応範囲

ここでは、本システムが対応できるウイルス、ワームあるいはそれに準ずる通信について述べていく。

7.2.1 本システムが対応可能な通信

本システムは SYN パケットの数をカウントし、日常的に行われている通信で見られる SYN パケットの数量を越えるような通信を制限対象としている。このような実装で対応可能な通信は、 TCP で多量の接続試行を試みるものである。具体的には下記のような例が挙げられる。

マスメール送信型ウイルス

マスメール (大量メール) 送信型ウイルスは 1999 年 3 月ごろに見られた Melissa[13] 以降、主流となった。それまでは、ウイルスはファイルに感染し、感染したファイルをメールあるいはファイルの共有、フロッピーなどの持ち運び可能な記憶媒体を介して他の計算機に感染していくのが主流であった。

Melissa はメーラーの Outlook に登録されているアドレス帳を用い、 50 のアドレスに対してウイルスに感染したファイルを送信するもので、メールが大量送信され、このウイルスの発生時にはメールサーバの負荷が高くなり、サーバダウンなどの被害も発生した。

このタイプのウイルスは、メールを多量に送信するために一度に多数の SMTP (25/tcp) コネクションを開始しようとする。そのため SMTP の接続試行回数の増加が発生し、これまでそのネットワークで見られた SMTP(25/tcp) での接続試行回数を上回った場合、本システムで検知・対応することが可能である。

図 7.3 に、モニタリングを行った実在ネットワークでの SMTP 接続試行回数を示した。この実在ネットワークでは 2 台の SMTP サーバが稼働しており、それ以

外の計算機からはSMTP のコネクションはまったく観測されなかった。必ずこの 2 台のサーバに接続後、外部に向かって送信が行われるものと推測できる。時折バースト的に通信が増えているが、それ以外では一分間の接続数は 100 前後で安定している。

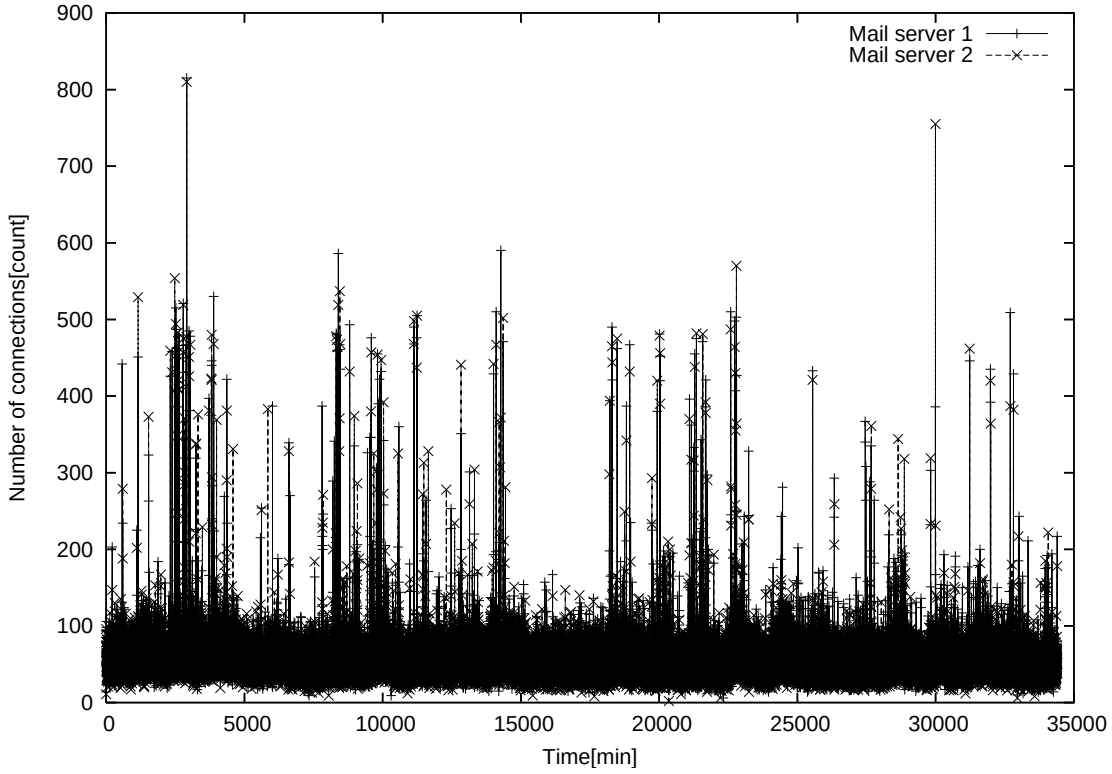

図 7.3: 実在組織の SMTP 接続試行回数

メールマガジンの発行業務などを行っている会社組織などではその発行時間 (大抵はネットワークの負荷を避けるために夜間に送信される) の閾値を手動で高めに設定しておけば、誤検知を回避することが可能であると考えられる。

Code Red, Nimda など 80/tcp を利用するワーム

全世界で 25 万台以上のマシンが感染した Code Red[14] とそのシリーズは Web サイトへの DoS を行ったり、感染先の探索に 80/tcp を用いるため、80/tcp の接続試行回数が激増する。通常 web の閲覧を行っているところに加えて、多量の

80/tcp での SYN パケットが観測されるため明らかに接続試行回数が増加していることを確認することができる。現在 Code Red は Code Red III まで出現している。 Code Red III ではそれまでの Code Red II の稼働期限が 2001 年 9 月末であったものが 34952 年 9 月末まで延長されている。

Nimda は複数の感染手法を持つが、 IIS のセキュリティ・ホールを悪用するために 80/tcp での接続試行を多量に繰り返す。このワームの出現以降、同じセキュリティホールを狙うウイルスが主流となってきており今後も同様なトラフィックが観測されるものと考えられ、本システムでの検知が可能である。

Blaster など TCP で感染拡大するワーム

Blaster ワーム [15] は 2003 年 7 月 17 日に公開された『RPC インターフェイスのバッファオーバーランによりコードが実行される』というセキュリティ・ホールを悪用する。この RPC サービスはサーバ、クライアントの種別なく Windows NT 4.0/2000/XP/Server 2003 でデフォルトで有効となっており、 135/tcp を用いてリモートで悪用可能である。実在ネットワークのモニタリングでは 135/tcp は全く観測されておらず、通常使用する可能性が低いポートである。このポートからの通信が普段行われていないにもかかわらず、多量に 135/tcp での通信が観測された場合、本システムにより即座に対応可能である。この他にも TCP コネクションを利用したワームであれば対応可能であると考えられる。

ネットワークを独占利用するロボット

Wget などの自動 web 取得ツールを組織内部ネットワークから動かす場合も、ウイルス、ワームほどではないものの、通常の web 閲覧に比べると多量の 80/tcp などでの接続試行が発生する。ウイルス、ワームはランダムに作成した接続先に対し接続試行を行うため相手先のホストが存在しない可能性が高い。しかし web 取得ツールのようなロボットは、接続先を予め手動で指定し、取得に向かうため相手先

ホストは存在する可能性は高い。ウイルスは相手先が存在しなかった場合に、接続を取消して次の接続試行を行うためある一定の単位時間内に非常に多量の接続試行を発生させるという違いが見られる。しかし通常こういったツールを動かし多量にネットワークリソースを使う行為は、管理者が認識の上行われるべきであり、もしも管理者の知らないところで多量にこのような通信を発生させた場合には、本システムで制限を行う事により、他の利用者へ迷惑をかけることを回避できる。

7.2.2 本システムの拡張・応用で対応可能な通信

本システムでは現在のところ TCP の SYN パケットの量を基準にして検知および IPFW コマンドによる接続制限を行っている。しかし他のプロトコル (UDP, ICMP) についても、使用プロトコルの判別部分を追加することによって拡張可能である。それにより具体的に下記のような例に対応できると考えられる。

Slammer ワーム

本システムに UDP プロトコル判断部分と処理を追加する事により Slammer ワームのような UDP など別プロトコルを利用した伝播抑制にも応用可能である。その場合、 TCP のように SYN パケットをカウントするのではなく、流量 (パケットサイズ×観測されたパケット個数) をカウントし、通常の利用量から大きくかけ離れたものを検知する方法が考えられる。

また IPFW のオプションとして dummynet[16] という帯域制御プログラムが存在する。今回、本システムでは dummynet の利用は行わなかったが、通常の IPFW のコマンド同様、異常検知時に渡すコマンドとして扱うことが可能である。

7.2.3 本システムが対応できない通信

本システムは高速な大規模拡散を狙うウイルス、ワームを対象に設計・実装を行った。そのため下記のような通信には対応できない。

Fly under the rader

Fly under the radar とは 十分な時間的間隔を取り、ポートスキャンなどの行為を行うことである。他の多量に行われている通信にまぎれてひっそりと行われるため、単位時間あたりに多量のSYN パケットを出すことを指標としている本システムでは対応できない。しかし、ウイルス、ワームの感染活動においては、高速な大規模拡散が一番の目的であり、それに伴うネットワークの輻輳が深刻な問題となっている。この方法は、ウイルス、ワームや侵入方法の研究を行う人々にとって、発見し対策するまでに十分な時間を与えることになる。

7.3 本システムの今後の展望

7.3.1 他 OS での実現性

本システムは、今後の拡張性を考え、広いプラットフォームでの動作を可能にする pcap ライブラリを用いて実装を行った。現在 IPFW を用いて通信制限を行っているため FreeBSD 上でのみ実装および検証を行っているが、 Linux などに搭載されているパケットフィルタリング型ファイアウォールの iptables などコマンドでルールを渡すファイアウォールであれば、本システムは応用可能であると考えられる。

7.3.2 拡張に向けての改善点

現在、 TCP の接続試行のみについて制限を行うような実装になっているが、今回の手法は他のプロトコルや、流量を判断に用いる方法でも適用可能である。

例えば UDP の流量をカウントし、通常の流量を大幅に越える場合、ファイアウォールのルールで該当通信をおこなっているホストからの通信流量の上限を設定し対応するなどという方法も考えられる。

しかしいくつかの機能を追加していく際にもっとも気を付けなければならない点は、複雑になりすぎて、管理者が扱いにくいものになってはならないということである。

第 8 章

おわりに

本研究では、定常的なトラフィック観測結果を統計化し、それを元に異常な通信の可能性を発見し、自動対応を行うシステムの設計および実装を行った。また、実存のウイルスを用い、実験環境での検知および自動対応実験を行い、その有効性を確認した。

本システムの特長は下記のとおりである。

- ログ処理の自動化
- ログ保存のためのストレージを節約
- 内部からの通信動向の把握を容易に
- 組織ネットワークのゲートウェイでの DDoS 対策
- 穏やかな通信規制の実施

本研究ではウイルス、ワームの伝播について予備実験と調査を行い、その結果を基にして特徴を把握した上で本システムの設計、実装を行った。そして、ウイルス、ワームの第一の目的である短期間での大規模拡散の抑制を、緩やかな通信規制によって実現することができた。

そのため、既存の侵入検知システムや近年新たに商用として登場している侵入防止システム (IDP: Intrusion Detection and Prevension) とは異なる優位性を示すことができたといえる。

導入や運用管理が難しいシステムは、管理コストの増大や管理者の負担増大、あるいは管理者の無関心を引き起こす。本システムは、簡単に導入し、そのままにしておくだけでも十分な自動対策が行えるだけでなく、本システムの出力から、管理するネットワークの構成の変化などに自然と管理者の目が向くように促すことができる。

謝辞

本研究を遂行するにあたっては、いろいろな方々にお世話になりました。

まず、指導教官の多田好克先生には日頃から熱心なご指導、そしてご鞭撻を賜わりました。安田絹子助手と佐藤喬助手には、研究方針や研究内容について多くの御指導を頂き、必要な機材の準備等でいつもお世話になりました。また、ご多忙中にもかかわらず論文の草稿を丁寧に読んで下さり、大変貴重なご助言をいただきました。ここに厚く御礼申し上げます。研究室 OB の楯岡孝道さんからは実験用ウイルスのご提供と実験方針についてのアドバイスを頂きました。博士課程の福田伸彦さんには本システムの実装について、貴重なアドバイスを頂きました。

そして、本研究を遂行できたことは、研究方針や方法論について議論をし、共に研究生活をおくってきた多田研、そして Vytas 研の学生諸氏おかげでもあります。最後に、これらの皆さんに感謝いたします。

参考文献

[1] JPCERT: “JPCERT/CC Alert 2003-01-27 UDP 1434番ポートへのスキャンの増加に関する注意喚起,” http://www.jpcert.or.jp/at/2003/at030001.txt.

[2] 日本エフ･セキュア株式会社: “F-Secure ウイルス情報 -Slammer-,” http://www.f-secure.co.jp/v-descs/v-descs3/slammer.htm.

[3] David Moore, Vern Paxson, and Stefan Savage, et.al,: “Inside the Slammer Worm,” *IEEE Security and Privacy*, 1(4):33-39, July 2003.

[4] Sotiris Ioannidis, Angelos D. Keromytis, Steven M. Bellovin, and Jonathan M. Smith: “Implementing a Distributed Firewall,” *ACM Conference on Computer and Communications Security , Athens, Greece, November 2000.*

[5] John Ioannidis, and Steven M. Bellovin: “Implementing Pushback: Router-Based Defense Against DDoS Attacks,” *NDSS, February 2002.*

[6] 日本電信電話株式会社 (NTT): “Moving Firewall - News Release 030218,” http://www.ntt.co.jp/news/news03/0302/030218.html.

[7] Jelena Mirkovic, Gregory Prier, and Peter Reiher: “Source-End DDoS Defense,” *Second IEEE International Symposium on Network Computing and Applications, April, 16 - 18, 2003, Cambridge, Massachusette.*

[8] S. Staniford-Chen, S. Cheung, R. Crawford, and M. Dilger: “GrIDS - a graph based intrusion detection system for large networks,” *National Information Systems Security Conference, October, 22-25, 1996.*

[9] T.Toth, and C.Kruegel: “Connection-history Based Anomaly Detection,” *Proceedings of the IEEE Workshop on Information Assurance and Security, West Point, NY, June 2002.*

[10] Gregory R. Ganger, Greg Economou, and Stanley M. Bielski: “Finding and containing enemies within the walls with self-securing network interfaces,” *Report CMU-CS-03-109, January, 2003.*

[11] 独立行政法人情報処理推進機構: “新種ウイルス「W32/Nimda」に関する情報,” http://www.ipa.go.jp/security/topics/newvirus/nimda.html.

[12] Tim Carstens: “Programming with pcap,” http://www.tcpdump.org/pcap.htm.

[13] JPCERT: “CERT 勧告 CA-99-04-Melissa-Macro-Virus,” http://www.jpcert.or.jp/tr/cert_advisories/CA-99-04.txt.

[14] JPCERT: “JPCERT/CC REPORT 2001-08-15,” http://www.jpcert.or.jp/wr/2001/wr011201.txt.

[15] JPCERT: “Windows RPC の脆弱性を使用するワームに関する注意喚起,” http://www.jpcert.or.jp/at/2003/at030006.txt.

[16] L. Rizzo: “ Dummynet: A Simple Approach to the Evaluation of Network Protocols,” *ACM Computer Communication Review, 27(1), January 1997.*