修士論文　　2011年度（平成23年度）

# 侵入検知システムの検知シグネチャ自動生成機構の設計と実装

慶應義塾大学大学院 政策・メディア研究科

重松 邦彦

平成24 年2 月13 日

修士論文要旨 - 2011 年度 (平成 23 年度)

# 侵入検知システムの検知シグネチャ自動生成機構の設計と実装

本論文では，侵入検知システムのシグネチャの自動生成機構を提案する．現在の侵入検知システムは，ベンダーやボランティアによって手作業でシグネチャが作成されており，False Positive，False Negative のトレードオフの問題や反映まで時間がかかるなど解決されていない問題が多々ある．中でも，False Positve と False Negative の誤検知の問題は深刻で，様々な検知手法が提案されている．

インターネットが広く普及した結果，悪意ある人物は経済的な利益を狙って特定の組織を狙う一方，無作為に狙った計算機を踏み台利用のために狙うこともある．しかも，攻撃者は攻撃を隠蔽し，多様な攻撃経路を使って試みる．大量の不正通信がある中，検知したい通信を検知するためには，シグネチャを自動生成し，即時反映する必要がある．

本論文で取り扱うシグネチャの自動生成の研究は，2000 年頃から提案されているものの，まだ実用化に至っていない．この研究は基礎的な部分から積み上げていく，研究の第一歩としての位置づけである．

手法としては，平常時の通信と不正な通信をそれぞれファイルテーブルに保存し，フローテーブルとセッションテーブルを作成することで通信の抽象化を試みる．このデータベースの利用により，収集した通信データ群からセッションごとのポート番号や IP アドレスや文字列などを集約し，不正な通信の特徴を定義するシグネチャを自動生成する．シグネチャは平常時の通信の誤検知を排除し，不正な通信時に出現する通信のみを検知する機能を設計，実装した．

評価方法としては，以下の点を挙げる．

- 他のシグネチャ自動生成機構との機能差異による定性評価
- シグネチャを自動作成するためのパラメータ評価

この研究の成果により作成したシグネチャを使って，未知の不正な通信群の検知に成功し，シグネチャの自動生成機構の基礎研究に貢献した．

キーワード:
1. セキュリティ， 2. 侵入検知システム，3. シグネチャの自動生成，4. マルウェア

慶應義塾大学大学院　政策・メディア研究科

重松 邦彦

Abstract of Master's Thesis - Academic Year 2011

# An Automatic Signature Generation Mechanism for Intrusion Detection System

In this thesis, I propose an automatic signature generation Mechanism for an Intrusion Detection System (IDS). In current systems, IDS signatures are made by vendors and volunteers, and we are faced with a false positive-false negative trade-off as well as concerns about the time to deploy updated signatures. The problem of false positive and false negative is difficult to solve.

The Internet is pervasive, and harbors malicious people aiming at specific economic benefits by attacking systems connected to the network. An attacker can hide the attack by using a variety of vectors. As new malware constantly appears, we need to generate and deploy the new signature in order to detect the attack traffic.

Automatic signature generation has been investigated since 2000, but is not yet deployed in practice. This study is positioned as the first step in basic research as we build a complete system using the basic parts.

My method begins by saving normal traffic and malicious traffic in separate files, allowing us to distill the traffic to create the session table and the flow table. By using this database, the port numbers and IP addresses of each traffic session from the group collected data are used to automatically generate a signature. Normal traffic is excluded from the signature, and I design and implement the system that the signature detects only bad traffic.

This approach is evaluated using three methods:
- Evaluation of the parameters for automatic signature creation
- Comparison to Snort signatures and detection rates

Using this method, we succeeded in detecting malware in unknown traffic, contributing to the advancement of automatic signature generation mechanisms for intrusion detection system.

Keywords :
1. Security, 2.Intrusion Detection System, 3.An Automatic Signature Generation Mechanism, 4.Malware

Keio University Graduate School of Media and Governance

Kunihiko Shigematsu

# 目 次

**第 1 章　はじめに** **1**
1.1 はじめに . . . . . 1
1.2 本研究の動機と目的 . . . . . 2
1.3 本研究の方針 . . . . . 3
1.4 本論文の構成 . . . . . 3

**第 2 章　背景** **4**
2.1 IDS とは . . . . . 4
2.2 既存技術とその問題点 . . . . . 4
2.3 IDS の評価の問題点 . . . . . 6
2.4 IDS の構成要素 . . . . . 6
2.4.1 検知対象による分類 . . . . . 7
2.4.2 分析方法による分類 . . . . . 16
2.5 本論文の着眼点 . . . . . 18
2.6 まとめ . . . . . 18

**第 3 章　関連研究** **22**
3.1 侵入検知システムの分類 . . . . . 23
3.1.1 データ収集機能による分類 . . . . . 23
3.1.2 データ解析機能またはログ解析機能の違いによる分類 . . . . . 23
3.1.3 アクション機能の違いによる分類 . . . . . 24
3.2 シグネチャの自動生成機構 . . . . . 24
3.3 関連研究と本研究の違い . . . . . 25

**第 4 章　アプローチ** **27**
4.1 概要 . . . . . 27
4.2 不正な通信群の作成 . . . . . 27
4.3 平常な通信群の作成 . . . . . 27
4.4 ヘッダー情報の処理 . . . . . 27
4.5 ペイロードの文字列処理 . . . . . 29
4.6 シグネチャ作成のアルゴリズム . . . . . 29
4.7 まとめ . . . . . 30

**第 5 章　設計** **31**
5.1 機能要件 . . . . . 31
5.2 DB 設計 . . . . . 31
5.2.1 テーブルの構成 . . . . . 31
5.2.2 ファイルテーブル . . . . . 32
5.2.3 フローテーブル . . . . . 33
5.2.4 セッションテーブル . . . . . 33
5.3 シグネチャのフォーマット . . . . . 35
5.4 ポート番号の抽出 . . . . . 40
5.5 IP アドレスレンジ情報の抽出 . . . . . 41
5.6 文字列の処理 . . . . . 42

**第 6 章　評価** **43**
6.1 評価手法 . . . . . 43
6.2 評価用データの作成 . . . . . 43
6.3 パラメータ評価 . . . . . 43
6.3.1 ポート番号の抽出 . . . . . 43
6.3.2 IP アドレスの抽出と IP アドレスのレンジ情報抽出 . . . . . 44
6.3.3 文字列の抽出 . . . . . 44
6.3.4 パラメータ評価 . . . . . 44
6.4 Snort のシグネチャと本手法との比較 . . . . . 45
6.5 まとめ . . . . . 46

**第 7 章　結論** **50**
7.1 まとめ . . . . . 50
7.2 今後の展望 . . . . . 50

**謝辞** **52**

# 図 目 次


2.1 不正アクセスと誤検知の関係 . . . . . 7
2.2 IDS の構成要素 . . . . . 8
2.3 ホスト型 IDS の構成図 . . . . . 9
2.4 ホスト型 IDS の概要 . . . . . 10
2.5 NIDS の構成図 . . . . . 11
2.6 NIDS の概要 . . . . . 12
2.7 一般的バッファオーバーフロー攻撃検知用のルール . . . . . 19
2.8 パターンマッチングによる攻撃検知のイメージ . . . . . 21

3.1 本研究の範囲 . . . . . 22

4.1 不正通信の取得方法 . . . . . 28
4.2 平常通信の取得方法 . . . . . 29
4.3 シグネチャ作成のアルゴリズム . . . . . 30

5.1 DB 設計 . . . . . 32
5.2 ファイルテーブル . . . . . 33
5.3 フローテーブル . . . . . 34
5.4 セッションテーブル . . . . . 35
5.5 ポート番号の抽出 . . . . . 40
5.6 通信先 IP アドレスの抽出 . . . . . 41
5.7 文字列処理 . . . . . 42

6.1 抽出したポート番号 . . . . . 43
6.2 不正なポート番号を抽出したシグネチャの例 . . . . . 44
6.3 抽出した IP アドレスのサンプル . . . . . 44
6.4 抽出した IP アドレスのレンジのサンプル . . . . . 45
6.5 IP アドレスレンジを使ったシグネチャの例 . . . . . 45
6.6 抽出した文字列のサンプル . . . . . 46
6.7 パラメータによる評価 . . . . . 47
6.8 不正データによる Snort のシグネチャと本手法のシグネチャによる比較 . . 48
6.9 平常データによる Snort のシグネチャと本手法のシグネチャによる比較 . . 49

# 表 目 次

2.1 ネットワーク型 IDS の方式と利点・損点 . . . . . 11
2.2 IDS がイベント検知後に実行するアクション . . . . . 14
2.3 ネットワーク型 IDS & ホスト型 IDS を検知機能で比較 . . . . . 15

3.1 侵入検知システムの自動生成機構のまとめ . . . . . 26

5.1 ルールオプション . . . . . 38

# 第1章　はじめに

## 1.1　はじめに

インターネットの利用形態の多様化が進むとともに，人々の生活もネットワークへの依存度が高まっている．ネットワークの障害が組織に及ぼす悪影響も増大しており，企業の持つ個人情報の流出事件が増加している．大きなニュースになった初期の情報流出事件として，02 年に発生した東京ビューティーセンター (TBC) の事件が挙げられる．この事件では，同社のアンケートや問い合わせを利用した約 5 万名の個人情報がウェブサイトから閲覧可能になっていたという [1]．住所や氏名，年齢，職業のほかスリーサイズや体の悩みなども含まれており，さらに流出した情報が Winny で二次拡散するなどして大きな事件となった．この事件原因は，会社側のウェブサイトの設定に関する不備と言える．

03 年には，ファミリーマートの「ファミマ・クラブ」から約 18 万名 [2]，ローソンの「ローソン・パス」から約 56 万名の個人情報が流出した [3]．この 2 件はそれぞれの業務委託先から情報が流出した可能性の高い事件であった．翌 03 年には，東武鉄道のメールマガジン「102@Club」会員約 13 万名の個人情報が流出した．この事件は，流出した個人宛に架空請求の郵便が届くケースが続発したことにより判明した．その後の調査で，サーバへの通常業務以外の委託会社の ID を使ったアクセスが確認されている [4]．また，コスモ石油では,「コスモ・ザ・カード」会員のうち 92 名分のカード情報が流出した．これも東武鉄道の事件同様，架空請求の郵便により判明しており，コスモ石油がシステム開発を委託していた会社の社員が不正に個人情報をダウンロードした可能性が高いとしている [5].

04 年から情報流出事件の報道が多くなったのは，03 年の「個人情報の保護に関する法律」，いわゆる個人情報保護法成立による影響が大きい．これを機に，多くの企業が個人情報の保管状況を確認し，その結果，05 年には大量の情報流出事故が発表されている．特に銀行関連で大量の個人情報紛失が相次いで発表されたが，これらの多くは「誤破棄」が原因とされている．他の企業でも流出事故原因の多くは「紛失」であった．また，シティグループ，マスターカード，カカクコム [6]，ヤフー BB でも深刻な情報流出事件が発生している．シティグループでは 390 万件以上 [7]，マスターカードは 1390 万件以上のカード情報が不正アクセスにより流出し，カード不正使用による二次被害も甚大なものとなった [8]．ヤフー BB における情報流出では，元社員が全加入者 660 万名の個人情報を持ち出し，DM に利用した事件が発生している [9].

06 年には，テレビ番組出演者の個人情報や取材候補先担当者などの個人情報が Winny 経由で流出した事件や，富士ゼロックスシステムサービス [10] や DION で流出した個人情報による恐喝未遂事件が発生している [11]．不正アクセスによる情報流出に加え，Winny

への流出や恐喝まがいの事件が目立った時期でもあった．07 年にはアメリカンファミリー生命の業務用 PC が盗難に遭い，約 15 万件の顧客情報や約 20 万件の契約情報が流出．保険業界で過去最大の流出事件となった [12]．

08 年には，音楽関連機材の通販サイトである，サウンドハウスのウェブサイトがサイバー攻撃に遭い，約 10 万名近くの個人情報が流出した．これは中国からの SQL インジェクションによる攻撃であった [13]．また，ゲームメーカのセガでは，アルバイトの個人情報がグーグルマップ経由で流出するなど，他国を経由してのサイバー攻撃や新しい流出経路が目立ち始めた年となった [14]．

10 年には海外からのサイバー攻撃が顕在化し，サミーネットワークスのオンラインゲームサイト「777townnet」から約 173 万名ものID とパスワードとメールアドレスが流出し [15]，ユニットコム（PC パーツの通販サイトであるフェイス/ツートップ）の通販サイトでも約 25 万名の個人情報が流出した [16]，さらに，スクウェア・エニックスのオンラインゲームサイト「プレイオンライン」などが攻撃を受けている [17]．

そして，11 年にはソニーグループをはじめとする大規模なハッキング，三菱重工業への不正アクセスなどが発生している．三菱重工業への不正アクセスでは，造船や国防，発電プラントなどの情報にアクセスされた可能性があり，またこれらに設置されたマルウェアは，中国，香港，米国，インドなどのサイトと通信した痕跡が確認されているという [18]．このように，企業を狙ったサイバー攻撃は年々大規模になっている．

インターネットを社会基盤としてより発展させていくためには，外部からの攻撃への対策を講じサーバを守ることが重要である．外部からの攻撃を容易に受けてしまうようなサーバでは重要なサービスを運営することができない．今後，インターネットをより安全で社会基盤として利用できるようにするためには，攻撃者からの特徴的な通信を検知し十分な耐性をサーバに持たせねばならない．

本研究では，Snort のシグネチャを不正通信から生成するシステムを提案する．Snort は無料でオープンソースの侵入検知システムであり Martin Roesch によって 1998 年に作られた [19]．コンピュータネットワークの通信内容を積極的に検査し，ネットワークの攻撃などといった，不正アクセスの疑いがあると思われるものについては，ネットワークの管理者へ攻撃の事実を通知する．

## 1.2 本研究の動機と目的

本研究の動機は，ネットワークセキュリティの確保は不可欠なものとなったものの，現実には様々な被害が発生しており，その問題を解決する必要があるからである．第 1.1 節で述べたように，現実には様々な被害が発生しており，その対策は急務である．マルウェア検知，暗号通信，セキュアプロトコルやセキュアな環境構築技術等，セキュリティ確保についての多方面からの研究が積極的に行われているが，特に，侵入検知システムの False Positive と False Negative，すなわち誤検知については，実運用上既に大きな問題となっている [20]．この問題を解決するために，シグネチャの自動生成機構を設計，開発し実用に耐えうるシグネチャ自動生成機構を提案する．

## 1.3 本研究の方針

本研究の方針は，不正な通信データからシグネチャを自動生成し，評価用の通信データを使って正しく検知することである．そのために，2 つの方針を用いて解決を図る．

1. パケットのヘッダー情報の抽象化
   不正な通信データからシグネチャを生成するためには，不正な通信データからシグネチャ化しやすいデータ群にまとめる必要がある．シグネチャは，1 つのシグネチャで複数の不正な通信を検知するのが望ましい．そこで，パケットのヘッダーと文字列を抽象化するためにデータベースを用いる．まず，通信データをファイルテーブルに保存し，フローテーブルとセッションテーブルを作成することで通信の抽象化を試みる．このデータベースの利用により，収集した通信データ群からセッションごとのポート番号や IP アドレスレンジや文字列などを集約し，不正な通信の特徴を定義するシグネチャを自動生成する．さらに，検知すべき通信からシグネチャの生成までを行う．

2. 特徴的な文字列の抽出
   パケットのヘッダー情報で抽象化した後．ペイロードの特徴的な文字列の抽出を行う．平常時の通信には出現せず，不正な通信にのみ出現する特徴的な文字列を抽出する．この抽出された文字列を使って，パケットのヘッダー情報とセットでシグネチャの生成を行う．

## 1.4 本論文の構成

本論文は 7 章から構成される．第 2 章では，侵入検知システムの概要と問題点についての背景を述べる．第 3 章では，関連研究について述べ，本研究との違いについて述べる．第 4 章では，第 2 章で述べた課題の要因を導き出し，その解決手法について述べる．第 5 章では，第 4 章で述べた設計を基に開発した実装について述べる．第 6 章では，本提案手法を定量的・定性的な側面から評価する．最後に第 7 章で本論文の結論と，今後の展望を述べる．

# 第2章　背景

本章では，侵入検知システムの果たす役割と課題について説明する．

## 2.1　IDSとは

侵入検知システムとは，ネットワークを流れるパケットやコンピュータ内の情報の状態を検査することにより，侵入やその前兆が発生してないかを判定するシステムである．情報の安全性を確保するための一つの手段としてファイアウォールとともに，侵入検知システム（以下，IDS：Intrusion Detection System と呼ぶ）が使われている．最近では，侵入を検知した場合，その通信を遮断するなどの方法で侵入を防御する機能をもつシステムもあり，これを IPS(Instrusion Prevention System) などと呼んで区別することもある．本研究で提案するシステムは，IDS である．

## 2.2　既存技術とその問題点

マルウェア検知，暗号通信，セキュア OS，セキュアプロトコル，セキュア環境構築技術等，セキュリティ確保のために多方面からの研究成果が積極的に行われている．しかし，それらの多くは末端通信ノードに何らかの変更を必要とするものであり，実利用面からの早急な問題解決に対応できないという性質がある．一方，ファイアウォールや IDS は，通信末端ノードへの変更なしにネットワークシステム全体のセキュリティ確保を実現しようとする技術であり，現状ネットワークのセキュリティ対策としての即効性に優れている．

IDS の課題としては，

1. False Positive(過剰検知) の低減
   IDS の誤検知の割合を測るための指標として，False Positive と False Negative の二つがある．False Positive とは，IDS が本来検知すべきではない，つまり不正ではない事象を不正行為として検知してしまうことを指す．通信全体の割合を図 2.1 に示す．IDS の警告をログに記録する場合，False Positive の量が次第に増えてくると，True Positive が False Positive に埋もれてしまう．False Positive を無視せずに，極力減らすべきである．しかし，False Positive を低く抑えようとすると，False Negative の発生確率が増加し，逆に False Negative を低く抑えようとすると False Positive の発生確率が増加して運用管理者への負担が大きくなる傾向がある．そのことによっ

て，IDS の運用においては，False Positive と False Negative を最小にするよう継続的にチューニングを提案する必要がある．

2. False Negative(過小検知) の低減
   False Negative とは，本来検知すべき不正行為を見逃してしまうことを指す．False Positive(過剰検知) の低減で説明したように，False Positive とトレードオフの関係がある．

3. IDS 熟練運用者の確保
   攻撃ツールやワームなどの不正侵入行為のプロセスが複雑化し複数の脆弱性を悪用するものが出現してきた現在，シグネチャの記述性の向上や関連した通信を一つの固まりとして評価する仕組みの研究が始まっている．このアプローチによる IDS の Fapse Positive 低減の努力は，今後ますます求められてくると考えられる．これは実運用面において，ネットワークの知識とその上で動作しているアプリケーションの知識から検知の精度を向上させるアプローチである．IDS の技術に熟練した技術者がいなくては，適切なシグネチャの記述が出来ず，反映までに時間がかかってしまう．

4. パフォーマンスの確保
   インターネットの接続回線の速度が，かつては 100Mbit/s が大半であったが，1Gbit/s 接続は普通に行われるようになり，今後ますます高速化が進んでいくと思われる．大量のパケットを処理するための，大規模ネットワークに耐えうるパフォーマンスの確保が望まれる．

5. 導入・運用コストの低減
   ベンダーやボランティアによって作成されたシグネチャは，FN を防ぐために FP が出やすいと考えられる．この問題を解決するために，運用者はシグネチャのチューニングをする必要性がある．シグネチャは常時更新されるため，運用者がシグネチャをチューニングするので運用コストがかかってしまう．

6. 暗号通信への対応
   通信経路上のデータを暗号化すれば，盗聴による通信内容の漏えいを防ぐことができる．半面，暗号化した通信経路上で不正な操作が行われた場合，IDS や IPS などで検知できなくなってしまう．検知できない例としては，クロスサイト・スクリプティング（XSS）や SQL インジェクションなど Web アプリケーションの脆弱性を悪用した攻撃を SSL 通信上で実施した場合などが挙げられる．特に，大規模な情報漏えいにつながるデータベースの攻撃を検知できないことは大きな問題となる．IDS, IPS を導入しても，そのメリットを大きく損なうことになる．もちろん，攻撃されても成功しないセキュアなアプリケーションを開発しておくことが理想だが，新たな攻撃手法が編み出されるもしくは診断漏れが起こると言った状況が想定され，現実問題としてそのような理想的な状態は期待できない．通信監視が行えないことは，大きなリスクを抱えることになる．

などが挙げられる [20]．特に，IDS の False Positive と False Negative，すなわち誤検知については，実運用上既に大きな問題となっている．本論文で解決する問題は，False Positive(過剰検知) の低減と False Negative(過小検知) の低減及び導入・運用コストの低減である．

## 2.3 IDS の評価の問題点

1. 検知のカウント基準
   IDS に関する議論を一般的に行う際に問題となるのが，不正行為やインシデントのカウント方法である．たとえば，文献 [21] には，14 カ月間に 76 件という数字や，1 日に 100 万件という数字が現れ，その絶対値には幅があり，その客観的基準の設定が必要と考えられる．実は，カウント対象が大きく異なるためこのような数字が出てくることになる．ここでこれらのカウントについて整理する．カウント対象を細かい単位から列挙していくと，送信元・受信先を区別しない 1 パケット単位のシグネチャマッチ数，送信元に着目したカウント，受信先に着目したカウント等となる．

   (a) 送信元・受信先を区別しない 1 パケット単位のシグネチャマッチ数

   (b) 送信元・受信先を区別しない複数パケット単位の不正アクセスパターンマッチ数

   (c) 送信元に着目したカウント数

   (d) 受信先に着目したカウント数

## 2.4 IDS の構成要素

一般的なネットワーク型 IDS は主に，不正検知と異常検知という二つの手法を用いて攻撃や不正アクセスを検知する．不正検知とは，取り込んだパケットとネットワーク型 IDS に登録された複数のシグネチャとを比較し，不正アクセスを検知する手法（Misuse Detection）である．または，異常検知（Anomaly Detection）とは，通常とは異なる振る舞いを検知する方法で，未知の手法による攻撃も発見することができる．ログイン時刻や使用コマンド，ネットワークトラフィックの状況などが判断基準となる．

カリフォルニア大学デービス校による共通侵入検知フレームワーク（CIDF：Common Intrusion Detection Framework [22]）では，4 種類の機械からなるコンポーネントを用いた IDS の構成を提案している．このフレームに基づく一般的な IDS のモデル図を図 2.2 に示す．

図 2.2 のような IDS では，各構成要素は以下の 4 つに分類できる．

- イベント生成部：Event generator（E-box）
  システム環境の中から，侵入検知に必要なイベント情報を入力として獲得する．

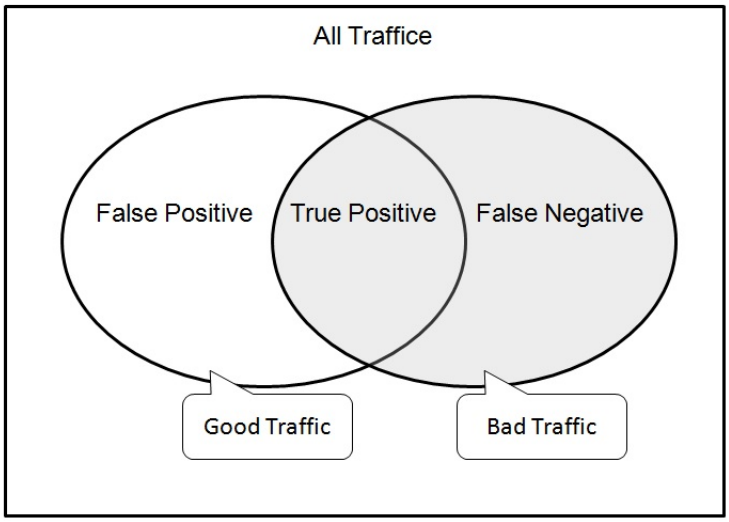


図 2.1: 不正アクセスと誤検知の関係

- イベント分析部：Event analyzer（A-box）
  侵入を検知するためにイベント解析を行うモジュール．さまざまなタイプの分析手法をこのモジュールに適用する．
- イベントデータベース：Event database（D-box）取得したイベント情報を格納しておくデータベース．
- レスポンスユニット：Response unit（R-box）検知結果に基づいて，プロセスの停止，接続の判断，ファイル設定変更，管理者への通知などの侵入に対する対処を実施するためのモジュール．

### 2.4.1 検知対象による分類

IDS は，その動作形能や監視対象によって，ネットワーク型 IDS とホスト型 IDS の二つに分類することができる．

1. ホスト型 IDS（Host-based Intrusion Detection System - ホスト型 IDS）
   ホスト型 IDS とは，監視対象となるホストのログやコマンドのヒストリから利用状況を監視し，不正アクセスと判断できるアクセスや操作を発見した場合に通知する

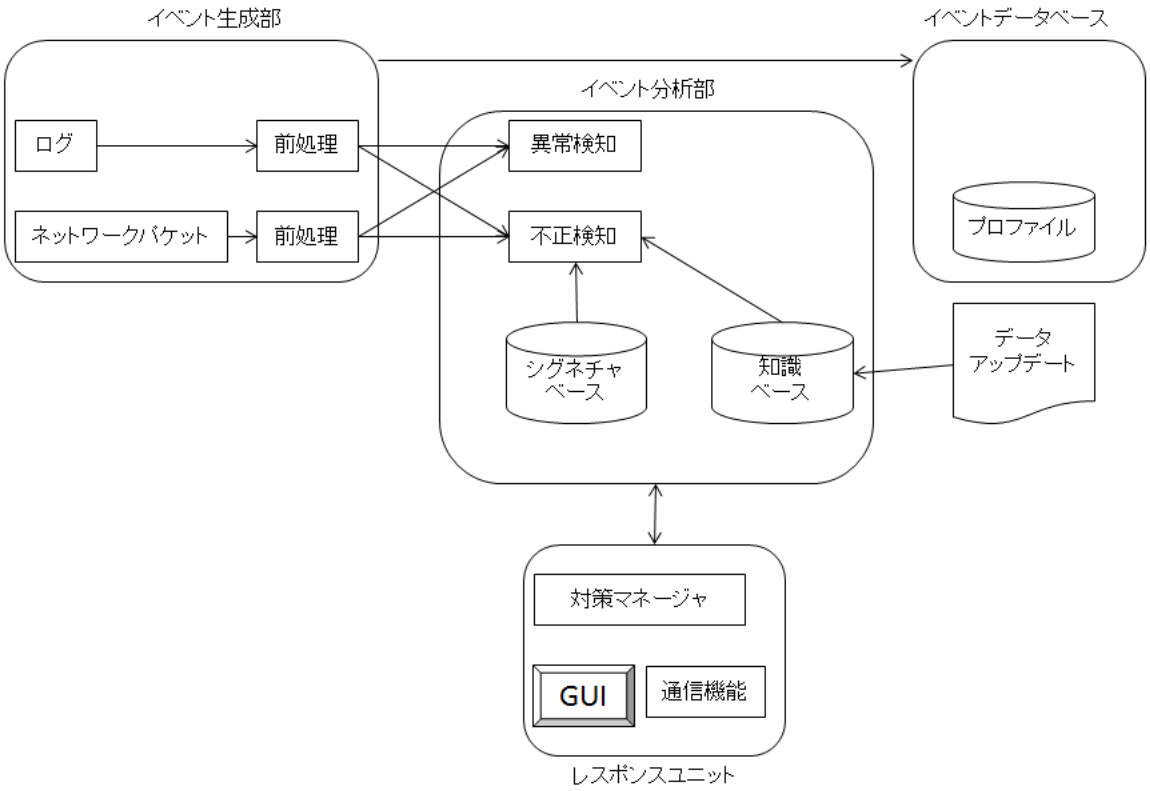


図 2.2: IDS の構成要素

システムである．ホストのネットワーク接続状況，システム・コールの状況を監視対象として考慮するものもある．ファイルの不正改ざんを検知する整合性チェック・ツールも，このカテゴリに含むことが多い．ホスト型 IDS の構成を図 2.3 に示す．ホスト型 IDS でよく用いられるシグネチャとしては，攻撃コードが生成するファイル名，プロセス名，攻撃コードが生成するファイル中のバイト列がある．代表的なツールとしては，Tripwire[23] や chkrootkit[24] がある．Tripwire は，Linux 上のファイルやディレクトリを監視し，変化があれば管理者に知らせてくれるツールであり，chkrootkit は，攻撃コードが生成するファイル名やプロセス名をシグネチャとして用いて異常があれば管理者に知らせてくれるツールである．また，現在多くのマルウェア対策ソフトもホスト型 IDS は一種で，攻撃コードが生成するファイル中のバイト列をシグネチャとして利用して検知を行なっている．ホスト型 IDS の概要を図 2.4 に示す．

シグネチャを用いないホスト型 IDS では，サーバが異常な動作をしていることを検知する方法が主流である．例えば，サーバが発行するシステムコール列の異常，サー

バが出力するログの内容の異常，サーバのファイルが改変されていることなどを検知する．Linn[25] らのシステムでは，攻撃コードが発行したシステムコールを検知する．これは，プログラム中の全てのシステムコールについて，システムコールが発行される命令の位置をあらかじめ調べておくことで実現している．こうすることにより，攻撃コードが発行したシステムコールは，システムコールを発行する命令の位置が既知の命令の位置と異なるため，システムによって検知される．ReVirt[26] らは，仮想マシンを用いたホスト型 IDS である．Revirt では，仮想マシンの状態をチェックポイントとして保存しておき，保存以降に発生したハードウェア割り込みなどの非決定性のあるイベントを全てログに保存しておく．このようにすることで，サーバが攻撃されたことを検知した後に，攻撃の実行過程を全て再生することができる．しかし，これらのシステムは攻撃された後のサーバを調べるものであり，攻撃を止めることは出来ない．

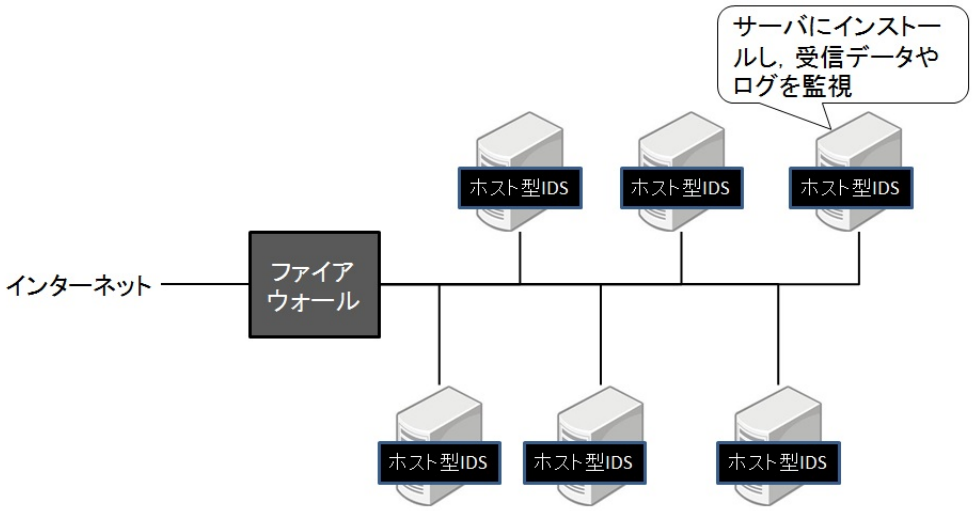


図 2.3: ホスト型 IDS の構成図

2. ネットワーク型 IDS（Network-based Intrusion Detection System - NIDS）
ネットワーク型 IDS とは，ネットワークを流れるパケットを監視し，不正なパケットが通過した場合や不正なプロトコルによる通信が行われた場合，またネットワーク・トラフィックなどに通常とは異なる状態が発生した場合に検知して通知するシステムである．ネットワーク型 IDS の構成を図 2.5 に示す．ネットワーク型 IDS は，監視専用の機器を監視対象となるネットワークセグメントに接続して使用する．そ

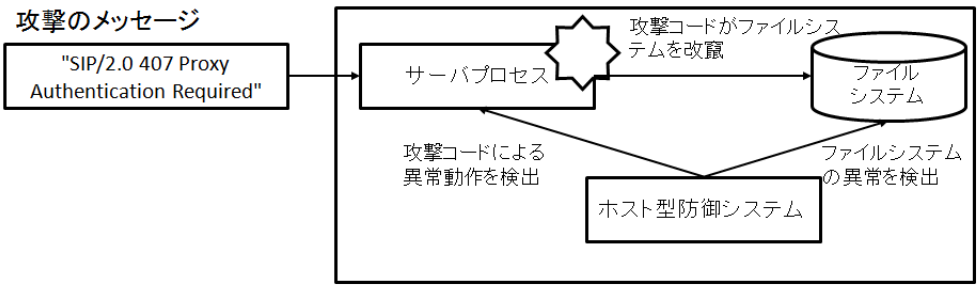


図 2.4: ホスト型 IDS の概要

の機器は，自身が接続されたネットワークを流れるパケットをリアルタイムに監視し，あらかじめ設定されたルールに基づいて不正なアクセスや不審な事象などを検知する．検知した結果は管理用のソフトウェアがインストールされた端末の画面に通知したり，メールによって通知したりすることが可能である．ネットワーク IDS の概要を図 2.6 に示す．

ネットワーク型 IDS には，自ホスト宛のパケットのみを監視するもの，同一ネットワーク内の他ホスト宛のものを含む全トラフィックを監視するものの 2 つのタイプがある．また，パケットのヘッダー情報のみを見て検知処理を行うものと，トラフィックの内容までを監視対象とするものがある．ネットワークの全トラフィックを監視する場合，ネットワーク内での配置や取得データの選択が検知に大きく影響することになる．

ネットワーク型 IDS には，シグネチャ方式と非シグネチャ方式がある. その利点と欠点について表 2.1 で整理する．シグネチャ方式は，攻撃方式にはシグネチャ方式と非シグネチャ方式がある．シグネチャ方式は，攻撃メッセージの特徴をあらかじめシグネチャとして記述しておき，メッセージがシグネチャと一致するかどうか検査する．この方式は様々なサーバを防御することができ，オーバヘッドがそれほど大きくないため，現在広く使われている．しかし，個々の攻撃に対してシグネチャが必要なため，新しい攻撃が現れた場合，シグネチャを作成する必要がある．

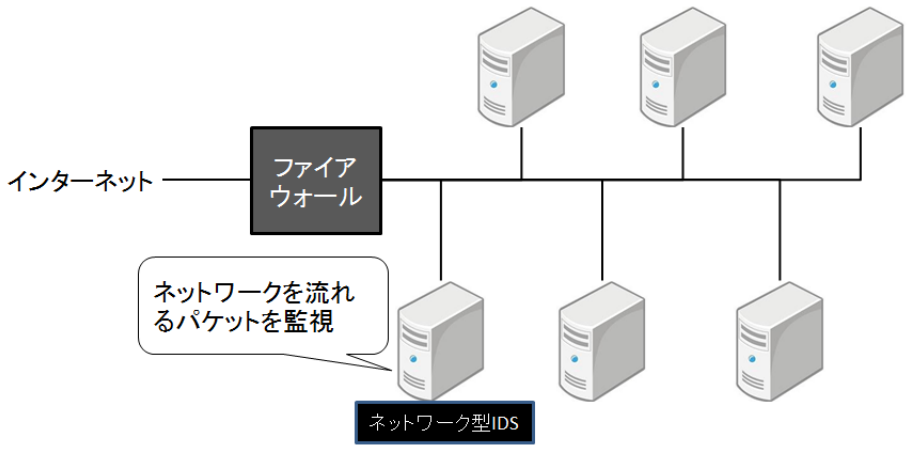


図 2.5: NIDS の構成図

非シグネチャ方式は，シグネチャを用いず，メッセージ中に一般的な攻撃メッセージとしての特徴があるかどうかを検査する．これにより，新しい攻撃が現れたとしても対応したシグネチャを作る必要がない．しかし，比較的オーバヘッドが高く，またその方式から仕様できる環境が非常に限定されてしまう．以下ではこれらの防御システムについて詳しく説明する．

広く使われているネットワーク型 IDS としては，Snort と Bro が知られている．Snort では，攻撃メッセージのバイト列をシグネチャとして用いている．そして，検査するメッセージがシグネチャのバイト列と一致した場合に，そのメッセージを攻撃メッセージと見なす．Bro ではバイト列を正規表現で表した．表現力の高いシグネチャを用いることができる [31]．シグネチャを用いるネットワーク型 IDS システムでは環境に対する縛りが比較的少なく，様々なサーバに対する防御が行える．例えば，Snort の用いるシグネチャには HTTP，FTP といったネットワーク・サーバのみならずクライアント環境の様々なソフトウェアへの攻撃に関するシグネチャもある．また，このような防御システムではメッセージを検査するオーバヘッドが比較的小さくできる．例えば，Vasiliadis らの攻撃システムではシグネチャによるメッセージの検査に GPU を利用することで，2.3Gbit/s のスループットを達成している [32]．しかし，これらのシステムでは，シグネチャは個々の攻撃に対応したものをそれぞれ作らなければならないという問題がある．この問題を解決するため，シグネチャを

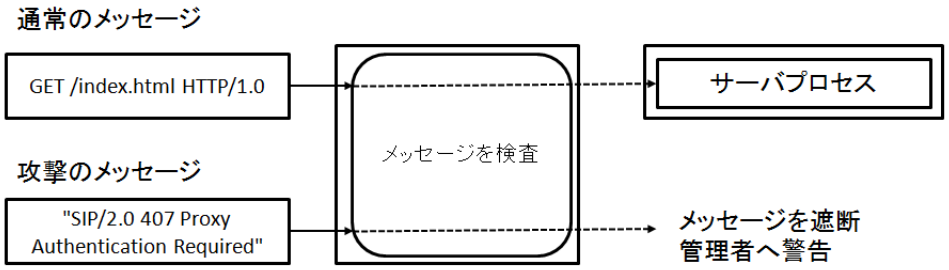


図 2.6: NIDS の概要

表 2.1: ネットワーク型 IDS の方式と利点・損点

| 方式 | 代表的なシステム | 利点 | 欠点 |
|---|---|---|---|
| シグネチャ方式 (不正検知) | Snort[19]，Bro[27] | 低オーバヘッド，多種のサーバを防御可能 | 個々の攻撃に対しシグネチャの作成が必要 |
| 非シグネチャ方式 (異常検知) | PAYL[28], SigFree[29], Network-level Code Emulatio[30] | シグネチャの作成が不要 | 高オーバヘッド，限定的用途 |

自動的に生成する研究 [33] と，シグネチャが不用なネットワーク型防御システムを作る研究 [34] が行われている．

ネットワーク型 IDS の機能上の限界や運用上の課題を以下に示す．

- 暗号化されたパケットは解析できない

通信パケットが暗号化されたとき，ネットワーク型IDSは検知できない．従って，HTTPS通信 [35] における攻撃や不正なアクセスを検知するには，SSLアクセラレータ [36] を導入する必要がある．その他の暗号化通信については，個別に検討が必要である．

- 攻撃は検知できても侵入を検知できない
  ネットワーク型IDSは攻撃が行われたことを検知するが，実際に侵入が行われたことは検知できないことが多い．
- 不正なアクセスを防御できない
  インライン接続で使用する機器ではないため，ネットワーク型IDSで攻撃を排除するには限界がある．不正アクセスを防御するために，IPS[37] やウェブアプリケーションファイアウォール [38] を利用する必要がある．
- アプリケーションに対する攻撃を検知できない
  SQLインジェクションをはじめ，組織がサイト用に独自の仕様で開発したアプリケーションプログラムの脆弱性を突いた攻撃などはほとんど検知できない．
- 誤検知
  第 2.3 項で前述したIDSの誤検知によって，ネットワーク型IDSにおける誤検知の割合を測る指標として，False PositiveとFalse Negativeがある．False Positiveとは異常な行為が発生しているにもかかわらず，それを不正として検知できないことを指し，False Positiveとは正常な行為を不正として検知してしまう誤検知のことを指す．

IDSによって，イベントを検知した後で，表 2.2 に示すようなアクションを実行することが可能がある [39]．

また，ネットワーク型IDSとホスト型IDSを主に検知機能で比較し，表 2.3 にまとめる．

| 分類 | 応答内容 | NIDS | HIDS |
|---|---|---|---|
| 通知・記録機能 | 管理用コンソールへのアラート通知 | ○ | ○ |
| | 指定されたアドレスに対してメールで通知 | ○ | ○ |
| | 指定されたホストにSNMPトラップを送信 | ○ | ○ |
| | ログ出力（ローカル環境） | ○ | ○ |
| | Syslogサーバへのメッセージ送信 | ○ | ○ |
| プログラム実行・停止機能 | 指定されたプログラムの実行又は停止 | ○ | ○ |
| ファイルやレジストリの復元 | 変更が確認されたファイルやレジストリを保存しているオリジナルの状態に復元 | × | ○ |
| セッション切断，接続制限機能 | TCPコネクションの切断（RSTパケットの送信） | ○ | × |
| | UDP， ICMPの遮断（ICMP port unreachableの送信） | ○ | × |
| | ファイアウォールのACLを動的に変更して防御 | ○ | × |
| | アカウントのロックアウト，ログインの拒否，上位権限への昇格制限 | × | ○ |
| | 特定のファイルへのアクセス制限 | × | ○ |
| | 受信した特定のパケットの破棄 | × | ○ |
| その他その他 | 検知した通信に対する応答 | ○ | × |

表 2.2: IDSがイベント検知後に実行するアクション

表 2.3: ネットワーク型 IDS & ホスト型 IDS を検知機能で比較

| 項目 | NIDS | HIDS |
|---|---|---|
| 導入方法 | 監視の対象となるネットワークセグメントに接続（専用のハードウェアが必要） | 監視の対象となるホストにインストール（専用のハードウェアは不要） |
| 監視方法 | 流れているパケットを監視（暗号化されたパケットは監視不可） | 常駐しているホスト上で行われている操作やポートの状態などを監視 |
| ポートスキャン | ○ | ○ |
| BOF 攻撃 | ○ | ○ |
| DoS 攻撃 | ○ | △ |
| サーバプログラムへの各種コマンドの発行 | ○ | × |
| ログインの成功・失敗 | △ | ○ |
| SQL インジェクション | △ | △ |
| 重要なリソースへのアクセス | × | ○ |
| 不正なプログラムのインストール | × | ○ |
| Web コンテンツの改ざん | × | ○ |
| メールによるファイルの流出 | × | ○ |
| 攻撃の排除方法 | × | × |

△ は一部検知可能を意味する

### 2.4.2 分析方法による分類

IDS は異常検知（Anomaly Detection）と，不正検知（Misuse Detection）という 2 つの手法に分類することができる．

1. 異常検知（Anomaly Detection）
   システムやユーザーの正常時における振る舞いを記録したプロファイルデータを用意しておき，このデータと取り込んだパケットを比較し，大きく異なったものを異常として検知する手法が異常検知である．不正なデータを直接検知するのではなく，通常と異なる行為を検知するので，まずユーザのシステム利用のログイン時刻や使用時間など，ネットワークトラフィックの状況から，正常な振る舞いを定義しなければならない．これ以後，システムやユーザの振る舞いを監視し，正常状態から逸脱するときに警告を発する．例えば，“ 1秒当たりに 5MB の通信量が存在する状態 ” が平常時であると定義されており，それを超過した場合は異常と見なすと定義されていた場合，もし 1 秒間に通信量が 10MB であれば IDS は異常であることと判断し，警告を発する．

   異常検知という手法によって，検知可能な事象には次のようなものがある．

   - プロトコルの仕様に反したデータの流れ
   - プロトコルの仕様に従っていないヘッダ情報を持つパケット
   - 大量に発行されたコマンドなど

   この方法では，異常な行為を見つけるものにしたがって，未知の攻撃（zero-day attack[40]）や正常なパケットを大量に送りつけるタイプの DoS 攻撃 [41] などに対応することが可能である．しかし，システムの変動やユーザの操作などに大きく影響を受けるという欠点があり，正常なのに異常として警告をする誤検知（False Positive）の発生確率が高いと言われている．一般に異常検知の手法を用いた IDS は，システムやユーザの振る舞いの変化を検知するために，侵入以外の原因による状態の変化も異常として検知してしまう．このために，侵入がないときに侵入があるかのように警告を発してしまう．False Positive をはじめ，異常検知システムを実用するために，以下の問題点を解決する必要がある．

   - 正常プロファイルの作成の問題
     異常検知において，正常な状態と異常な状態を区別するという技術があり，なにを「正常」と定義するのかという問題がある．正常プロファイルを作成している時に不正なことが行われていないことを保証するのは難しい．その上で，大学をはじめ，変動が激しい使い方が想定されるシステム管理者の環境において不正の区別が困難であるという問題も存在する．
   - しきい値など各種パラメータを設定する手法の問題
     異常検知手法を用いた IDS を実用化にするために，さまざまなパラメータによって検知精度をチューニングできる．これらのパラメータは，誤警報数の推

移を見極めつつ経験的に設定し，本章で紹介したようなアルゴリズムによる自動設定によって決定する．ただし，このようなパラメータに関する一般的な手法を確立するという困難がある．

- シグネチャの作成データが操作される可能性
  順次にユーザプロファイルを学習するアノマリディテクション型IDS は，意図的にプロファイルを操作される危険性があることに注意しなければならない．たとえば，通常午前中にログインするユーザのアカウントを不正に使用する侵入者が深夜にログインしたら，異常として警告を発するが，毎日ログイン時刻を変化させていった場合，学習するようなシステムは不正なログイン時間を検知するのが難しい．

2. 不正検知（Misuse Detection）
   不正検知とは，予めデータベースに定義しておいた攻撃・侵入行為の特徴と照合して攻撃・侵入を検知する手法である．あらかじめ登録する特徴情報のことをシグネチャと呼ぶこともあるため，シグネチャベース検知と呼ぶこともある．シグネチャは，数値，文字列，ルールなどの形態をとる．既知の攻撃パターンのシグネチャを前もってデータベースに蓄積しておき，ネットワークからの入力情報と合致するシグネチャがデータベースの中で見付かる場合には，”攻撃あり”と判断できる．エキスパートシステムのルールをシグネチャを利用する例として，図2に一般的バッファオーバーフロー攻撃検知用のルールを示す．このルールは異常に長いパラメータを持つ EXEC コマンドの実行を監視しており，それを検知した時点で Alert メッセージを発する．

   不正検知手法を用いた IDS の多くが，パターンマッチングの手法を用いている．パターマッチングは，入力データにルールやシグネチャによって記述されたパターンと同一なものが含まれているか検索する処理を指す．パターンマッチングによる攻撃検知のイメージを図 2.8 に示す．

   パターンマッチングによって検知可能な事象には，次のようなものがある．

   - ポートスキャン
   - 脆弱性検査ツールによるスキャン
   - OS やサーバソフトウェアの既知の脆弱性を突いた攻撃（BOF 攻撃 [42] など）
   - ネットワークを通じて行われるパスワードクラッキングなど

   不正検知は，シグネチャによって攻撃を判断するため，シグネチャに登録されていない未知の攻撃パターンに対して，異常であるにも関わらず正常として判断し，警報を発しない False Negative を引き起こす．未知の侵入や攻撃を検知することができないため，常に新しい侵入方法に対応したシグネチャをデータベースに追加する必要がある．したがって，既知の侵入パターンをすべて検知するためには，膨大なシグネチャを必要とする．ただし，シグネチャの増加に比例して IDS の処理に負担

が増加する．また，不正検知は，トラフィックの急激な増加，偽装通信などの検知も困難である．

不正検知の特徴は次のとおりである．

- リアルタイムで検知が可能
  ネットワークの入力情報をリアルタイムでパターンと比較するため，侵入を検知するまでのタイムラグが発生しない．
- IDS によってルールの記述方法が異なる
  IDS によってユーザ定義のルール記述方法が異なることである．そのため，IDS に応じたシグネチャの記述方法をユーザが習得する必要がある．また，一部の IDS にはルールの記述方法が非常に難解なものがある．
- サイト独自のアプリケーションの脆弱性を突いた攻撃の検知が不可能
  一般的に使用されている IDS は非常に多くのシグネチャをもっているが，それらは手法や仕様がある程度公になっている市販製品や，広く使われているシェアウェア，フリーウェアなどに対する攻撃パターンである．したがって，組織がサイト用に独自の仕様で開発したアプリケーションプログラムの脆弱性を突いた攻撃などをパターンマッチングによって検知することはできない．
- 異常検知より誤検知率が低い
  IDS が捕捉しようとする特定の行為を EOI と呼ぶことがある．不正検知は，検知する EOI を利用者が規定し，その規定にしたがってパターンを記述する．そのため，パターンが適切に記述されていれば，利用者の意図しない EOI を検知することはない．そのため，異常検知に比べて False Positive を低く抑えることができる．

## 2.5 本論文の着眼点

第 2.3 節で述べたように，Snort を利用してシグネチャを自動生成する．本論文では，管理者の負担が少なくなるようにネットワーク型 IDS のシグネチャを作成することを目的として，不正な通信からできるだけ汎用的なシグネチャを作成することである．作成したシグネチャが正常な通信では検知せず，不正な通信のみ検知するシグネチャの生成方法を提案する．

## 2.6 まとめ

本章では，IDS の基本的なことについて挙げた．検知対象により分類し，ネットワーク型 IDS とホスト型 IDS がある. 検知方法に基づき分類した場合，シグネチャ型 IDS と非シグネチャ型 IDS に分類することができる．また，分析手法に基づき分類した場合，異常検知と不正検知に分類することができる．

1 alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"EXPLOIT esignal STREAMQUOTE buffer overflow attempt"; flow:to_server, established; content:"¡STREAMQUOTE¿"; nocase; isdataat:1024, relative; content:!"¡/STREAMQUOTE¿"; within:1054; nocase; reference:bugtraq, 9978; classtype:attemptedadmin; sid:2489; rev:2;)
2 alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"EXPLOIT esignal SNAPQUOTE buffer overflow attempt"; flow:to_server, established; content:"¡SNAPQUOTE¿"; nocase; isdataat:1024, relative; content:!"¡/SNAPQUOTE¿"; within:1052; nocase; reference:bugtraq, 9978; classtype:attemptedadmin; sid:2490; rev:3;)
3 alert tcp $EXTERNAL_NET any -> $HOME_NET 548 (msg:"EXPLOIT AFP FPLoginExt username buffer overflow attempt"; flow:to_server, established; content:"—00 02—"; depth:2; content:"?"; within:1; distance:14; content:"cleartxt passwrd"; nocase; byte_jump:2, 1, relative; byte_jump:2, 1, relative; isdataat:2, relative; reference:bugtraq, 10271; reference:cve, 20040430; reference:url, www.atstake.com/research/advisories/2004/a050304-1.txt; classtype:attemptedadmin; sid:2545; rev:4;)
4 alert tcp $EXTERNAL_NET any -> $HOME_NET 749 (msg:"EXPLOIT kadmind buffer overflow attempt"; flow:established, to_server; content:"—00 C0 05 08 00 C0 05 08 00 C0 05 08 00 C0 05 08—"; reference:bugtraq, 5731; reference:bugtraq, 6024; reference:cve, 20021226; reference:cve, 20021235; reference:url, www.kb.cert.org/vuls/id/875073; classtype:shellcodedetect; sid:1894; rev:8;)
5 alert tcp $EXTERNAL_NET any -> $HOME_NET 751 (msg:"EXPLOIT kadmind buffer overflow attempt"; flow:established, to_server; content:"—00 C0 05 08 00 C0 05 08 00 C0 05 08 00 C0 05 08—"; reference:bugtraq, 5731; reference:bugtraq, 6024; reference:cve, 20021226; reference:cve, 20021235; reference:url, www.kb.cert.org/vuls/id/875073; classtype:shellcodedetect; sid:1895; rev:8;)
6 alert tcp $EXTERNAL_NET any -> $HOME_NET 749 (msg:"EXPLOIT kadmind buffer overflow attempt"; flow:established, to_server; content:"—FF FF—KADM0.0A—00 00 FB 03—"; reference:bugtraq, 5731; reference:bugtraq, 6024; reference:cve, 20021226; reference:cve, 20021235; reference:url, www.kb.cert.org/vuls/id/875073; classtype:shellcodedetect; sid:1896; rev:8;)


図 2.7: 一般的バッファオーバーフロー攻撃検知用のルール

7 alert tcp $EXTERNAL_NET any -> $HOME_NET 751 (msg:"EXPLOIT kadmind buffer overflow attempt"; flow:established, to_server; content:"—FF FF—KADM0.0A—00 00 FB 03—"; reference:bugtraq, 5731; reference:bugtraq, 6024; reference:cve, 20021226; reference:cve, 20021235; reference:url, www.kb.cert.org/vuls/id/875073; classtype:shellcodedetect; sid:1897; rev:8;)
8 alert tcp $EXTERNAL_NET any -> $HOME_NET 749 (msg:"EXPLOIT kadmind buffer overflow attempt"; flow:established, to_server; content:"/shh//bi"; reference:bugtraq, 5731; reference:bugtraq, 6024; reference:cve, 20021226; reference:cve, 20021235; reference:url, www.kb.cert.org/vuls/id/875073; classtype:shellcodedetect; sid:1898; rev:8;)
9 alert tcp $EXTERNAL_NET any -> $HOME_NET 751 (msg:"EXPLOIT kadmind buffer overflow attempt"; flow:established, to_server; content:"/shh//bi"; reference:bugtraq, 5731; reference:bugtraq, 6024; reference:cve, 20021226; reference:cve, 20021235; reference:url, www.kb.cert.org/vuls/id/875073; classtype:shellcodedetect; sid:1899; rev:8;)
10 alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"EXPLOIT gobbles SSH exploit attempt"; flow:to_server, established; content:"GOBBLES"; reference:bugtraq, 5093; reference:cve, 20020390; reference:cve, 20020639; classtype:misc-attack; sid:1812; rev:5;)

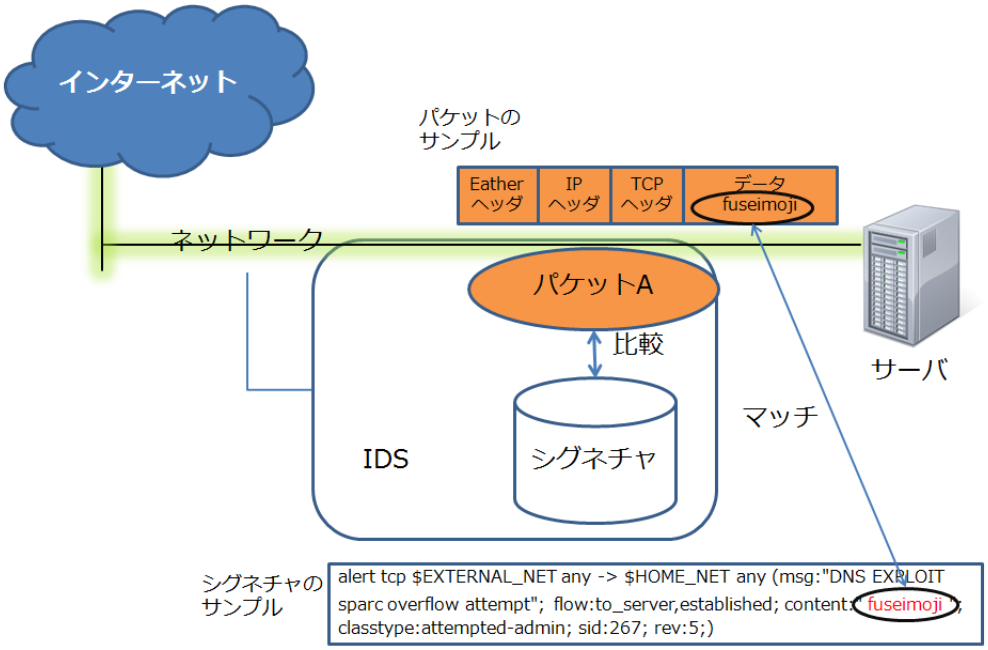


図 2.8: パターンマッチングによる攻撃検知のイメージ

# 第3章　関連研究

本章では，本研究の関連研究についてまとめる．まず，外部からの攻撃に対して防御する仕組みの原則について説明し，本研究対象をを図 3.1 に示す．本章では，まず第 3.1 節において，侵入検知システムの分類について説明し，第 3.2 節について説明する．これらの関連研究と本研究との違いを，第 3.3 節で説明する．

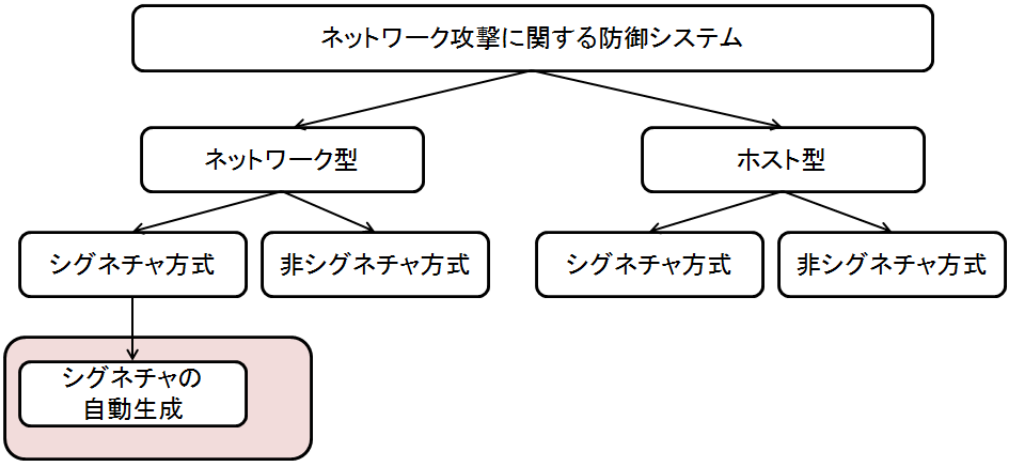


図 3.1: 本研究の範囲

## 3.1 侵入検知システムの分類

### 3.1.1 データ収集機能による分類

IDS は，データ収集機能の違いによりネットワーク型，分散型，ホスト型に分類される．ネットワーク型 IDS は，ネットワークインターフェースをデータ収集源として，そこを流れるパケットを監視することにより，不正な通信を検知するものである．通常ファイアウォールと同様の位置に設置することにより，広範囲の監視を可能としている．例えば，インターネットと組織内ネットワークの境界に設置することにより，組織内ネットワークのすべてのホストに対するインターネットからの不正行為を検知することが可能となる．検知可能な不正行為としては，例えば，DDoS:Destributed Denial of Service Attack(分散サービス不能攻撃)，広範囲のネットワークアドレスに対するポートスキャン攻撃などがある． ネットワーク型 IDS の変形として，分散型がある．これは，ネットワーク型 IDS のデータ収集機能を，場合によってはデータ作成機能など他の機能を含めて，ネットワーク上に複数配置した IDS のことで AAFID[43] があり，広域での侵入検知や，多様なデータ解析機能の連携効果が特徴である． ホスト型 IDS は，ホスト内の各種ログファイル，ソケットインターフェースやメモリの状態などをデータ収集源とし，変更や改ざんまたは定義外のアクセスを監視し不正を検知するものである．監視対象のホスト個々にインストールする必要があるため，実運用では，組織内の主要なホスト，例えばインターネットから直接アクセスを受ける Web サーバや FTP サーバなど適用範囲を絞って利用することが多い．適用範囲がホスト単位に限定される反面，当該ホストのメモリ利用状態などネットワーク型 IDS では収集不可能なホストごとの詳細な情報を利用して不正行為の検出ができるという利点がある．検知可能な不正行為としては，例えばバッファーオーバフロー攻撃，不正なログイン等がある．

### 3.1.2 データ解析機能またはログ解析機能の違いによる分類

IDS は，データ解析機能またはログ解析機能の違いによりシグネチャマッチ型（または，不正検知型），ログ解析型に分類される．

シグネチャマッチ型は，事前に定義されている不正アクセスパターンと収集したデータをパターンマッチ等により比較し不正を検知するものである．不正アクセスパターンをいかに正確に，ないしは，厳密に記述できるかで，誤検知（つまり，False Positive や False Negative）の度合いが大きく変化する．既に知られているマルウェアの検知に優れている一方，未知や亜種の不正行為を検知するには，シグネチャを常に最新の状態に保持し続けなければならないという煩雑さの解決も運用上の一つの課題となっている．

異常検知型 IDS は，正常な利用状態におけるシステムやユーザの挙動をもとにプロファイルを作成し，このプロファイルと収集したデータを比較しその差異が大きい場合に不正

として検知するものである．概念的には未知の不正行為も検知できる能力を持っているが，実際はバッファーオーバーフロー攻撃検知のような特定条件下でその能力を発揮するに止まっており，正しいとされるプロファイルを作成するのに時間がかかる．作成したプロファイルは組織のセキュリティポリシや運用状態によって変化するため，シグネチャのようにネットワークコミュニティ全体の財産として共有し相互利用することが難しいという欠点もある．

異常検知型 IDS の研究は古くから行われており，IDS 研究はここから始まったといえるが [44][45]，近年まではシグネチャマッチ型 IDS の研究が主であり，異常検知型 IDS の概念は，ログ解析型 IDS で利用されてきた．ホスト型 IDS において，バッファーオーバーフロー攻撃の検知等を目的として，関数の呼び出し前後に関数呼び出しが正しく実行されているかを検証する処理を挿入する方法 [46] 脆弱な変数に関連付けて作成した検査子を検証することによりバッファオーバーフローを検知する方法 [47]，CPU の例外デバック/命令トレース機能を用いて実行アドレスの遷移を監視する方法 [48]，バイナリーコードの情報と実行時のスタックの情報を照合することによって制御フローの異常を検知する方法 [49] 等の提案が行われるようになってきており，特定の攻撃に対するものではあるが，異常検知型 IDS の研究も盛んになってきている．

DShield は，不正行為発生情報をインターネット利用者に提供することを目指したログ解析機能を実現するシステムであるともいえるが，これと同じ視点に立ち，ログ解析機能に着目した研究も行われている． 実用的な利用として，セキュリティ監視センタにおける監視等を目的とした研究が進んでいる．ログ解析機能の研究では，データマイニングやテキストマイニングの手法をネットワークセキュリティに適用するアプローチも多く見られる．

### 3.1.3 アクション機能の違いによる分類

IDS は，アクション機能の違いにより，狭義の IDS と IPS に分類される．狭義の IDS は，データ解析部が生成する不正行為に関する判定結果等を受け，ネットワーク管理者に不正行為が発生したことを通知したり，ログに記録するなどの不正行為の検知機能のみが実現されており，防御機能はない．

一方，IPS は，これら狭義の IDS の機能に加え，不正行為が発生している通信を遮断し，ハニーポットへ誘導するなど，不正行為が実際の被害に繋がらないようなアクションを発生させるものである．この分類に沿って，狭義の IDS について，いくつかの既存システムを分類すると表 1 になる．

## 3.2 シグネチャの自動生成機構

シグネチャ方式の欠点を補うため，シグネチャを自動生成するシステムが提案されている．このようなシステムでは，攻撃に共通するバイト列をシグネチャとして取り出すことで，新しい攻撃に対して素早くシグネチャを作ることができる．しかし，これらの自

動生成されたシグネチャは攻撃に対して最適なシグネチャではないことが多い．例えば，Singh らによる EarlyBird[50] はワームに対するシグネチャを自動生成するシステムである．EarlyBird では，内部ネットワークから共通なバイト列を持つメッセージが大量に発生した際に，その共通なバイト列をシグネチャとする．これらのシステムでは共通のバイト列しか見ていないため，生成されたシグネチャに攻撃に関係ない部分のバイト列が含まれてしまう．また，Kreibich らの HoneyComb[51] はサービスを提供しない囮のホストを用いて攻撃メッセージの収集を行い，同じポート番号で収集されたメッセージに共通するバイト列を取り出してシグネチャとする．このため，人手で作成したシグネチャに比べて精度は低いと考えられる．Mahoney[52] の方式では，学習データが tcpdump ファイル，特徴ベクトルがパケットヘッダの各変数であり，機械学習アルゴリズムが確率計算である．確率計算では，ある事象が与えられるとき，その事象が攻撃である確率を計算している．そして，その攻撃モデルによる確率計算からサンプルの攻撃有無を判定する．学習データを必要とするため，教師あり学習は教師なし学習に比べ検知率が高いが教師情報を含む学習データを必要とする．また，実環境において攻撃発生時刻や攻撃種別などの教師情報を含む学習データを得ることは困難である．

## 3.3 関連研究と本研究の違い

本研究において，シグネチャの自動生成機構を提案する．
これまでの関連研究を，表 3.1 にまとめた．シグネチャの自動生成はハニーポットや DMZ などから正常な通信を含めてシグネチャが作成される．例えば，EarlyBird は，DMZ に設置したハニーポットに連続したバイトシーケンスを異常な通信と判断し，シグネチャを作成する．中には，正常な通信も含まれてしまうため，False Positive が多い．対象とする攻撃も Worm に限られたものが多く，False Negative が多発し，実用に耐えることが難しい．

本研究ではシグネチャ自動生成における基礎研究の第一歩である．不正通信のみ出現する通信の特徴からシグネチャを自動生成し，平常な通信に通信に出現する特徴は排除する．このような方針での先行研究はまだ存在しないと考えられる．False Positive と False Negative を減らし，シグネチャの即時反映ができることから，関連研究のシグネチャ自動生成機構と比べて，どの通信をシグネチャ化したか把握しやすいという特徴がある．

表 3.1: 侵入検知システムの自動生成機構のまとめ

| | Traffic source | Generates Contextual Signature | Semantics Aware | Signature Generation Algorithm | Target Attack Class |
|---|---|---|---|---|---|
| Nemean | Honey pots | Yes | Yes | (MSG)Clustering and automate learning | General |
| Autograph | DMZ | No | No | (COPP) Partitioning content blocks | Worm |
| Early Bird | DMZ | No | No | Measuring packet-content prevalence | Worm |
| Honycomb | Honey pots | No | No | Pairwise LCS across connections | General |

# 第4章　アプローチ

## 4.1　概要

シグネチャの生成には，事前準備が必要となる．まず，第 4.2 節で，不正通信の作成方法を説明し，第 4.3 節で，平常通信の作成方法を説明する．第 4.4 節で，パケットのヘッダー情報の抽象化を図る．抽象化することで，検知すべき通信を明確にし，不正な通信にのみ出現するポート番号を見つける．第 4.5 節で，作成した DB から不正通信で検知すべき通信を特定する．また，第 4.6 節で，検知すべき通信からシグネチャを作成するまでのアルゴリズムについて説明する．

## 4.2　不正な通信群の作成

不正な通信のすべてを収集することは不可能なので，マルウェアに利用される通信を不正な通信として本論文では定義する．Malware Domain List[53] では悪意のある Web サイトの URL をまとめており，攻撃コードが含まれる Web サイトも公開されている．その概要を，図 4.1 に示す．この一覧に含まれる URL を 2008 年 7 月 16 日から同年 10 月 2 日までの間に仮想マシン上で動作する Internet Explorer(Version: 6.0.2600.0000) によって閲覧し，20 分間通信を監視することにより，180 件の URL を調査した．サイズは，約 1GB である．

## 4.3　平常な通信群の作成

平常な通信のすべてを収集することは不可能なので，著者が研究室のネットワークを利用し，普段利用している平常時の通信を正常な通信群とした．その概要を，図 4.2 に示す．期間は， 2011 年 8 月 16 日〜8 月 19 日までの間にアクセスした通信データを利用している．データサイズが約 1GB に達するまで保存し続けた．320 件の URL を調査した．

## 4.4　ヘッダー情報の処理

特定の通信データを抽出するためには，全ての通信データを保存する必要がある．検知すべき通信を把握するために，各通信の通信情報を持つ DB を作成する．これは，収集し

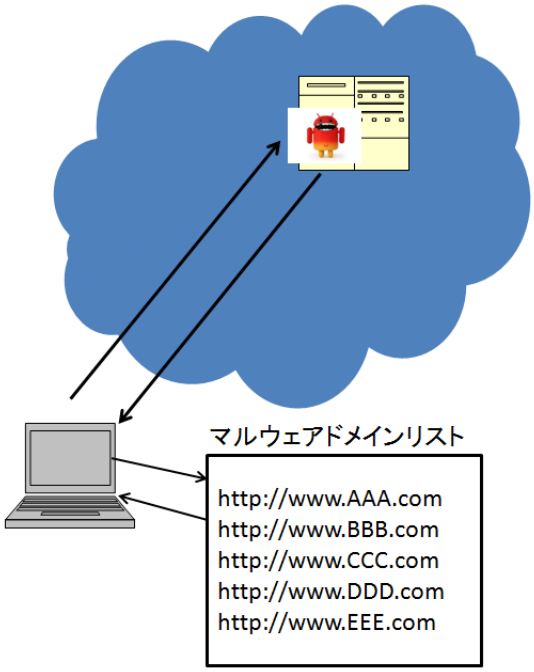


図 4.1: 不正通信の取得方法

た通信データを保存したファイルに何らかの解析をかけなければ，特定の通信データを探すためにすべての通信データを解析し，検索しなくてはいけないため効率が悪く，解析作業全体にかかる時間が増えてしまうためである [54].

まず，ファイルテーブルには通信データを保存しているファイルの特徴を格納する．ファイル名，ファイルサイズ，通信パケットの総数，通信データの総量，データ通信時間等，通信ファイルの情報を把握できる．また，確認プロトコルとして TCP，UDP，ICMP 以外を未確認プロトコルとした．フローテーブルには，同じ IP アドレスの組み合わせから IP 通信フロー毎にデータが格納されている．フローテーブルに送信元 IP アドレス，送信先 IP アドレスや送受信パケット数送受量を格納する．フローの特徴を格納することで，通信相手とのパケット数とパケット量を求めることができる．セッションテーブルには同じフロー内で発生した複数の通信について，監視対象のホストが使用した通信ポートの組み合わせを一つのセッションとみなし，セッション毎の特徴を格納している．セッションテーブルを調べることで，使用した通信ポートに関わらず，HTTP や IRC などの特定の通信が存在したかどうかを判別する．使用プロトコルは，TCP，UDP，ICMP のいずれかが格納される．また，ICMP は一つの通信パケットとして独立したものとして扱う [54].

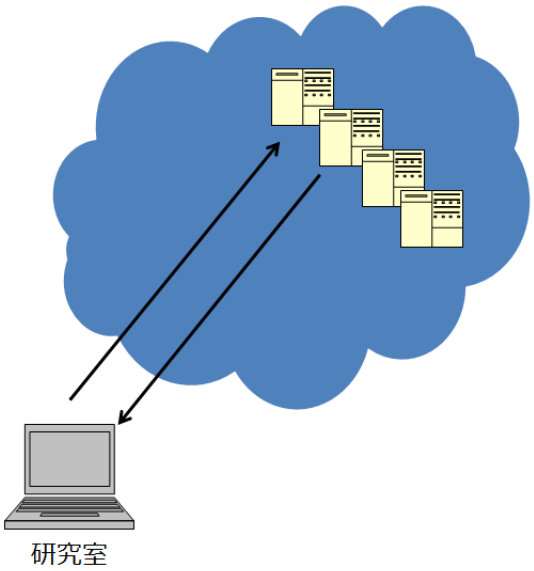


図 4.2: 平常通信の取得方法

## 4.5 ペイロードの文字列処理

ペイロードの文字列も異常な通信のみに出現する文字列を抽出する．まず，平常な通信に出現する文字列をファイルに保存し，特定の区切り文字で単語に分ける．不正な通信の文字列に出てくる単語だけをペイロードの文字列とする．

## 4.6 シグネチャ作成のアルゴリズム

シグネチャ作成のアルゴリズムは，図 4.3 に示す．まず，DB 登録処理は，不正な通信と平常な通信をそれぞれ DB 登録処理を行う．DB とは，ファイルテーブル，フローテーブル，セッションテーブルを指す．登録処理の正常終了を確認する．次に，ヘッダー抽象化処理を行う．ヘッダー抽象化処理で，不正な通信からシグネチャを作成する．文字列抽出処理では，平常な通信には含まれず，不正な通信のみに出現する文字列について取り扱う．

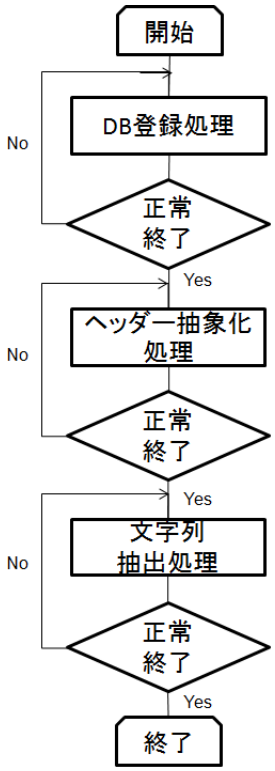


図 4.3: シグネチャ作成のアルゴリズム

## 4.7 まとめ

本章では，シグネチャ作成のための学習データの生成方法とパケットヘッダーの抽象化する方法について述べた．また，抽象化したパケットヘッダー情報と文字列を組み合わせシグネチャを作成するまでの過程を示した．第 5 では，これらの抽象化する手法の詳細について述べ，第 6 章ではこれらの手法を実際に検証し，評価する．

# 第5章　設計

本章では，4 章で述べた目的を達成するために必要な設計を行う．

## 5.1　機能要件

不正な通信をシグネチャ化するためには，まず通信を抽象化し，検知対象を明確にする必要がある．この問題の解決は通信データファイルに前処理を行うことであるが，具体的な解決方法は以下の 5 点の条件を満たしていなければならない．

1. シグネチャのフォーマットへの対応
   最終的には，抽象化した結果をシグネチャのフォーマット形式に容易に変更できる必要がある．

2. 処理の自動化
   処理はシグネチャ作成の前処理であり，検知するための通信を整理する必要がある．莫大な通信データを処理するためには，自動化することは必須となる．

3. 処理結果の保存
   処理の結果得られる情報は，処理を行った直後にのみ必要となる訳ではないため，得られた情報を保存することで，後から異なる条件で通信データを特定できる必要がある．その為にはある程度の保存領域が必要である．

4. 通信データの特徴を捉えた結果
   処理前には得られなかった通信データの特徴が，処理の結果得られることが望ましい．

5. 検索可能な結果
   処理の結果得られた通信データの情報群は研究者が一度に把握でき，任意の特徴を持つ通信データとそうでない通信データを選別できることが望ましい．

## 5.2　DB 設計

### 5.2.1　テーブルの構成

本データベースには不正な通信データの特徴となるメタデータを格納する．シグネチャ生成のために，メタデータには通信データ全体のパケット数などの特徴だけでなく，同じ

IP アドレスの組み合わせとポート番号による通信データ（フロー）毎の特徴，また同じフロー内で発生した同じ TCP/UDP 通信ポート番号の組み合わせによる通信データ（セッション）毎の特徴が含まれる．これらのメタデータを同じデータベース内に保存する場合，データベース内の別のテーブルに格納する必要がある．上記の条件に基づき作成したデータベースの構成を図 5.1 に示す．本データベースはファイルテーブル，フローテーブル，セッションテーブルの 3 つのテーブルから構成する．

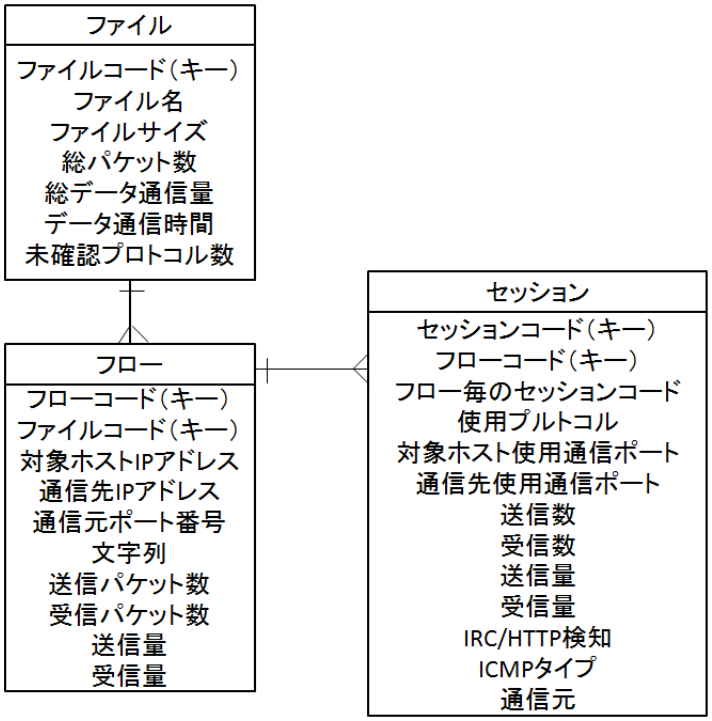


図 5.1: DB 設計

### 5.2.2 ファイルテーブル

ファイルテーブルには通信データを保存しているファイル全体の特徴を格納する．ファイル名， ファイルサイズ，通信パケットの総数，通信データの総量，データ通信時間等，通信ファイルの情報を把握できる．また，本データベースでは TCP，UDP，ICMP のプロトコルに対応しているが，上記以外のプロトコルが用いられていた場合，そのプロトコルによる通信パケット数を未確認プロトコル数として格納することで，疑わしい通信プロトコルが使用されているかどうかを判別できるようにする．その場合当該の組み合わせの通信パケットの情報は以下のフローテーブル，セッションテーブルには格納されない．ファイルテーブルの型を図 5.2 に示す．

| カラム | 型 |
|---|---|
| ファイルコード(キー) | INTEGER PRIMARY KEY AUTOINCREMENT |
| ファイル名 | TEXT NOT NULL |
| ファイルサイズ | INTEGER NOT NULL |
| 総パケット数 | INTEGER NOT NULL |
| 総データ通信量 | INTEGER NOT NULL |
| データ通信時間 | INTEGER NOT NULL |
| 未確認プロトコル数 | INTEGER |

図 5.2: ファイルテーブル

### 5.2.3 フローテーブル

フローテーブルには同じ IP アドレスとポート番号の組み合わせを利用することで，IP 通信フロー毎にデータが格納されている．フローテーブルに通信先ホストの IP アドレスや送受信パケット数等の特徴を格納することで，通信相手を調べることができる．今回使用する通信データは監視対象のホストが定まっているため，監視対象の IP アドレスでない方の IP アドレスを通信先 IP アドレスとして一つのカラムにまとめている．そのため，送受信は監視対象のホスト視点である．フローテーブルの型を図 5.3 に示す．

### 5.2.4 セッションテーブル

セッションテーブルには同じフロー内で発生した複数の通信について，監視対象のホストが使用した TCP/UDP 通信ポートと通信先のホストが使用した通信ポートの組み合わせを一つのセッションとみなし，セッション毎の特徴を格納している．セッションテーブルを調べることで，使用した通信ポート番号や，ポート番号に関わらず HTTP や IRC などの特定の通信が存在したかどうかを判別する．使用プロトコルには TCP，UDP，ICMP のいずれかが格納され，他のプロトコルについては上記にあるようにこのテーブルでは扱われない．また，ICMP プロトコルの通信は一つの通信パケットを独立した一つのセッ

| カラム | 型 |
|---|---|
| セッションコード（キー） | INTEGER PRIMARY KEYAUTOINCREMENT |
| フローコード（キー） | INTEGER NOT NULL |
| フロー毎のセッションコード | INTEGER NOT NULL |
| 使用プルトコル | INTEGER NOT NULL |
| 対象ホスト使用通信ポート | INTEGER NOT NULL |
| 通信先使用通信ポート | INTEGER NOT NULL |
| 文字列 | TEXT |
| 送信数 | INTEGER |
| 受信数 | INTEGER |
| 送信量 | INTEGER |
| 受信量 | INTEGER |
| IRC/HTTP検知 | TEXT |

図 5.3: フローテーブル

ションとして扱う．IRC/HTTP 検知カラムでは，セッションが IRC もしくは HTTP の通信であるかどうかを確認する．セッション内の一つのパケットデータの先頭に IRC 接続時に見られる命令のうち NICK，PASS，JOIN コマンドのいずれかが存在した場合，そのセッションを IRC とみなし，このカラムに IRC と入力する．

また，パケットデータの先頭に主な HTTP 命令である GET もしくは POST コマンドが存在した場合，そのセッションを HTTP とみなし，このカラムに HTTP と入力する．一般的に IRC のポートは 6667， HTTP のポート番号は 80 であるが，これらのポート番号を用いると通信が検知や遮断されてしまう恐れがあるため，これらの TCP/UDP プロトコルを用いる際に独自のポート番号で通信を行うマルウェアによる不正通信が存在する．そのため，プロトコルを判定する機能を付加した．ICMP タイプカラムでは， ICMP プロトコルの通信タイプを数値で格納する．同カラムは ICMP プロトコルの場合にしか用いない．また，ICMP プロトコルの場合は両通信ポート，送受信数，送受信量のカラムは用いられない．本データベースでは監査対象のホストと通信先のホストの通信データを格納しているが，通信を開始したのがどちらのホストであるかについてはセッションテーブルの通信元カラムに格納する． 先に通信を行ったのが監査対象のホストであった場合，同カラムに sent を格納し，また先に通信を行ったのが外部のホストであった場合，同カラムに recv を格納する．このカラムにより，セッションを開始したのがどちらのホストであるかを判別することができる [54]．セッションテーブルの型を図 5.4 に示す．

| カラム | 型 |
|---|---|
| セッションコード（キー） | INTEGER PRIMARY KEYAUTOINCREMENT |
| フローコード（キー） | INTEGER NOT NULL |
| フロー毎のセッションコード | INTEGER NOT NULL |
| 使用プルトコル | INTEGER NOT NULL |
| 対象ホスト使用通信ポート | INTEGER NOT NULL |
| 通信先使用通信ポート | INTEGER NOT NULL |
| 文字列 | TEXT |
| 送信数 | INTEGER |
| 受信数 | INTEGER |
| 送信量 | INTEGER |
| 受信量 | INTEGER |
| IRC/HTTP検知 | TEXT |
| ICMPタイプ | INTEGER |
| 通信元 | INTEGER |

図 5.4: セッションテーブル

## 5.3 シグネチャのフォーマット

シグネチャの作成は，侵入検知システムの Snort のフォーマットに従う [55]．ルールは，1. ルールアクション部，2. プロトコル部，3. アドレス部，4. ポート番号部，5. パケット方向部，6. オプション部の 6 つから成り立っている．

1alert 2tcp 3$EXTERNAL_NET 4any -> 5$HTTP_SERVERS 6$HTTP_PORTS(7msg:”WEB-IIS cmd.exe access”; 8flags:A+; 9content:”cmd.exe”; 10nocase; classtype:web-applicationattack; 11sid:1002; rev:5;)

上記の例では，ルールアクション部は 1，プロトコル部が 2，アドレス部が 3.5，ポート番号部が 4,6，ルールオプション部が 7,8,9,10,11 になる．

1. ルールアクション部

- alert
  ルールにマッチングしたパケットのアラートとログを出力する．
- log
  ルールにマッチングしたパケットのログのみ出力する．

- pass
  ルールにマッチングしたパケットを無視する.
- activate
  ルールにマッチングしたパケットのアラートを出力し，dynamic ルールに動作を引き継ぐ.
- dynamic
  activate から動作を引き継ぐまで何もせず，activate にマッチングしたルールに対して動作する.

2. プロトコル部
   プロトコル部には，tcp，udp，icmp，ip，ARP，IGRP，GRE，OSPF，RIP，IPX などが適用される.

3. アドレス部
   アドレス部には，IP アドレスを記述する場合と変数で記述する場合がある.
   - IP アドレスを記述する場合
     - 単一のホストを指定 192.168.0.1/32
     - ネットワークアドレスを指定 192.168.0.0/24
     - 複数のアドレスを指定 [192.168.10.0/24,192.168.20.5/32]
     - 全てのホストを指定 any
   - 変数を記述する場合 snort.conf ファイルに「var $HOME_NET 192.168.0.132」とあれば，アドレス部には$HOME_NET を記述する．!$HOME_NET は 192.168.0.1 以外のホストになる.

4. ポート部
   ポート番号部もアドレスと同様にポート番号を既述する場合と，変数で記述する場合がある.
   - ポート番号を記述する場合
     - 単一のポート番号を指定する場合
       80
     - ポート番号の範囲を指定する場合
       100：1024(100〜1024 番まで指定)
       ：600(〜600 番まで指定)
       500：(500 から 1024 番まで指定)
     - 全てのポートを指定する場合
       any

- 変数を記述する場合
  snort.conf ファイルに「var $HOME_NET 192.168.0.132」とあれば，アドレス部には$HOME_NET を記述する．
  !$HOME_NET は 192.168.0.1 以外のホストになる．変数が指定されていない場合はエラーになる．

5. パケット方向部
   パケット方向部は，パケットの流れを指定する．

   - 192.168.0.1/32 any -> 192.168.0.10/32 80
     192.168.0.1 から 192.168.0.10 の 80 番ポートへのアクセス

   - 192.168.0.1/32 any <> 192.168.0.10/32 80
     192.168.0.1 と 192.168.0.10 の 80 番ポートの双方向アクセス

6. ルールオプション部
   ルールオプション部は 38 個の種類がある．詳細を，表 5.1 にまとめる．

| オプション | 説明 |
|---|---|
| msg | アラートとパケットのログに出力されるメッセージ |
| logto | 通常の出力ファイルとは別に指定したファイルにパケットのログを出力 |
| ttl | IP ヘッダの TTL 値を指定 |
| tos | IP ヘッダの TOS 値を指定 |
| id | IP ヘッダのフラグメント ID 値を指定 |
| ipoption | IP オプションを監視 |
| fragbits | IP ヘッダのフラグメントビットを監視 |
| dsize | パケットペイロードのデータサイズを指定 |
| flags | TCP フラグを指定 |
| seq | TCP のシーケンス番号を監視 |
| ack | TCP の ACK 番号を監視 |
| itype | ICMP のタイプを監視 |
| icode | ICMP のコードを監視 |
| icmp_id | ICMP ECHO ID を監視 |
| icmp_seq | ICMP ECHO のシーケンス番号を監視 |
| content | パケットペイロードのパターンを指定 |
| contentlist | パケットペイロードのパターンを記載したファイルを使用 |
| offset | content のオプションで，パターンマッチングを行う開始位置を指定 |
| depth | content のオプションで，パターンマッチングを行う終了位置を指定 |
| nocase | 大文字と小文字の区別をしない |
| session | アプリケーションレイヤの記述形式を指定 |

表 5.1: ルールオプション

| オプション | 説明 |
|---|---|
| rpc | RPC サービスを監視 |
| resp | ルールにマッチしたセッションを強制切断 |
| react | ルールにマッチしたセッションをブロック |
| reference | 攻撃参照 |
| sid | Snort ルール ID |
| rev | 修正バージョン番号 |
| classtype | ルールの分類 |
| priority | ルールの危険度 |
| uricontent | パケットの URI 部をパターンマッチング |
| tag | ルールにマッチしたパケットログの拡張版 |
| ip_proto | IP ヘッダーのプロトコルを指定 |
| sameip | 送信元と送信先が同一 IP を決定 |
| stateless | Flow とともに使用.TCP に依存しない攻撃を監視 |
| regex | ワイルドカードを使用可能にする |
| flow | ルールにマッチングしたパケットを再構築する |
| flagoffset | IP フラグメントオフセット値を 10 進数で指定 |
| rawbytes | telnet のデータが異常でないかを解読 |
| distance | content と content の間が何バイト離れているかを厳密にチェック |
| within | content と content の間が何バイト以内であるかを厳密にチェック |

## 5.4 ポート番号の抽出

不正な通信にしか存在しないポート番号を抽出する．セッションテーブルにある通信先使用通信ポートを利用して，不正な通信群と平常な通信群から不正な通信群だけに出現するポート番号を抽出する．詳細を，図 5.5 に示す．

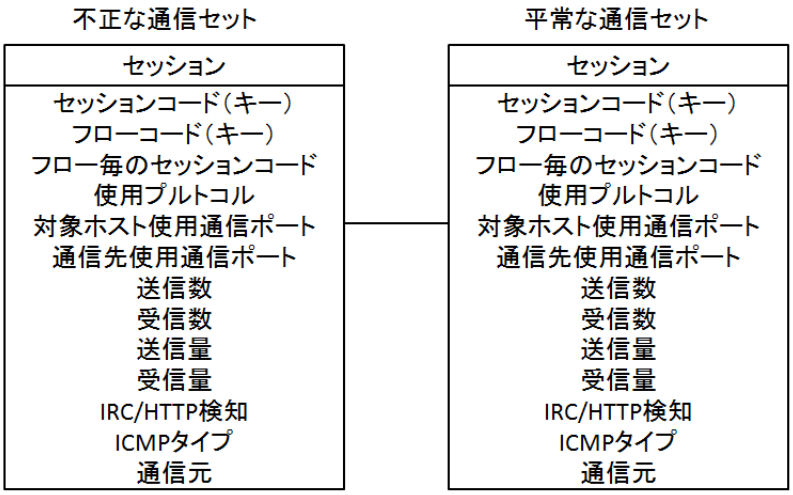


図 5.5: ポート番号の抽出

## 5.5 IP アドレスレンジ情報の抽出

不正な通信にしか存在しない送信先 IP アドレスを抽出する．フローテーブルにある通信先 IP アドレスを利用して，不正な通信群と平常な通信群から不正な通信群だけに出現する通信先 IP アドレスを利用して抽出する．ただし，5.4 章で，抽出したポート番号によって抽出したポート番号の送信先 IP アドレスを抽出しない．IP アドレスは抽象度を上げるために,「whois -h whois.radb.net IP アドレス」[56] から IP アドレスレンジ情報から取得する．詳細を，図 5.6 に示す．

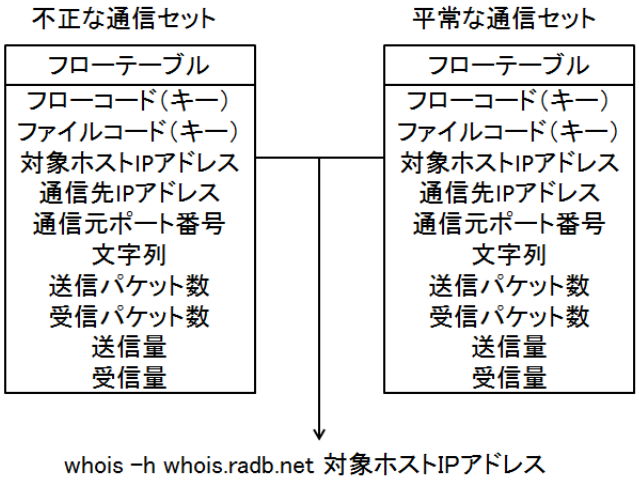


図 5.6: 通信先 IP アドレスの抽出

## 5.6 文字列の処理

不正な通信にしか存在しない文字列を抽出する．平常な通信に存在する文字列を特定の区切り文字を使って，文字列群をファイルに格納する．特定の区切り文字は,「:」「;」「,」「?」「」（スペース）であり，平常な通信群に含まれない文字列を抽出する．詳細を，図 5.7 に示す.

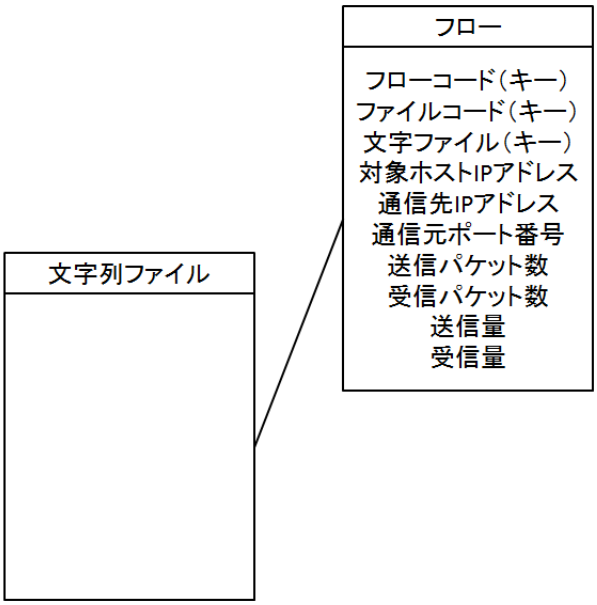


図 5.7: 文字列処理

# 第6章　評価

## 6.1　評価手法

本研究における，評価手法を 2 点述べる．シグネチャの生成には，ポート番号，IP アドレス，IP アドレスのレンジ情報，文字列を利用することを第 4 章で述べた．その構成要素の組み合わせの中で，最も検知率が上がるパラメータの組み合わせ評価する．さらに，現行の Snort と本研究で実装したシステムでの定量評価を行う．また，検知すべき通信は 1 つのセッションの中で 1 つでも検知すれば，不正を検知したものとして判断を行う．

## 6.2　評価用データの作成

評価用のデータ作成としては，第 4.2 節で作成した不正な通信と同様の取得方法で行う．ここで取得したデータは，不正通信として，この一覧に含まれる URL を 2008 年 7 月 16 日から同年 10 月 2 日までの間に仮想マシン上で動作する Internet Explorer(Version: 6.0.2600.0000) によって閲覧し，20 分間通信を監視することにより，160 件の URL を調査した．サイズは，約 1GB である．

## 6.3　パラメータ評価

### 6.3.1　ポート番号の抽出

不正な通信にしか存在しないポート番号を特定する．このポート番号は，第 5.4 で説明した実行結果を図 6.1 述べる．これらのポート番号は，不正な通信にしか存在せず，平常な通信には存在しなかったポート番号である．本システムでは，これを不正通信に利用するポート番号と特定する．

25，29，30，81，82，110，135，157

図 6.1: 抽出したポート番号

```
alert tcp any any -> any 25 (msg:"bad port number 25 "; flow:established,to_server;
sid:10000000; rev:1;)
```

図 6.2: 不正なポート番号を抽出したシグネチャの例

### 6.3.2 IP アドレスの抽出と IP アドレスのレンジ情報抽出

平常な通信には存在せず，不正な通信にしか存在しない IP アドレスを抽出する．第 5.5 で述べた IP アドレスを図 6.5 に示す．抽出した IP アドレスから，whois -h whois.radb.net のコマンドを使ってアドレスレンジを抽出した．

```
114.139.154.170
114.229.56.44
114.229.62.9
114.241.10.173
115.126.2.110
115.126.2.111
116.1.236.66
116.1.62.184
116.103.48.2
116.111.200.211
116.111.224.119
```

図 6.3: 抽出した IP アドレスのサンプル

### 6.3.3 文字列の抽出

平常な通信には存在せず，不正な通信にしか存在しない文字列の抽出を行う．不正通信軍にしか出現しない文字列の抽出結果を，図 6.6 に示す．

### 6.3.4 パラメータ評価

ポート番号と IP アドレスによる組み合わせ，ポート番号と文字列による組み合わせ，ポート番号＋ IP アドレス＋文字列による組み合わせ，ポート番号＋ IP アドレスレンジ＋

```
114.138.0.0/15
114.224.0.0/12
115.126.0.0/17
116.1.0.0/16
116.100.0.0/14
116.103.0.0/16
116.108.0.0/14
116.111.0.0/16
116.111.192.0/19
116.111.224.0/19
```

図 6.4: 抽出した IP アドレスのレンジのサンプル

```
alert tcp any any -> 114.138.0.0/15 any (msg:"bad ip address";
flow:established,to_server; sid:10000000; rev:1;)
```

図 6.5: IP アドレスレンジを使ったシグネチャの例

文字列による組み合わせをそれぞれ評価用の不正通信で評価を行なった．その結果を，図 6.7 に示す．

## 6.4 Snort のシグネチャと本手法との比較

現行の Snort の最新シグネチャと本論文の手法によるシグネチャ生成機構による検知率の比較を行う．評価用の不正データを使い，True Positive による評価を行なった．その結果を，図 6.8 に示す. 不正な通信データについては，TP が大幅に増え，Snort の最新のシグネチャより高い検知率を取得することが出来た．

評価用の平常データを使い，True Positive と False Positive による評価を行なった．その結果を，図 6.9 に示す. 平常通信による検知については，検知しないことが正しい．その通信については，True Negative の評価軸を利用する．平常な通信データを使った場合，シグネチャ生成機構によるシグネチャの方が少し，True Negative はわずかに増加し，False Positive もわずかに減少したことで，平常な通信の検知についても優位性が見られる．

E2E9CAE6-1E7B-4B8E-BABD-E9BF6292AC29
Ahhhh
My
Mouth
WHATISIT
Girl
r00t
rewt
wh00t
friday
StoogR
wank
bot.about
bot.dns
bot.open

図 6.6: 抽出した文字列のサンプル

## 6.5 まとめ

本研究で利用したアドレスレンジとポート番号と文字列から Snort のシグネチャの自動生成を行うことに成功した．IP アドレスレンジ，ポート番号，文字列の組み合わせから作成したシグネチャは，不正な通信データを利用した場合，Snort で提供されているシグネチャより高い True Positive を取ることができた．また，平常な通信データを使った場合，わずかながらも TN の増加，FP の減少に成功した．

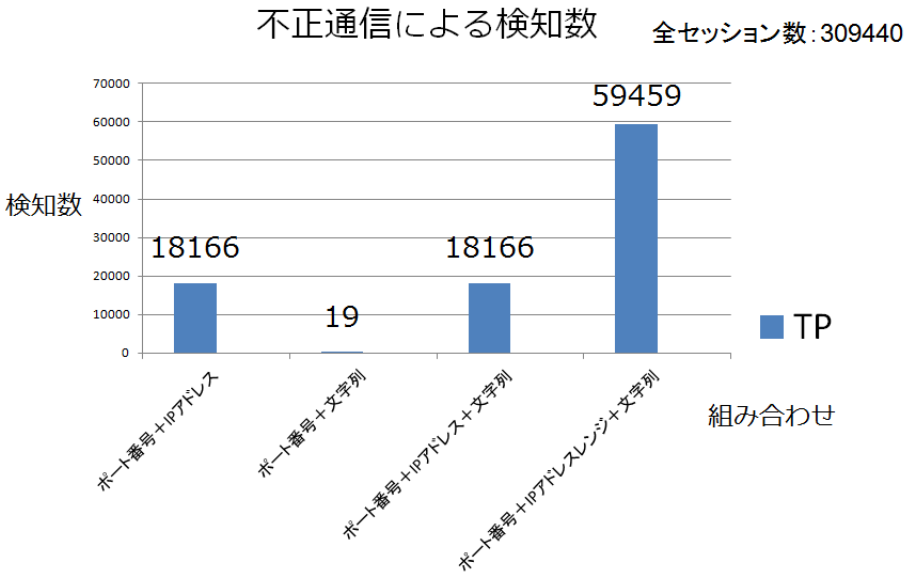


図 6.7: パラメータによる評価

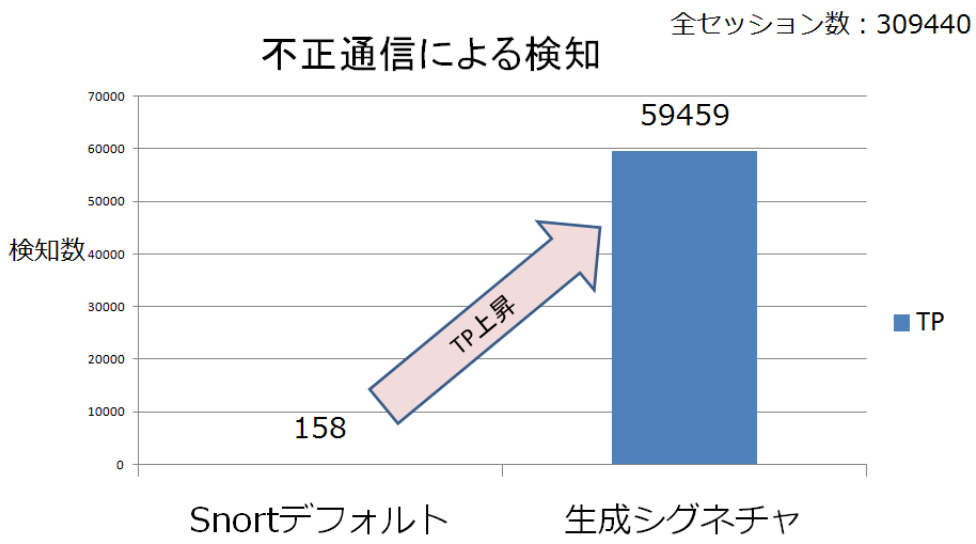


図 6.8: 不正データによる Snort のシグネチャと本手法のシグネチャによる比較

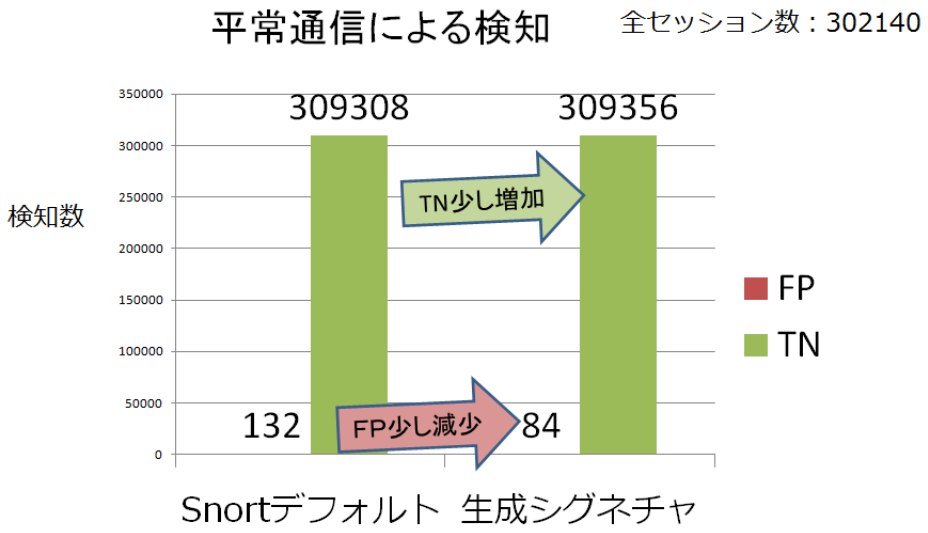


図 6.9: 平常データによる Snort のシグネチャと本手法のシグネチャによる比較

# 第7章　結論

本章では，本論文の成果をまとめ，本研究の目的を実現するために今後の展望を述べる．

## 7.1　まとめ

現在のインターネットは我々の生活に不可欠な社会基盤となっており，様々なサービスがインターネット上で動作している．しかし，一方で，悪意ある攻撃者がサービスを提供するサーバに対してネットワーク経由で攻撃を行う事例が後を絶たない．そこで，ネットワーク経由で攻撃からサーバを守ることが重要になってくる．

ネットワーク経由での Exploit コードが実行されサーバに侵入し動作する．このようなネットワーク経由での攻撃に対する防御策としてネットワーク型 IDS やホスト型 IDS のような防御システムが用いられている．現在の防御システムでは，シグネチャと呼ばれる個々の攻撃コードに特有な情報を防御のために利用している．しかし，そのような防御システムでは，シグネチャのない攻撃コードへの対処はできない．

本研究では，不正な通信データからシグネチャを自動生成し，平常な通信では検知しないシグネチャを自動生成することに成功した．さらに，パラーメタとして IP アドレスではなく IP アドレスレンジを活用し，ポート番号，文字列を活用することで，現行の Snort のシグネチャより高い検知率を達成でき，さらに平常の通信についも若干ではあるが誤検知を減らすことに成功した．

## 7.2　今後の展望

シグネチャの自動生成システムは，学習データの選択と抽象化の粒度の精査が必要となる．不正な通信，平常な通信に何をどう定義するのかは残課題である．本論文では，不正な通信をマルウェアドメインリストによるアクセスから取得し，平常時の通信を普段研究室で利用しているネットワークをキャプチャし利用した．不正な通信群の中にも平常が混在されていることを考えられるし，平常な通信も不正な通信に気付いていないだけの可能性を否定できない．本論文で作成したシグネチャ自動生成機構は，学習データによって大きくシグネチャが変化し，IP アドレスレンジの活用でもまだシグネチャの数が多いと思われる．

さらに，これまで提案されてきた自動生成機構と本手法による比較が困難であると言える．この問題は，これまでの自動生成機構でも様々な評価データが使われており，統一されていない．そのため，検知数や検知率などが絶対値として比較することができない．よって，他の自動生成機構との比較が難しく優位性を説明することがあまり出来ていない．

# 謝辞

本論文の作成にあたり，ご指導頂いた慶應義塾大学環境情報学部教授村井純博士，同学部教授徳田英幸博士，同学部教授中村修博士，同学部准教授楠本博之博士，同学部准教授高汐一紀博士，同学部准教授三次仁博士，同学部准教授植原啓介博士，同学部専任講師中澤仁博士，同学部准教授 Rodney D.Van Meter III 博士，同学部教授武田圭史博士, 同大学政策・メディア研究科特任講師斉藤賢爾博士，同大学政策・メディア研究科特別研究講師吉藤英明博士，同大学政策・メディア研究科特別研究講師佐藤雅明博士に感謝致します．特に武田圭史博士は，大学院への入学準備から指導して頂き，入学後も研究で行き詰まる私に対して根気強く指導して頂きました．本当に感謝します．

そして本研究を進めていく上で，様々な励ましと助言，お手伝いをいただきました．村井研究室 OB 水谷正慶博士, 田崎創博士，六田佳祐氏，峯木厳氏，江村圭吾氏，黒宮祐介氏，梅田昂翔氏，Doan Viet Tung 氏，福岡英哲氏に感謝します．特に水谷正慶博士には，研究の方向性や考え方などアドバイスを頂きました．

慶應義塾大学政策・メディア研究科後期博士課程岡田耕司氏，空閑洋平氏，堀場勝広氏，工藤紀篤氏，久松剛氏，松園和久氏，松谷健史氏, 鈴木詩織女史，同修士課程永山翔太氏，波多野敏明氏，上原雄貴氏，山口修平氏，佐藤弘崇氏，三部剛義氏，米村茂氏，宮崎圭太氏，澤田暖氏に感謝致します．上原雄貴氏とは，研究グループの運営や研究室での活動を手助けをして頂きました．

慶應義塾大学徳田・村井合同研究室研究グループ ISC の研究生である相見眞男氏，碓井利宣氏，Vu Xuan Duong 氏，関根冬輝氏，Pham Van Hung 氏，山本知典氏，吉原洋樹氏，有馬怜文氏，大矢崇央氏，鴻野弘明氏，小松真氏，中島明日香女史，三ツ木あかね女史，由井卓哉氏，吉原大道氏，Do Trung Kien 氏，露木航平氏，中安恒樹氏，深谷哲史氏，吉田裕氏，小澤麗女史，Nguyen Anh Tien 氏に感謝します．特に，吉原大道氏と吉田裕氏には忙しい中，研究について議論したり，論文の校正など細かな作業を手伝って頂きました．鴻野弘明氏については，苦しい論文執筆時にも研究室で楽しい一時を提供して貰いました．さらに，徳田・村井・楠本・中村・高汐・バンミーター・植原・三次・中澤・武田合同研究プロジェクトの皆様に感謝致します．

イングランドプレミリーグのアーセナルの監督であるアーセン・ヴェンゲル氏と名古屋グランパスの監督であるドラガン・ストイコビッチ氏に感謝します．ヴェンゲル氏が名古屋グランパスで繰り広げたパスサッカーを観て，サッカーの楽しさに目覚めさせられ人生を楽しいものにしてくれました．

また，ストイコビッチ氏は，名古屋グランパスの選手時代の華麗なプレーを魅せられ氏のファンとなりました．引退後も監督として万年中位だった名古屋グランパスを初のリーグ優勝に導いてくれました．両氏の存在無くして，有意義な 2 年間を過ごすことはできま

せんでした.

最後に，社会人を経て 2 年間の大学院での研究生活を許可してくれた父久雄と母陽子と私の家族に心から感謝します.

# 参考文献


[1] 勝村 幸博. 【速報】エステティック TBC の顧客情報漏洩、一人当たり 3 万 5000 円の賠償金. http://itpro.nikkeibp.co.jp/article/NEWS/20070208/261427/, 2007. 2011 年 12 月 16 日確認.

[2] 本間 純. ファミマが個人情報漏洩で釈明、流出経路は特定できず. http://itpro.nikkeibp.co.jp/free/NC/NEWS/20031119/136657/, 2003. 2011 年 12 月 16 日確認.

[3] 田中 克己. 新たな段階に入った個人情報漏洩事件・事故. http://itpro.nikkeibp.co.jp/free/ITPro/OPINION/20040119/1/, 2004. 2011 年 12 月 16 日確認.

[4] 高橋 信頼. 東武鉄道メール・マガジン会員の個人情報流出，最大 13 万 1742 件漏えいの可能性. http://itpro.nikkeibp.co.jp/free/ITPro/NEWS/20040326/142021/, 2004. 2011 年 12 月 16 日確認.

[5] 鷹木 創. コスモ石油、92 万人の個人情報漏洩〜被害者にガソリン代 500 円分付与. http://internet.watch.impress.co.jp/cda/news/2004/06/09/3425.html, 2004. 2011 年 12 月 16 日確認.

[6] 笹原 英司. 「価格.com ショック」不正アクセスの詳細. http://itpro.nikkeibp.co.jp/article/SMBIT/20050830/220309/, 2005. 2011 年 12 月 16 日確認.

[7] ITmedia エンタープライズ. Citigroup、顧客情報 390 万人分を紛失. http://www.itmedia.co.jp/enterprise/articles/0506/07/news031.html, 2005. 2011 年 12 月 16 日確認.

[8] ITmedia エンタープライズ. MasterCard のクレジットカード情報が流出. http://www.itmedia.co.jp/enterprise/articles/0506/18/news008.html, 2005. 2011 年 12 月 16 日確認.

[9] INTERNET Watch. ソフトバンク BB、450 万件超える Yahoo! BB の個人情報漏洩を認め謝罪. http://internet.watch.impress.co.jp/cda/news/2004/02/27/2252.html, 2004. 2011 年 12 月 16 日確認.

[10] Security NEXT. 富士ゼロックスシステムサービスから戸籍データを持ち出した協力会社社員が逮捕. http://www.security-next.com/004618, 2006. 2011 年 12 月 16 日確認.

[11] 岡田有花.「DION」400 万人分のユーザー情報流出　KDDI 小野寺社長が謝罪. http://www.itmedia.co.jp/news/articles/0606/13/news065.html, 2006. 2011 年 12 月 16 日確認.

[12] 保険のニュース 保険市場タイムズ. アフラック（アメリカンファミリー生命保険）の個人情報流出&不正販売！http://www.hokende.com/news/news_0efA0SyAc.html, 2011. 2011 年 12 月 16 日確認.

[13] SOUND HOUSE. 不正アクセスに伴うお客様情報流出に関するお詫びとお知らせ. http://www.soundhouse.co.jp/shop/News.asp?NewsNo=1561, 2008. 2011 年 12 月 16 日確認.

[14] 株式会社セガ. アルバイト応募者の個人情報の流出に関するお詫びとお知らせ. http://sega.jp/topics/081104_2/, 2008. 2011 年 12 月 16 日確認.

[15] ITmedia ニュース.「777TOWN.net」から 173 万人分の ID・パスワード・メールアドレス流出. http://www.itmedia.co.jp/news/articles/1011/15/news079.html, 2010. 2011 年 12 月 16 日確認.

[16] 日本ネット経済新聞. ユニットコム 不正アクセス被害で個人情報 25 万人が漏えい. http://web-tan.forum.impressrd.jp/e/2010/10/07/8985, 2010. 2011 年 12 月 16 日確認.

[17] SQUAREENIX HOLDING. 子会社における個人情報漏洩について. http://www.square-enix.com/jpn/news/2011/html/b5a4197d73f8d0220bab95b9bb021a4a.html, 2011. 2011 年 12 月 16 日確認.

[18] atmc tokyo. 三菱重工業へのサイバー攻撃【中国・香港・米国・インド等のサイトに情報送信か】. http://atmc-tokyo.com/international_affairs/11929/, 2011. 2011 年 12 月 16 日確認.

[19] http://www.snort.org/.

[20] 藤田直行. 侵入検知に関する誤検知低減の研究動向. **電子情報通信学会論文誌**, 2006.

[21] 沢田篤史，高倉弘喜，岡部寿男. 開放型大規模ネットワークのための IDS ログ監視支援システム. **情報学論**, 44:1861–1871, Aug 2003.

[22] http://gost.isi.edu/cidf/.

[23] http://www.tripwire.co.jp/.

[24] Nelson Murilo and Klaus Steding-Jessen.

[25] C. M. Linn, M. Rajagopalan, S. Baker, C. Collberg, S. K. Debray, and J. H. Hartman. Protecting against unexpected system calls. *SSYM'05 Proceedings of the 14th conference on USENIX Security Symposium - Volume 14*, 2005.

[26] George W. Dunlap, Samuel T. King, Sukru Cinar, Murtaza A. Basrai, and Peter M. Chen. ReVirt: Enabling intrusion analysis through virtual-machine logging and replay, 2002.

[27] `http://bro-ids.org`.

[28] Ke Wang and Salvatore J. Stolfo. Anomalous payload-based network intrusion detection. *Lecture Notes in Computer Science*, 2004.

[29] Wang Xinran and Pan Chi-Chun and Liu Peng and Zhu Sencun. SigFree: A Signature-Free Buffer Overflow Attack Blockern, 2010.

[30] Michalis Polychronakis, Kostas G. Anagnostakis, and Evangelos P. Markatos. Network-level polymorphic shellcode detection using emulation. *Computer Virology*, pages 257–274, 2006.

[31] Robin Sommer and Vern Paxson. Enhancing byte-level. network intrusion detection signatures with context. *CCS '03 Proceedings of the 10th ACM conference on Computer and communications security*, 14, 2003.

[32] Giorgos Vasiliadis, Spiros Antonatos, Michalis Polychronakis, Evangelos P. Markatos, and Sotiris Ioannidis. Gnort: High Performance Network Intrusion. *Lecture Notes in Computer Science*, 2008.

[33] Greg Williams. Honeycomb Creating Intrusion Detection Signatures Using Honeypots, 2011.

[34] Michalis Polychronakis, Kostas G. Anagnostakis, and Evangelos P. Markatos. Network-level polymorphic shellcode detection using emulation. *Computer Virology*, pages 257–274, 2006.

[35] E.Rescorla. HTTP over TLS. *Internet RFC 2818*, 5 2000.

[36] Chad W. Engelgau and Sanjeet Singh. *Overview of SSL Acceleration Implementations*. Dell Power Solutions, 3 2002. `http://www.dell.com/content/topics/global.aspx/power/en/ps1q02_ssl`.

[37] 出口 雄一. *IDS* と *IPS*, 9 2006. `http://itpro.nikkeibp.co.jp/article/COLUMN/20060830/246798/`.

[38] 独立行政法人情報処理推進機構 (IPA). *Web Application Firewall* **読本**, 10 2010. `http://www.ipa.go.jp/security/vuln/documents/waf.pdf`.

[39] 上原 孝之. **情報セキュリティスペシャリスト**, pages 311–316. 株式会社 翔泳社, 10 2011.

[40] Tony Bradley. Tony bradley. `http://netsecurity.about.com/od/newsandeditorial1/a/aazeroday.htm`.

[41] Carnegie Mellon University 's Computer Emergency Response Team (CERT). Denial of service attacks. `http://www.cert.org/tech_tips/denial_of_service.html`, 2001.

[42] Maciej Ogorkiewicz and Piotr Frej. *Analysis of Buffer Overflow Attacks*. Windows OS Security, 7 2004. `http://www.windowsecurity.com/articles/Analysis_of_Buffer_Overflow_Attacks.html`.

[43] Balasubramaniyan and Garcia-Fernandez and Isacoff and Spafford and Zamboni. An architecture for intrusion detection using autonomous agents. *IEEE*, 1998.

[44] James P. Anderson. Computer Security Threat Monitoring and Surveillance, 1980.

[45] An Intrusion-Detection Model, 1987.

[46] 鑪 講平, 田端 利宏, and 櫻井 幸一. 能動的な制御フローの変更による異常検知手法の提案. **情報処理学会研究報告**, 2005.

[47] 長野 文昭, 鑪 講平, 田端 利宏, and 櫻井 幸一. データ改ざん検出によるバッファオーバフロー検知システムの提案. *CSEC*, 2005.

[48] 安藤類央 and 武藤佳恭. 改良例外ハンドラを用いた実時間オーバーフロー防御システム. **情報処理学会 コンピュータセキュリティ研究会** *CSS2004*, 2004.

[49] 大山恵弘, 王維, and 加藤和彦. 静的解析に基づく侵入検知システムの最適化. **情報処理学会**, 2005.

[50] Sumeet Singh and Cristian Estan and George Varghese and Stefan Savage. The EarlyBird System for Real-time Detection of Unknown Worms. *ACM SIGCOMM Computer Communication Review*, 2004.

[51] Christian Kreibich and Jon Crowcroft. Honeycomb: creating intrusion detection signatures using honeypots. *ACM SIGCOMM Computer Communication Review*, 2004.

[52] Matthew. V.Mahoney and Philip K.Chan. P. K.: Learning nonstationary models of normal network traffic for detecting novel attacks. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[53] Malware Domain List. `http://www.malwaredomainlist.com/`.

[54] 榊 辰哉, 水谷 正慶, 武田圭史, and 村井純. メタデータ作成によるマルウェア通信解析手法の提案. **マルウェア対策研究人材育成ワークショップ**, 2009.

[55] Snort のルールの作成. http://jem.es.land.to/security/snortrule.html.

[56] 日本ネットワークインフォメーションセンター. http://www.nic.ad.jp/ja/newsletter/No27/100.html.