卒業論文　　2005年度 (平成17年度)

NIDS運用の効率化に関する研究

慶應義塾大学 環境情報学部

氏名：水谷　正慶

指導教員

慶應義塾大学 環境情報学部

村井 純

徳田 英幸

楠本 博之

中村 修

高汐 一紀

湧川 隆次

平成18 年1 月21 日

卒業論文要旨 2005 年度 (平成 17 年度)

NIDS 運用の効率化に関する研究

論文要旨

Network-based Intrusion Detectoin System(NIDS) とは管理ネットワークのトラフィックを取得し，セキュリティイベントを検知するシステムである．NIDS 運用の目的はリスクの高いインシデントの早期検出とインシデント発見後の調査の 2 点である．しかし NIDS 運用には 2 つの課題があり，効果的な運用が難しくなっている．

1 つめの課題は，NIDS が検知するセキュリティイベントの多くはそのリスクを評価することが難しい点である．リスクが高いと考えられるセキュリティイベントの発見が，実際に被害をおよぼすインシデントの発見へと繋がる．しかし，既存の NIDS が出力するセキュリティイベントには，リスク評価のための十分な情報が付与されていない．2 つめの課題は，複数セキュリティイベントの発生傾向や一連の全体像を把握するのが困難な点である．NIDS が検知するセキュリティイベントの数は膨大であり，全てのセキュリティイベントをネットワーク管理者が 1 つずつ解釈し全体像を把握するのは不可能である．

このような運用上の課題を解決するために，本論文では新しい NIDS の構成モデルを提案し，以下のシステムを構築した．不正侵入の試みや調査行為だけではなくその応答も含めて監視することによって，セキュリティイベントのリスクを判断できる **Session based NIDS**，内部ネットワークのホストが利用している OS 情報を管理し，セキュリティイベントの種類と比較することで影響を及ぼす可能性があるセキュリティイベントだけを抽出する **OS based Risk Evaluation System**，膨大な数のログを高速で視覚化し，セキュリティイベントの発生頻度や発生パターンの認識を容易にする **Security Event Log Visualizer**．本論文では，この 3 つのシステムを設計，実装し，それぞれを評価した．さらに，この 3 つのシステムを提案したモデルにしたがって導入し，NIDS 運用の効率化を実現した．

キーワード
1．インターネット，2．セキュリティ，3．ネットワーク侵入検知

慶應義塾大学 環境情報学部

水谷 正慶

Abstract of Bachelor's Thesis

Academic Year 2005

A Research of Efficient NIDS Operations

Network-based Intrusion Detection Systems (NIDSs) detect security events from their administrative domain network traffic. Operation of NIDS has two purposes; 1) immediate detection of high risk incidents, and 2) investigation of detected incidents. However, NIDS has two operational issues, which prohibit NIDS to operate effectively.

First issue in NIDS operation is that it is difficult to evaluate the level of risks of a security event detected by the system. Detection of high risk security events is important, because it leads to discovery of an incident that causes actual damages. However, security event information produced by traditional NIDS is not sufficient enough to evaluate the level of the risks of a security event. Second issue is that it is difficult to understand the trends of security events or the summaries of security events at large. The number of security events detected by NIDS is huge, so it is impossible for network administrators to investigate the security events one by one to understand the whole picture.

In order to solve these NIDS operational issues, a new construction model is proposed in this paper. Three systems; 1) Session based NIDS, 2) an OS based Risk Evaluation System, and 3) Security Event Log Visualizer, are also developed based on the model.

The Session based NIDS monitors not only attack traffics and probes but also corresponding responses to measure their results if they were a success or a failure. The OS based Risk Evaluation System evaluates the level of risks of a security event with consideration of host's Operating System. The Security Event Log Visualizer aggregates huge logs of security events, which were collected by the system, and visualizes them as summarized information.

This paper shows the design, implementation and evaluation of these three systems. These systems have been introduced in the running network along with proposed construction model, and have realized the efficient NIDS operation.

Keywords :
1. Internet, 2. Security, 3. Network Intrusion Detection

Keio University , Faculty of Environmental Information

Masayoshi Mizutani

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景

インターネットに接続している計算機（以下ホスト）は，常に様々な脅威にさらされている．2003年1月に発生したSlammerワーム [1] や2003年8月に発生したMS Blasterワーム [2] は，インターネットを通じて急速に世界中のホストに感染し，各所で業務に支障をきたすなどの被害をもたらした．

独立行政法人 情報処理推進機構 (IPA) の試算によれば2003年1月から12月までの国内でのコンピュータウィルスによる被害総額は3025億円であったという [3]．

ワームの他にも，ホストがさらされている脅威は少なくない．米Avantgarde社は実験により，欠陥を修復していないMicrosoft Windows[4] XP SP1を使用しているホストがインターネットに接続した場合，約4分で不正侵入されてしまうと発表している [5]．

このように，インターネットを安全に利用するためにはインターネット上に様々な脅威があるという事実を認識し，その脅威による被害をなくすための対策を講じる必要がある．

1.2　リスク，セキュリティイベント，インシデント

本論文では，脅威によってホストになんらかの被害が発生する可能性を**リスク**と呼ぶ．ISO17799[6] では安全性の基準として機密性，完全性，可用性の3つが定められている．本論文ではこの3つの侵害に組織的な信頼の侵害をあわせた，4つによって生じる損害を，脅威による被害と定義する．

機密性の侵害　機密性とは，特定の情報へのアクセスが許可された者だけに限定されていることである．機密性の侵害の例として，不正な手段によってシステムの権限を取得される，ソフトウェアの不適切な設定により意図しない相手に情報が漏洩するなどが挙げられる．

完全性の侵害　完全性とは，提供される情報やその処理過程の正確さが保障されていることである．完全性の侵害とは，ソフトウェアが改ざんされ正常に動作しなくなる，ファイルに記録されている情報が改ざんされる，などが挙げられる．

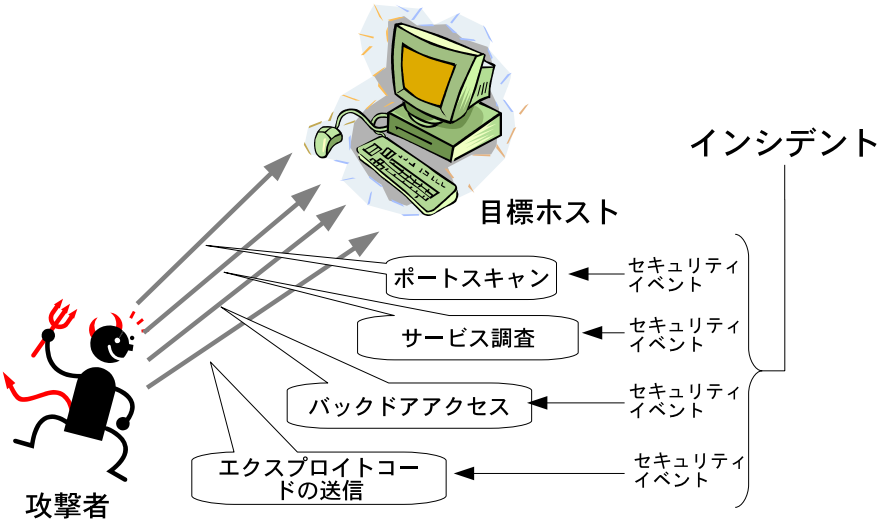

図 1.1: インシデントとセキュリティイベントの概念図

可用性の侵害　可用性とは，システムがダウンすることなく継続的に稼働していることである．可用性の侵害とは，ホスト内のプロセスやネットワークに異常が発生し，サービスが提供できなくなる状態である．

組織的信頼の侵害　法律などで禁止されている行為によって，組織の社会的な信頼を損なう可能性がある．内部ネットワークのホストから外部ネットワークにある別組織のホストに不正侵入を試みるなどの不正な行為や，違法な情報の取得，あるいは発信がこれにあたる．これは組織内部のユーザによって意図的に行われる場合と，不正侵入をうけることによって意図せずに行われている場合がある．

本論文では，具体的な被害の有無に関わらず，ネットワークを通じて意図的に機密性，完全性，可用性，組織的な信頼の侵害を試みる行為を，総称して**インシデント**と呼ぶ．インシデントは 1 つ，あるいは複数の事象から構成されている．このインシデントを構成する事象を**セキュリティイベント**と呼ぶ．セキュリティイベントの例としてホストが利用するソフトウェアの脆弱性調査や，不正侵入を試みる規定外のデータ送信，不正なプログラムによる外部との通信が挙げられる．

図 1.1はインシデントとセキュリティイベントの概念を示している．図 1.1のようにポートスキャン，サービス調査，エクスプロイトコードの送信，そしてバックドアへのアクセスは，それぞれセキュリティイベントとして扱われる．そして，目標ホストの機密性，完全性，可用性，組織的な信頼の侵害を目的とした，一連のセキュリティイベントで構成される行為がインシデントとなる．

1.3 インシデント発見と対応の必要性

インターネット上の脅威による被害を最小限にするためには，インシデントの事前対策，インシデントの発見，そして発生後の対応がそれぞれ必要である．インシデントの事前対策とは，特定の通信を遮断するファイヤーウォールやコンピュータウィルスを駆除するためのアンチウィルス製品の導入のような，インシデントの発生を防ぐための対策である．インシデントの発見とは，内部ネットワークで実際に被害をおよぼすインシデントの検出である．そして，発見後に被害規模の調査，対応戦略の決定，証拠の保全，被害の復旧まで行うのが発見後の対応（インシデントレスポンス [7] とも呼ばれる）である．インターネットにおける脅威は多岐にわたり，その全てを完全に防ぐのは難しい．暗号技術の権威である Bruce Schneier 氏も “Security is a process, not a product.” （セキュリティはプロセスであり，プロダクトではない）と述べている [8]．つまり，被害が発生する事を前提とし，被害の拡大を防ぐためのインシデントの発見，発生後の対応が重要となる．

例えば，ネットワーク管理者が特定の通信を遮断するファイヤーウォールや特定の通信内容を遮断する侵入防止システム (IPS) を導入したとしても，ホストのユーザが悪意のあるプログラムをダウンロードしてしまう可能性がある．また，既知の攻撃であっても，暗号化されている通信であればそれが攻撃であると判断するのが難しい．アンチウィルスやマルウェア対策のソフトウェアを利用すれば，ある程度は，悪意のあるプログラムを排除できる．しかし，このようなソフトウェアは，悪意のあるプログラムに含まれる固有のパターンをもとに検知，削除しているため，未知のウィルスについては有効ではない．さらに，ユーザ側で十分に注意したとしても，オペレーティングシステム (OS) をインストールした直後のホストは，その欠陥の修復パッチが適用されていないため，ネットワークの管理者側がファイヤーウォールや IPS による防御をしていなければ，ユーザが対策を施している最中に不正侵入される恐れがある．

被害をおよぼすインシデントは，発生から対応までの時間が長いほど被害が拡大しやすい [9]．そのため，被害をおよぼすインシデントを早期に発見・対応できれば，被害を最小限にとどめることができる．

このように，インシデントによる被害はあらゆるネットワークで発生する可能性がある．被害を最小限に防ぐためには事前対策だけではなく，インシデントの発見，事後対策を視野に入れたネットワーク運用を行わなければならない．

1.4 本論文の目的

前節で述べた通り，ネットワークに接続しているホスト，あるいはネットワークの管理者は何らかの手段でインシデントによる被害に備えなければならない．本研究では，インシデントによる被害発生の通知，およびインシデントの解析に有効な手段としてネットワーク型侵入検知システム（Network-based Intrusion Detection System，NIDS）に着目する．NIDS は，ネットワーク上に流れるトラフィックを監視し，インシデントであると考えられるセキュリティイベントを発見するシステムである．セキュリティイ

ベントを発見した後は，ファイルやデータベースへの記録，メールによるネットワーク管理者への通知のような予め定められた動作をする．

NIDS は，ネットワーク上でのセキュリティイベントの発見において有用なシステムであるが，実際の運用では，セキュリティイベントが大量に検出されるため，重要なインシデントを見逃してしまう，あるいは必要なセキュリティイベントの抽出が困難であるという問題がある．そのため，一般的に NIDS は運用が難しく，効率の悪いシステムとなっている．本論文における効率とは，作業負担に対して得られる結果を指す．少ない作業負担で多くの結果を得られるシステムを，効率の良いシステムと呼ぶ．

本論文では NIDS がもつ既存の問題点を明らかにし，これを改善するための手法として Session-based NIDS，OS 種別を基本としたリスク評価システム，セキュリティイベント情報の可視化という 3 つの手法を提案する．これによって，NIDS 運用の効率を改善する．

1.5 本論文の構成

第 2章において，NIDS の運用モデルを定義し，第 3章において運用における問題点を述べる．第 4章でそれぞれの問題点を解決するための手法として Session based NIDS，Target based NIDS，Log Visualization について論じる．第 5章，第 6章，第 7章でそれぞれの概要，設計，実装，および評価を述べる．そして，第 8章では各実装の連携について，その設計と実現方法を述べる．最後に第 9章で本研究によって改善された NIDS 運用の効率を評価し，結論を述べる．

第2章　NIDS

本章では本論文であつかう NIDS の特徴，運用の実態について述べる．

2.1　NIDS とは

ユーザが利用するホストとは別に，内部ネットワークに設置するインシデントの事前対策，発見，事後対応を目的としたシステムは，セキュリティデバイスと呼ばれている．例として，ファイヤーウォールや IPS が挙げられる．NIDS は，ネットワークを流れるインターネット通信（トラフィック）を監視することで，セキュリティイベントを発見するセキュリティデバイスの一種である．

初期の NIDS は 1990 年に発表された “A Network Security Monitor” [10] で述べられているシステムである．その後，1994 年に最初の商用 NIDS である NetRanger[11] が登場し，1998 年には最も有名なオープンソース NIDS の 1 つである Snort[12] が公開された．

2.2　NIDS 運用の目的

NIDS 運用の目的として**リスクの高いインシデントの早期検出**と，**インシデント発見後の調査**の 2 点が挙げられる．

リスクの高いインシデントの早期検出　NIDS はトラフィックを監視，分析することでセキュリティイベントを検知する．ネットワーク管理者は調査，もしくはプログラムによる自動判別によって，検知したセキュリティイベントの中から特に被害をおよぼす可能性が高いセキュリティイベントを発見する．その結果，深刻なインシデントが発生していれば，ネットワーク管理者は迅速に対処しなければならない．インシデントによる被害は時間と共に拡大する傾向があるため，インシデントの早期検出は，被害を最小限に抑えるための重要な要件である．

インシデント発見後の調査　ネットワーク管理者は，インシデント発見後，NIDS が記録したセキュリティイベントの中から，インシデントを構成するセキュリティイベントを抽出する．そして，抽出したセキュリティイベントを分析することで，ネットワーク管理者はインシデントがいつ，どのような要因で発生したか，といった手がかりを探し出すことができる．これは，現実世界における監視カメラの役割に相当する．ま

た，内部への被害の波及や，別のネットワークに対する不正侵入の試みが発生している可能性もある．このような調査も NIDS によって得られたセキュリティイベントの記録を用いることで可能となる．

2.3 NIDSのトラフィック取得手法

監視のためにトラフィックを取得する手法は，NIDS によってパッシブ型とインライン型の 2 種類に分類される．それぞれ，図 2.1と図 2.2に NIDS の構成例を示す．

パッシブ型 パッシブ型は，管理ネットワーク内のある点において複製（mirroring）したトラフィックを取得する．この手法は複製したトラフィックを利用するため，ネットワークに影響を与えにくい．しかし，短時間に大量のトラフィックを渡された場合，NIDS の性能によっては分析や記録の処理が間に合わずに，トラフィックを取りこぼしてしまう．

インライン型 インライン型は，図 2.2のように，ネットワークの gateway と内部ネットワーク (Internal Network) に設置する．これは，一方から流れてきたトラフィックを分析した後に，もう一方へとトラフィックを流す手法である．インライン型は分析したトラフィックのみを通過させるため，取りこぼしは起きない．しかし，NIDS が正常に機能しなくなった場合には，ネットワークに影響を及ぼす可能性がある．

NIDS はパッシブ型とインライン型のいずれか，あるいは両方の実装がある．NIDS が確実にトラフィックを検査するためにはインライン型によるトラフィック取得が望ましいが，NIDS の性能が低ければネットワークの通信速度や応答速度が低下する可能性がある．また，NIDS の故障やネットワーク管理者の設定ミスにより，ネットワークが不通になる可能性もある．そのため，多くの運用環境ではパッシブ型のトラフィック取得手法で運用されている．

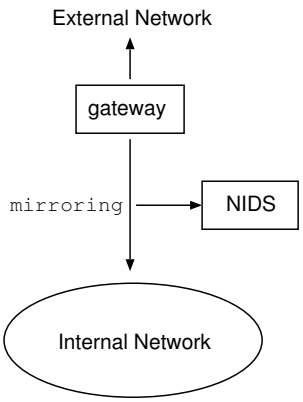

図 2.1: パッシブ型 NIDS の構成例

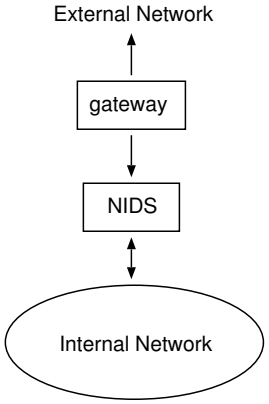

図 2.2: インライン型 NIDS の構成例

2.4 NIDSのセキュリティイベント検知手法

NIDSがセキュリティイベントを検知する手法は，ミスユース型検知手法とアノーマリ型検知手法に分類できる．

ミスユース型検知 ミスユース型検知とは，あらかじめ定められたトラフィックを発見する検知手法である．まず，セキュリティイベントであると考えられるトラフィックの特徴を示した**シグネチャ**を用意する．シグネチャには送信元および送信先IPアドレス，送信元および送信先ポート番号，トラフィックに含まれる文字列のパターン，その他IPヘッダやTCPヘッダの各項目などを，1つもしくは複数指定できる．このシグネチャと監視しているトラフィックの特徴を比較し，一致すればNIDSはセキュリティイベントを発見したものとする．あらかじめ定められた特徴によってセキュリティイベントを検知するため，検知結果は概ね正確だが，未知のセキュリティイベントを検出しにくいという特性がある．現在はミスユース型検知を採用しているNIDS実装が多い．実装の例としてはSnort[13]やPrelude[14]が挙げられる．

アノーマリ型検知 アノーマリ型検知とは，瞬間的なトラフィック流量や，送信先IPアドレスの数，送信先ポートの分布状態のような，トラフィックの傾向にそれぞれ閾値を設定し，その閾値を超えるトラフィックが発生した場合にセキュリティイベントが発生したと見なす手法である．閾値を適切に設定できれば，管理ネットワーク上のトラフィック異常を発見できるため，未知のセキュリティイベントも検知できる場合がある．ただし，閾値が適切でなければ誤検知が多発してしまう．したがって，ネットワーク管理者は当該ネットワークのトラフィック傾向を十分に理解しなければならず，ミスユース型検知に比べて運用が難しい．

アノーマリ型はネットワークの正常状態を定義し，それにもとづいて閾値を設定しなければならない．正常状態はネットワークによって様々であり，各ネットワークの管理者は自ら正常状態や閾値を判断しなければならない．これに対してミスユース型検知で用いるシグネチャは，他のネットワーク管理者が作成したシグネチャでも転用できるものが多い．なぜならば，コンピュータウィルスやプログラムを用いた攻撃によって生じるトラフィックの特徴は，ネットワークによって変化しにくいためである．以上のような理由から，本論文執筆時点ではミスユース型検知手法が主流となっており，ほとんどのNIDS実装は主な検知機構としてミスユース型を採用している．

2.5 NIDS運用の流れ

本論文ではNIDSの運用を監視，解析，調整の3つのフェイズに分けて説明する．各フェイズの遷移を図2.3によって示す．

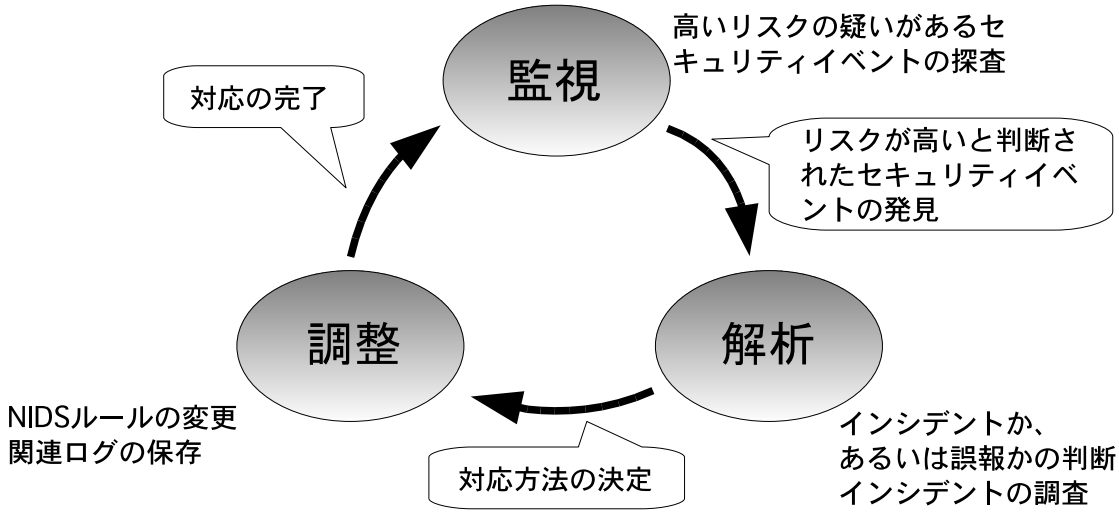

図 2.3: NIDS の運用におけるフェイズ遷移

監視フェイズ NIDS が検知するセキュリティイベントは深刻な被害をもたらすものばかりではなく，被害をもたらさないセキュリティイベントも含まれる．さらに，悪意のないトラフィックをセキュリティイベントとして検知するフォールスポジティブ（誤検知）を起こす可能性もある．NIDS はこのようなセキュリティイベントを多く検知するため，その全て解析するのは効率が悪い．したがって，監視フェイズでは NIDS が検知したセキュリティイベントをいくつかの手法によってリスク評価し，リスクが低いと判断されるセキュリティイベントを除外する．

解析フェイズ 解析フェイズでは，適切なインシデント対応をするためにセキュリティイベントを解析する．まず，監視フェイズで除外されなかったセキュリティイベントが，インシデントの一部であるか否かを判断する．インシデントであれば，関連する他のセキュリティイベントを抽出し，インシデントによる被害状況，インシデントの発生日時を調査する．調査結果は，インシデント対応方法の検討に利用される．あるいは，被害が発生しないインシデントであった場合は，当該セキュリティイベントがリスクがある，あるいはリスク不明と判断された理由について調査する．

調整フェイズ 調整フェイズでは，解析フェイズで得られた情報をもとに，今後適切に NIDS を運用するための調整をする．具体的には NIDS のシグネチャの変更や，あるセキュリティイベントを発見した場合の動作設定変更が挙げられる．被害をもたらすインシデントであった場合は，より詳細な情報を記録するように変更する．被害が発生しないインシデントであれば，誤検知回避のためにシグネチャを編集する，あるいは不要なシグネチャを無効化することで，NIDS を調整する．また，被害を発生させたインシデントに関して再び調査する可能性があれば，必要なセキュリティイベントログを別途保存しておく．

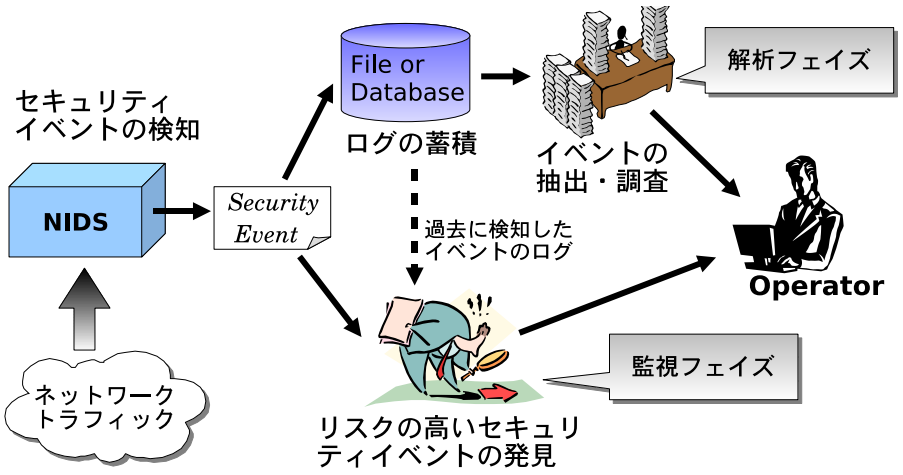

図 2.4: NIDSが検知したセキュリティイベントの流れ

各フェイズはそれぞれNIDSの運用に必要だが，被害をおよぼす可能性のあるインシデントの発見とインシデント発見後の調査という2つの目的を達成するためには，監視フェイズ，解析フェイズの迅速な処理が特に重要となる．監視フェイズ，解析フェイズにおいて，NIDSが検知したセキュリティイベントがどのように利用されるかについては，第2.6節で詳細に述べる．

2.6 NIDSが検知したセキュリティイベントの利用方法

NIDSが検知したセキュリティイベントは，主に監視フェイズと解析フェイズで用いられる．それぞれの用途に応じてセキュリティイベントは異なる流れで処理される．

2.6.1 監視フェイズにおけるセキュリティイベントのリスク評価

監視フェイズではNIDSによって得られたセキュリティイベントのリスクを評価し，リスクが高いインシデントの発見を支援する．セキュリティイベントのリスク評価手法を表2.1にまとめる．リスクの高いセキュリティイベントの発見手法は大きく分けて２種類となり，それぞれ自動的に処理できるものと，人間が自ら作業しなければならないものがある．リスクが高いセキュリティイベントの発見は表2.1のいずれか，あるいは複数を用いて行われる．

単独のセキュリティイベントに付随する情報にもとづいた発見　単独のセキュリティイベントに付随する情報にもとづいた発見とは，送信元IPアドレスや送信先IPアドレス，セキュリティイベントの種類のようなセキュリティイベントがもつ各種パラメータを利用してリスクを判別する手法である．1つ，あるいは複数のパラメータだけでリ

表 2.1: リスクが高いインシデント発見手法の分類

手法		例
単独のセキュリティイベントに付随する情報にもとづいた発見	自動	セキュリティイベント種別，送信元情報に基づいた条件判断
	手動	各ホストへ直接ログインしての実地調査，NIDS以外の監視装置からの情報による調査
セキュリティイベント発生状況の異常にもとづいた発見	自動	セキュリティイベント発生頻度の閾値を予め定める，比率分析，稀率分析
	手動	過去のセキュリティイベント発生状況との体感的な比較

スクを判別できる場合は，あらかじめその組み合わせを定義することによって，リスク判別を自動化できる．各パラメータだけではリスクを判別できないセキュリティイベントは，ネットワーク管理者が関連するIPアドレスやセキュリティイベントの発生時刻，使用していたプロトコル，トラフィックのペイロード情報を利用し，調査する．具体的には当該ホストへログインしての実地調査，あるいは別のセキュリティデバイスによって得られた情報の参照によってリスク判別されるが，これらの調査は管理者の負担が大きくなりがちである．そのため，リスクを判別できないセキュリティイベントを無視する方針で運用されているネットワークも多く存在する．

セキュリティイベント発生状況の異常にもとづいた発見 セキュリティイベント発生状況の異常にもとづいた発見とは過去のセキュリティイベント発生状況と現在の発生状況を比較し，その差異からリスクが高いと考えられるセキュリティイベントを発見する手法である．セキュリティイベントの発生状況とは，セキュリティイベントの発生頻度や発生パターンを指す．自動的にインシデントを発見する場合は，過去のセキュリティイベント情報を利用した比率分析や稀率分析を利用し，通常観測されにくいセキュリティイベントの発生や発生頻度の急増などを検知する手法[15]や，セキュリティイベントの種類毎に発生頻度の閾値を設定し，それを越えた場合にリスクの高いセキュリティイベントとみなす手法が用いられる．ネットワーク管理者が手動で過去と現在のセキュリティイベント発生状況を比較するためには，過去の発生状況を包括的に，かつ正確に把握する必要がある．これは，過去のセキュリティイベント数が多くなるに従い，作業負担が増加する．

2.6.2 解析フェイズにおけるインシデントの調査・分析

解析フェイズではログとして蓄積されているセキュリティイベントを発生期間やトラフィックの送信元ホスト，宛先ホストなどを指定した条件によって抽出し，インシデントの全容把握に利用する．調査，分析において重要なのはインシデント全体像の把握である．多くのインシデントは，複数のセキュリティイベントによって構成されている．インシデントを調査するには，関連すると考えられるセキュリティイベントのログを抽出し，以下に挙げる 3 点を調査する．

インシデントの有無の調査 最初にすべき調査は，リスクが高い，あるいはリスクが不明なセキュリティイベントが，インシデントの一部であるかの確認である．具体的には，検知されたセキュリティイベントのトラフィックの精査や関連する可能性のあるセキュリティイベントを抽出し，インシデントとしての傾向が見られるかの確認といった手段が挙げられる．特にリスクが不明であるセキュリティイベントは誤検知の可能性もあり，実際にインシデントが発生していない場合も多い．

インシデントによる被害状況の調査 セキュリティイベントがインシデントの一部であることが確認された場合，インシデントによってどのような被害がもたらされているかを調査する．例えば，不正侵入の試みがあったとしても，標的とされている脆弱性が修復済みのソフトウェアを利用していれば被害は発生しない．このようなセキュリティイベントは一般的に “Non-Effective Event” と呼ばれている．逆に被害をおよぼす可能性があるセキュリティイベントは “Effective Event” と呼ばれている．被害状況の調査では，主当該ホストにどのようなセキュリティイベントが発生したか，他の内部ネットワークホストにインシデントが波及していないか，外部ネットワークのホストに不正侵入などを試みていないかを調べる．また，インシデントの進行状況の確認として，不正侵入者によるシステムへのアクセスや，不正プログラムの活動が継続中であるか否かの調査も重要である．

インシデント発生日時の調査 より明確な被害状況を把握するために，当該インシデントの発生日時を調査する．インシデントの発生時刻が明確化すれば，他ホストへの被害の切り分けもより確実なものとなる．

多くの NIDS の実装では，検知したセキュリティイベントをモニター画面に表示し，ログ情報として閲覧するための専用のアプリケーションが提供されている．このようなアプリケーションをユーザインターフェースと呼ぶ．ネットワーク管理者がインシデントの全体像を把握するためには，多くのセキュリティイベントを一括して認識できるユーザインターフェースが望ましい．しかし従来の NIDS では，専用の管理コンソールで表示できる情報の量は制限され，全容の把握が難しい．

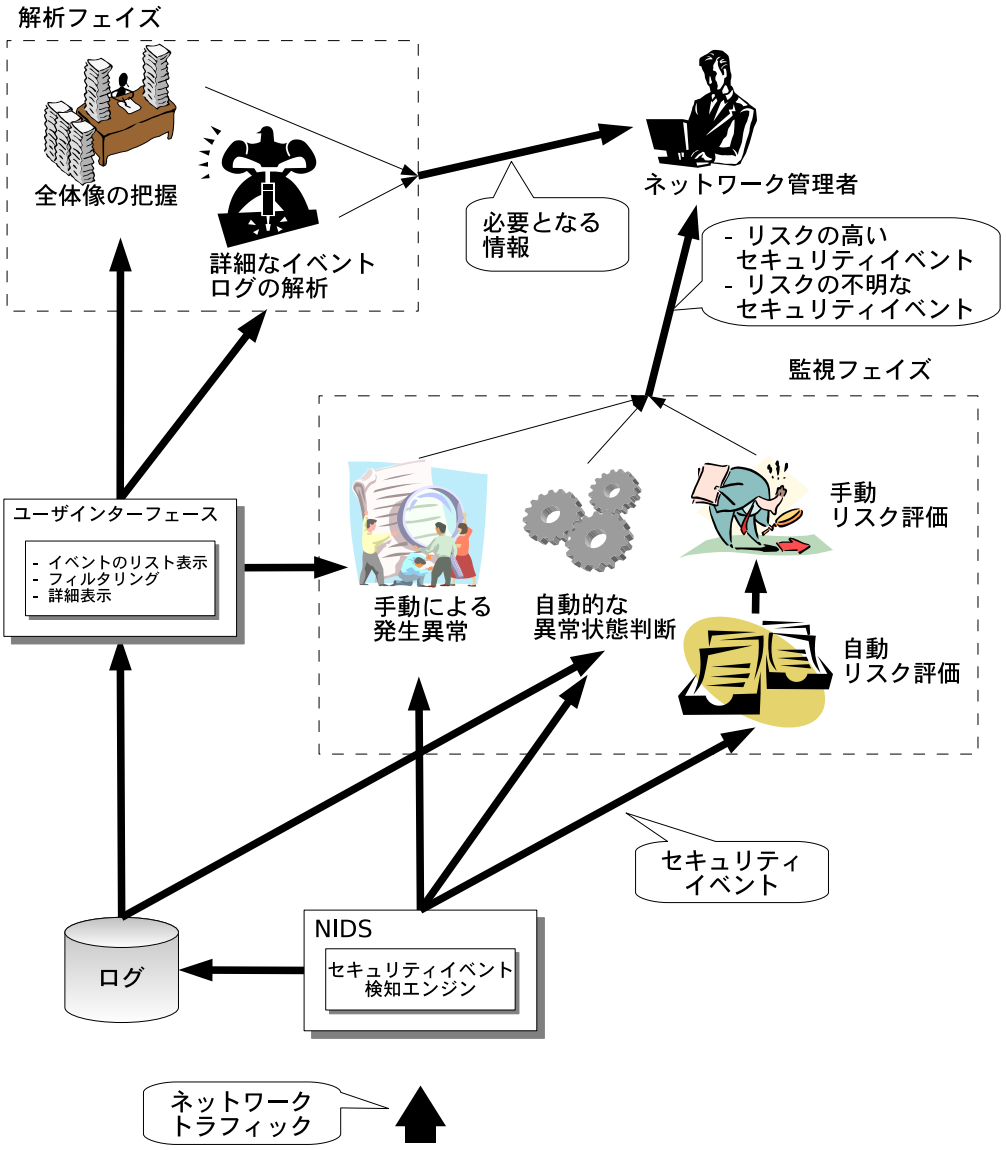

図 2.5: 従来の NIDS 運用構成

2.6.3 セキュリティイベント利用の流れ

これまでの説明を元に，NIDS で検知したセキュリティイベントのより詳細な利用の流れを図 2.5に示す．矢印は NIDS が検知したセキュリティイベントの流れを表している．検知したセキュリティイベントはログとしてファイル，あるいはデータベースに蓄積される．また，ログとしての蓄積とは別に，セキュリティイベント毎のリスク評価と発生状況の異常による高リスクなセキュリティイベントの発見に利用される．

ネットワーク管理者が自らセキュリティイベント発生状況の異常を検知するには過去のセキュリティイベントログをユーザインターフェースによって確認する．自動的にセキュリティイベント発生状況の異常を検知する場合は，判別用のソフトウェアな

どが直接過去のログをデータベースなどから引き出し，傾向を分析する．

セキュリティイベント毎のリスク評価では，まず自動判別のソフトウェアによってリスクの低いセキュリティイベントを除外する．除外されるリスクの度合いやリスクが不明なセキュリティイベントの扱いは，ネットワークの運用方針と，管理者がNIDS運用に費やせる作業量によって異なる．重要なネットワークであれば，ある程度リスクの低いセキュリティイベントやリスクが不明なセキュリティイベントについても解析フェイズで被害の確認などの作業を行うが，そうでないネットワークであればリスクが高いと判断されたセキュリティイベントのみを解析する．自動判別を経たセキュリティイベントは，さらにネットワーク管理者自身によってリスク評価され，不要なものを除外する．残ったセキュリティイベントは解析フェイズによってインシデントであるか等の判断をする．

解析フェイズでは蓄積されているセキュリティイベントのログを必要に応じて抽出する．抽出するための条件はセキュリティイベントの発生期間，送信元ホストや宛先ホストのIPアドレス，ポート番号，セキュリティイベントの種類などの1つ，あるいは複数となる．

2.7 NIDSで検知できるセキュリティイベントの種類

本論文では，ミスユース型検知のNIDSによって検知できるセキュリティイベントを以下のように分類する．

不正侵入の試み 不正な手段によって権限を取得する試みを検知したセキュリティイベント．具体的には，ソフトウェアが持つ脆弱性を利用して不正に認証手順を省略し権限を取得しようとする試みや，識別情報（パスワードなど）を，不正に取得しようとする行為が挙げられる．ソフトウェアのバッファオーバーフロー脆弱性を利用したエクスプロイト攻撃や，脆弱性を持つWebサービスへの攻撃，自律的に他のホストの脆弱性を利用して感染を拡大させるタイプのコンピュータウィルスが分類される．また，度重なる認証の失敗も，パスワードの総当たり攻撃（ブルートフォースアタック）として検知できる．

調査行為 内部ネットワークの構造や，各ホストに関する情報を収集する活動を検知したセキュリティイベント．外部から到達できる内部ネットワークのホストのIPアドレス，提供しているサービスの種類，使用しているOS，使用しているソフトウェアの種類やバージョンなどを様々な手法によって取得する．特定のツールの利用による調査はトラフィックに特徴的な部分があるため，検知が可能である．しかし，正常な通信に見せかけてスキャンする手法もあるため，全ての調査行為を正確に検知するのは困難である．また，複数トラフィックの監視によって，調査行為であると推測することもできる．例えば，ポートスキャンは短時間に大量のポートへアクセスするため，通常見られないトラフィックとして検知しやすい．

プロトコル異常 RFCなどで定められた通信プロトコルから，逸脱したトラフィックを検知したセキュリティイベント．具体的には特定プロトコル中での使用禁止文字の発見や，送信データのフォーマット異常，通信手順の異常などが挙げられる．例えばSimple Mail Trannsfer Protocol (SMTP)[16]やPost Office Protocol3 (POP3)[17]などのテキストベースのプロトコルでは，送信するコマンドと引数が全てASCII[18]の英数字および記号のみで構成されるものとして，Request For Comment (RFC)[19]で標準化されている．そのため，SMTPやPOP3の通信中に英数字や記号以外のデータが含まれていた場合，何らかの異常が発生していると判断し，セキュリティイベントとして検知する．プロトコル異常は未知の攻撃などを検知できる可能性が高い．しかし，プロトコルを正確に実装していないアプリケーションも少なくないため，リスクの無いトラフィックもセキュリティイベントとして多数検知されてしまう傾向がある．したがって，プロトコル異常を示すセキュリティイベントの多くは誤検知を前提としており，専ら状態異常による検知やインシデント発覚後の調査に利用される．

不審なトラフィック プロトコル異常ではないが，希にしか観測されないようなトラフィックを検知したセキュリティイベント．通常利用されない特殊なプロトコルや，異常にサイズが大きいパケットなどがこれに相当する．あるいは不正侵入のためのエクスプロイトコードに含まれる可能性が高いトラフィックパターンの検知も，このセキュリティイベントに分類できる．このような種類のトラフィックは悪意のあるものばかりではない．例えば，内部ホストのユーザが特殊なアプリケーションを使用していたり，データ転送中に偶然エクスプロイトコードに類似したトラフィックが流れる可能性もある．このセキュリティイベントもプロトコル異常のセキュリティイベントと同様，過剰検知を前提としており，主に状態異常による検知やインシデント発覚後の調査に利用される．

ポリシ違反 規定によって管理ネットワーク内で特定のアプリケーションの利用を禁止している組織がある．特に企業ネットワークにおいては業務に関連しないアプリケーションの利用を禁止している場合が多い．例としてチャットアプリケーション，ネットワーク対戦型ゲーム，P2Pアプリケーションなどが挙げられる．また，企業などの組織では，業務とは関連しないサイトの閲覧を禁じている．このようなアプリケーションの利用や，サイト閲覧の際に発生するトラフィックをNIDSで検知できる．

悪意のあるプログラムの活動 ユーザが意図せずインストールされたプログラムによって，情報を漏洩させる活動を検知したセキュリティイベント．具体的にはスパイウェアやマルウェアの活動が検知される．また，メールなどを経由して感染するコンピュータウィルスもこれに分類される．内部の管理ネットワーク内から外部に向かうトラフィックを検知することで，既に内部ネットワークで活動している悪意のあるプログラムを発見できる．「禁止されている行為」でも特定アプリケーションの利用を検知するが，「悪意のあるプログラムの活動」との違いは，ユーザが意図して利用しているか否かという点である．本論文における悪意あるプログラムとは，ユーザが意図して利用してい

ない，あるいは一部で意図しない動作をしているものを指す．

サービス妨害攻撃 大容量のトラフィックや大量のリクエストを送信することで，ネットワークインフラストラクチャの障害を引き起こしたり，アプリケーションを過負荷な状態にして，サービスの提供を阻害する攻撃を検知したセキュリティイベント．一般的に Denial of Service(DoS) 攻撃と呼ばれている．DoS の検知には DoS を目的として作成されたソフトウェアが送信するトラフィックの特徴や，通常より大きいサイズのパケットの発見によりセキュリティイベントとして検知する．

2.8 まとめ

本章では NIDS の特徴と運用実態について述べた．NIDS はネットワーク上のセキュリティイベント検出に有用なシステムである．NIDS は不正侵入の試み，調査行為，プロトコル異常など様々なセキュリティイベントが検知可能であり，検知されたセキュリティイベント情報は運用における監視，解析，調整の各フェイズで活用される．このような過程によってリスクの高いインシデント早期検出，そしてインシデント発見後の調査をするのが NIDS の運用目的である．

第3章　NIDSの運用における課題

本章ではミスユース型 NIDS の運用における課題を論じる．運用における課題は大きく分けて 2 つある．1 つは，セキュリティイベントに付随するパラメータのみでリスク評価できるセキュリティイベントが限定されており，多くのセキュリティイベントがリスク不明として扱われてしまう問題である．もう 1 つは，検知されるセキュリティイベントが膨大な数量となってしまうため，蓄積されたセキュリティイベントのログからインシデントの全体像や，ネットワークの全体的なセキュリティイベント発生状況の把握が困難となる点である．

3.1　セキュリティイベント毎のリスク評価における課題

セキュリティイベント毎に正確なリスク評価を行うためには，既存の NIDS から得られる情報に問題がある．以下でこの問題を説明する．

3.1.1　リスク評価が可能なセキュリティイベントの種類

第 2.7節で述べた各種セキュリティイベントのリスク評価可能性について表 3.1にまとめる．第 2.6節で述べたとおり，NIDS はセキュリティイベントの出力時に送信元 IP アドレス，送信先 IP アドレスなどのパラメータも併せて出力する．本論文におけるセキュリティイベントのリスク評価可能性とは，NIDS が出力するパラメータを手がかりとし，セキュリティイベントがどのような状況で発生したのか，あるいはセキュリティイベントの発生によってどのような状況が引き起こされたのかが推測可能であり，その結果からリスクを判断できる事を指す．例えば，NIDS が検知するセキュリティイベントは同じ種類のセキュリティイベントであっても，送信元によってその意味が異なる．そのため表 3.1では，検知されるセキュリティイベントの送信元が内部ネットワークか外部ネットワークかによって，さらに細分化している．表 3.1におけるリスクの評価可能性の欄は，NIDS が検知したセキュリティイベントに付随する情報のみでリスク評価できる場合は○，リスク評価が困難な場合は×，具体的なセキュリティイベント内容によって異なる場合は△としている．各セキュリティイベントのリスクは必ずしも表 3.1に従うものではなく，いくつか例外もある．

まず，リスク評価が可能であるとしたセキュリティイベントについて説明する．内部ネットワークより発信されたトラフィックから検知された不正侵入の試みや調査行為のセキュリティイベントは，トラフィックの宛先が外部ネットワークであれば結果の如

表 3.1: NIDSが検知できるセキュリティイベントとその評価可能性

セキュリティイベントの種類	送信元	リスクの評価可能性
不正侵入の試み	内部	○
	外部	×
調査行為	内部	○
	外部	△
プロトコル異常	内部	×
	外部	×
不審なトラフィック	内部	×
	外部	×
悪意のあるプログラムの活動	内部	○
	外部	×
ポリシ違反	内部	○
	外部	○
DoS攻撃	内部	○
	外部	○

何を問わずに組織の社会的信頼を損失する恐れがあり，早急な対処が必要となる．また，内部から外部への不正侵入の試みや調査行為，DoS攻撃は，既に内部ネットワークのホストが不正侵入を受けている可能性を示しており，リスクが高いと考えられる．悪意のあるプログラムの活動も検知した時点で外部に情報を漏洩しており，機密性が侵害されているとわかる．ポリシ違反のリスクは組織の規則によって様々だが，ポリシ違反がなされている事実は検知できるため，規則の方針が定まっていればリスクは評価できる．サービス妨害攻撃の検知もネットワークやアプリケーションが被害にさらされている状況を示しているため，リスク評価が可能である．

以上がリスク評価の容易なセキュリティイベントだが，表3.1の通り評価が困難なセキュリティイベントも多く存在する．リスク評価が困難なセキュリティイベントの詳細と，その根拠について以下で述べる．

不正侵入の試みのリスク評価　不正侵入の試みは，その結果が成功か失敗かによってリスクは大きく異なる．しかしその結果は常に同じではなく，各ホストの状況によって変化する．例えば，ある脆弱性を利用した不正侵入手法は，その脆弱性が未対策のホストに試行されれば成功する可能性が高く，セキュリティイベントのリスクは高いと言える．しかし，そのホストが対策済みであれば不正侵入は失敗するため，そのセキュリティイベントのリスクは低くなる．対策状況を確認できればリスクを判別できるが，各々のホストで脆弱性の対策状況は異なり対策するタイミングも異なる．セキュリティイベントが発生する度に各ホストへログインし，対策状況を調査するのは作業負担から考えても困難である．

また，ソフトウェア固有の脆弱性を利用する不正侵入の試みでは，攻撃対象が特定のアプリケーションや特定のOSでなければ，試みは失敗する可能性が高い．それにも関わらず，ツールを用いた不正侵入の試みは，ホストが利用しているソフトウェアと関係なく，全てのホストに行われる場合が多い．このような状況でもホストによってリスクは様々だが，既存のNIDSは不正侵入を試みられた，という事実のみをセキュリティイベントとして検知するため，付随する情報だけではリスクを評価できない．

ブルートフォースアタックも，成功か失敗かによってリスクが大きく異なる．ブルートフォースアタックの成功したか否かについては，別途アプリケーションのログなどを調査し判断しなければならない．しかし，不正侵入者はアプリケーションのログの改ざん，ルートキットなどによる侵入後活動の隠蔽を図る可能性があり，ホストを調査してもその痕跡を発見できない恐れがある．

調査行為のリスク評価　調査行為は各ホストで利用しているOSやアプリケーションの種類の特定，提供しているサービスの調査，サービスの設定などを調べるものであり，直接的な被害をもたらすセキュリティイベントはない．しかし，サービスの設定ミスや初期設定の放置によって，本来公開されるべきではない機密情報が漏洩していたり，意図していないサービスの提供が発覚する可能性のある調査行為もある．このような事実が攻撃者に発覚すれば，リスクの高いインシデントに発展する可能性が高い．このようなセキュリティイベントのリスクは各ネットワークの運用方針などによって異なるが，従来のNIDSから得られる情報では上述した状況の判断は難しい．これも，各ホストのサービス設定やソフトウェアの利用状態の調査によってリスクを評価できるが，不正侵入の試みと同様に設定などは動的に変化する可能性が高い．正確なリスク評価のためには，その都度調査する必要がある．

プロトコル異常のリスク評価　プロトコル異常はRFCなどの規定から逸脱している，本来起こるはずがないトラフィックを検知し，未知の攻撃やネットワークのトラブルなどを発見する．しかし，実際には規定されているプロトコルから逸脱した通信は数多く存在する．具体的に検知されたトラフィックが何を意味するのかは，各セキュリティイベントのデータ部分を精査しなければならないが，悪意のあるトラフィックやリスクの高いトラフィックの検知はごく稀であり，実運用において悪意のあるセキュリティイベントを抽出するのは難しい．

不審なトラフィックのリスク評価　不審なトラフィックもプロトコル異常のセキュリティイベントと同様，悪意のないトラフィックでも検知しやすいという欠点がある．

上述した通り従来のNIDSが検知するセキュリティイベントは，セキュリティイベントが持つ情報によってリスク評価できる種類が限られている．リスクが不明なセキュリティイベントも，ネットワーク管理者が実地調査や別のセキュリティデバイスが出力した記録との照合からリスクを判別できる場合があるが，現実的な運用方法ではない．第3.1.2項ではその実例を示す．

3.1.2 リスク評価不能なセキュリティイベントによるコスト増加の実例

付随する情報のみではリスクを評価できないセキュリティイベントも，実地調査などによってリスク評価ができる場合があると第3.1節で述べた．しかし，NIDSはセキュリティイベントのもつ情報だけではリスク評価できないセキュリティイベントを，日常的に多数検知している．

表3.2は2005年12月1日0時0分から，2005年12月2日0時0分までの24時間にあるWindowsホストに対して検知された，リスク評価が困難なセキュリティイベントの種類と検知数である．検知にはSnortを利用し，検知シグネチャはSnort標準で付属されるルール，http_decodeプリプロセッサ，win-rpc.rules[20]，BLEEDING-EDGE SNORT[21]を利用した．

このように，1つのホストに対して発生したセキュリティイベントについて，全て調査するだけでも大きなコストが必要となる．さらに管理ネットワークが中・大規模なネットワークであれば管理するホスト数に比例してセキュリティイベント数も多くなる．つまり，全てを正しく調査するには大変な負担を要する．

3.1.3 コスト増加にともなって生じる弊害

パラメータを用いたリスク評価ができるのは，限られたセキュリティイベントのみであると第3.1節で述べた．これはNIDS運用において2つの大きな弊害を引き起こしている．

まず1つは手動でリスクを評価するセキュリティイベント数が増加し，費用対効果が低下してしまう点である．第3.1.2項に示したように，リスクが不明なセキュリティイベントは，実ネットワーク環境において多数検知される．これらのセキュリティイベントを全て手動で調査するコストが高くなるのは明らかだが，調査によって得られる結果は多くない．なぜならば，インシデントの発生後対応[7]を適切に行っていれば，インシデントの発生によってより十分な事前対策を講じるようになるため，被害をおよぼすインシデントの発生数は減少すると考えられる．しかし，NIDSが検知するリスク不明のセキュリティイベント数は変化せず，運用コストは変わらない．したがって費用対効果が低下するため，リスク不明なセキュリティイベントは調査されなくなり，被害をおよぼすインシデントを構成するセキュリティイベントを検知したとしても，見逃してしまう可能性が高くなる．

もう1つはリスクの高いインシデントを迅速に発見するのが困難になる点である．現在のリスク評価可能なセキュリティイベントは十分ではなく，被害をおよぼすインシデントの発生直後にリスク評価可能なセキュリティイベントが検知されるとは限らない．例えば，コンピュータウィルスは内部ホストに感染した直後，他のホストに対する感染活動を開始するものがほとんどであるため，内部から外部への感染活動を検知すればインシデントを早期に，かつ高い確率で発見できる．不正侵入事件も，別ネットワークに対する攻撃の中継点として侵入ホストが利用される可能性もある．しかし，長期間にわたって潜伏し内部情報やパスワードをはじめとする機密情報を取得し続け

表 3.2: あるWindowsホストに対する24時間の検知セキュリティイベントとその検知数

セキュリティイベント名	攻撃の分類	送信元	検知数
(http_inspect) BARE BYTE UNICODE ENCODING	不審なトラフィック	外部	22
BLEEDING-EDGE EXPLOIT MS04-007 Kill-Bill ASN1 exploit attempt	不正侵入の試み	外部	2
BLEEDING-EDGE EXPLOIT MS04011 Lsasrv.dll RPC exploit (WinXP)	不正侵入の試み	外部	2
BLEEDING-EDGE EXPLOIT XML-RPC for PHP Remote Code Injection	不正侵入の試み	外部	10
BLEEDING-EDGE Potential SSH Scan	調査行為	外部	89
MS-SQL Worm propagation attempt	不正侵入の試み	外部	15
NETBIOS DCERPC IActivation little endian bind attempt	不正侵入の試み	外部	14
NETBIOS DCERPC ISystemActivator bind attempt	不正侵入の試み	外部	6
NETBIOS DCERPC Remote Activation bind attempt	不正侵入の試み	外部	14
NETBIOS SMB IPC$ unicode share access	不正侵入の試み	外部	1
NETBIOS SMB Session Setup AndX request unicode username overflow attempt	不正侵入の試み	外部	51
NETBIOS SMB Session Setup NTMLSSP unicode asn1 overflow attempt	不正侵入の試み	外部	35
NETBIOS SMB repeated logon failure	不正侵入の試み	外部	4
NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit attempt	不正侵入の試み	外部	2
NETBIOS SMB-DS IPC$ unicode share access	不正侵入の試み	外部	22
NETBIOS SMB-DS Session Setup AndX request unicode username overflow attempt	不正侵入の試み	外部	170
NETBIOS SMB-DS Session Setup NTMLSSP unicode asn1 overflow attempt	不正侵入の試み	外部	2
NETBIOS SMB-DS repeated logon failure	不正侵入の試み	外部	27
WEB-IIS view source via translate header	不正侵入の試み	外部	2
WEB-MISC WebDAV search access	不正侵入の試み	外部	12
SHELLCODE x86 NOOP	不審なトラフィック	外部	26
WEB-MISC robots.txt access	調査行為	外部	14
WIN-RPC ISystemActivator Interface Detected (135/tcp)	調査行為	外部	6
WIN-RPC LSA DS access (lsass.exe) Interface Detected (445/tcp)	調査行為	外部	3
WIN-RPC Plug and Play service (services.exe) Interface Detected (445/tcp)	調査行為	外部	15
WIN-RPC Remote Management Interface Detected (135/tcp)	調査行為	外部	3
合計			569

るケースや，Webサイトを改ざんされ被害が拡大してから表面化するケースなどもあり，早期発見が確実にできるとは言えない．NIDSの目的の1つは「リスクの高いインシデントの早期発見」だが，特に不正侵入については早期に発見できるとは言いがたく，NIDSを十分に活用できていない．

3.1.4 リスク評価可能なセキュリティイベントが少ない原因

リスク評価可能なセキュリティイベントが少ない原因は大きく分けて2つに分類できる．

1つは，NIDSがセキュリティイベントについて取得できる情報が限られている点である．既存のNIDSの多くはセキュリティイベントの発生に直接関わる情報のみを取得している．そのため，第3.1.1項でも述べたとおりリスク判別に利用できる情報は，主にセキュリティイベントの種類と送信元IPアドレス程度である．セキュリティイベントを効率的に評価するためには，セキュリティイベントだけではなく関連するトラフィックも分析し，より詳しい発生状況を確認する必要がある．

もう1つは，NIDS以外から得られるセキュリティイベントに関連した情報をネットワーク管理者が手動で収集している点である．セキュリティイベントが発生したホストへ直接ログインしての実地調査や，他のセキュリティデバイスからの情報取得などをセキュリティイベント毎に手動で収集するのは，大幅な負担となってしまう．このような作業を自動化し，さらに複数の情報を利用した条件を定めることでリスク評価作業の負担を軽減できると考えられる．

3.2 複数セキュリティイベントの全容把握における課題

第2.6.3項において述べたとおり，NIDSが検知したセキュリティイベントのログは，セキュリティイベント発生状況の異常発見やインシデントの全体像の把握に役立てられる．しかし，NIDSは長期間にわたり膨大な数量のセキュリティイベントを検知する．さらに，現在多くのNIDS実装で提供されているログのユーザインターフェースはセキュリティイベントをリスト状に表示している．この2つの事実により，ログとして蓄積されたセキュリティイベントの情報を効果的に利用するのが困難となっている．

3.2.1 セキュリティイベント検知数の実態

図3.1に2005年1月から同年12月までの間にsnortによって検知されたセキュリティイベント数の推移を示す．横軸は日付を表しており，縦軸はセキュリティイベントの検知数を表している．監視したネットワークは約500台前後のホストが接続し，その約1/3はFirewallによって外部からの接続を遮断している．NIDSのシグネチャは表3.2での検知に用いたシグネチャと同様である．

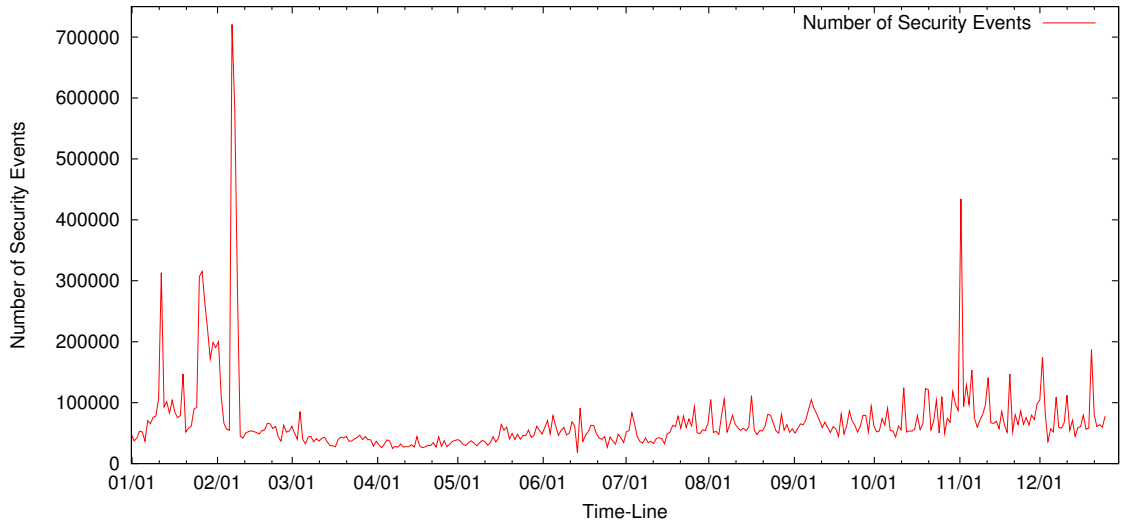

図 3.1: NIDS によって検知されたセキュリティイベント数の推移

図に示すとおり，一日あたり平均 68401 件のセキュリティイベントが検知されており，検知数が多い日は 10 万件を超えている．NIDS が検知するセキュリティイベント数はネットワークのアドレス範囲や構成によって変化するが，基本的には接続しているホスト数に比例して増加する傾向がある．より多くのホストが接続している大規模なネットワークでは，さらに膨大な数量のセキュリティイベントが検知されると予想される．

3.2.2 従来のログ表示用ユーザインターフェースの例

まず，ログ出力の例として，図 3.2に Snort[12] によるテキストファイルへのセキュリティイベントのログ出力を示す．どのようなセキュリティイベントがいつどのホストに対して起こったのかという内容が詳細に分かる反面，一括して表示できるセキュリティイベント数はごく限られてしまう．特定の情報を抽出するのは，条件が定まっていれば比較的容易に目的の情報を取得できる．しかし，このような表示形式によってセキュリティイベントを逐次的に確認し，全容を把握するのは明らかに現実的ではない．

セキュリティイベントのログを表示するユーザインターフェースの例として著名な RazorBack[22] と ACID[23] の画面を図 3.3と図 3.4にそれぞれ示す．

テキストファイルに出力されたセキュリティイベントの表示形式と比較すれば，いくらか一括表示できるセキュリティイベント数は増えているが，それでも表示件数には限りがある．実際の運用で，数千，数万件単位のセキュリティイベントの全容を把握するには不向きであると言える．

RazorBack も ACID も，条件の指定によるセキュリティイベントの抽出機能は充実している．しかし，セキュリティイベント発生状況の異常発見のためには抽出機能だけでは不十分であり，より各セキュリティイベントがどのような関連性をもっている

```
[**] [1:1417:9] SNMP request udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
01/20-03:04:55.654105 0:2:B3:EC:6C:D4 -> 0:E0:16:A0:EC:81
203.178.142.131:57681 -> 203.178.143.8:161 UDP TTL:254
Len: 85

[**] [1:1417:9] SNMP request udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
01/20-03:04:55.655367 0:2:B3:EC:6C:D4 -> 0:E0:16:A0:EC:81
203.178.142.131:57681 -> 203.178.143.8:161 UDP TTL:254
Len: 85

[**] [1:1411:10] SNMP public access udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
01/20-03:11:48.486076 0:2:B3:EC:6C:D4 -> 8:0:37:C:88:DD
219.52.100.54:1029 -> 203.178.143.108:161 UDP TTL:114
Len: 77

[**] [1:1417:9] SNMP request udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
01/20-03:11:48.486076 0:2:B3:EC:6C:D4 -> 8:0:37:C:88:DD
219.52.100.54:1029 -> 203.178.143.108:161 UDP TTL:114
Len: 77

[**] [111:2:1] (spp_stream4) possible EVASIVE RST detection [**]
01/20-03:30:08.834849 0:2:B3:EC:6C:D4 -> 0:6:5B:5:39:FE
61.156.20.97:10829 -> 203.178.143.28:22 TCP TTL:241
*****R** Seq: 0x3235D049  Ack: 0x0  Win: 0x0  TcpLen: 20

[**] [111:2:1] (spp_stream4) possible EVASIVE RST detection [**]
01/20-03:30:08.834854 0:2:B3:EC:6C:D4 -> 0:6:5B:5:39:FE
61.156.20.97:10829 -> 203.178.143.28:22 TCP TTL:241
*****R** Seq: 0x3235D049  Ack: 0x0  Win: 0x0  TcpLen: 20
```

図 3.2: Snort によるセキュリティイベントのログ出力例

か，あるいはどのようなタイミングでセキュリティイベントの発生頻度が変化したか，などの情報を的確かつ迅速に把握できなければならない．

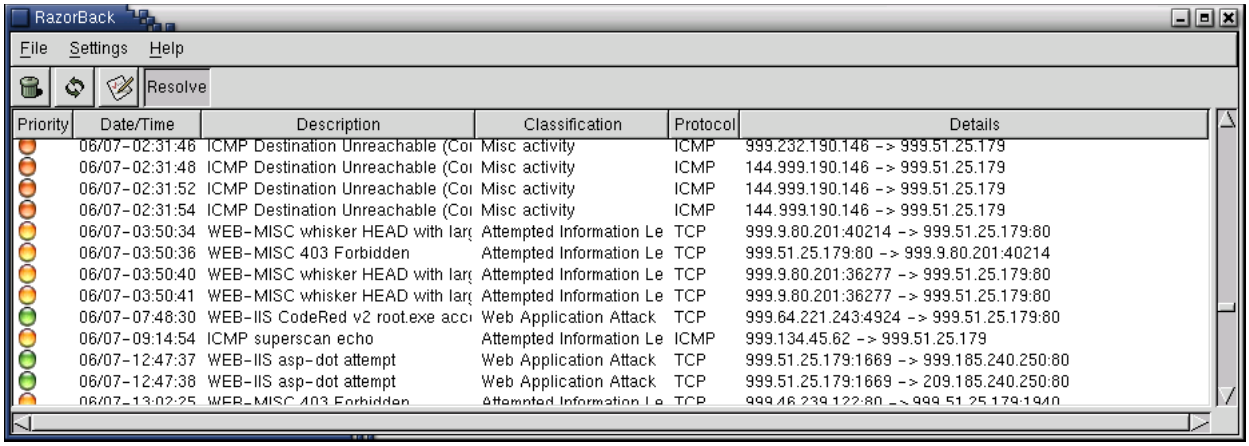

図 3.3: RazorBack を用いた Snort のログ表示インターフェース

また，リスト表示では各セキュリティイベントがどのようなタイミングで発生しているかを直感的に理解するのが難しい点も指摘できる．リスト表示では連続して発生しているように見える複数のセキュリティイベントでも，発生時間の間隔を考慮すると，実際はいくつかのグループに分かれる場合がある．特に被害をもたらすインシデントほど，複数のセキュリティイベントによって構成されるものが多く，インシデントの内容把握には関連セキュリティイベントの切り分けが必要となる．関連するセキュリティイベントを特定できれば，インシデントの発生時刻特定より容易となり，被害をおよぼしているインシデントにも迅速かつ適切な対応ができる．

ACID には全容把握を支援するための機能として，グラフ化機能が実装されている．図 3.5では ACID によってセキュリティイベントの発生件数を棒グラフによって表示している様を示している．ただし，このようにグラフ化したとしても，そのグラフに含まれるセキュリティイベントの内容は様々である．第 2.7節や表 3.1でも示したとおり，NIDS が検知するセキュリティイベントは多様な種類があり，一概にグラフとして件数を表示したとしても全容の把握とは言い難い．また，具体的に全容を把握しにくい例を図 3.6に示す．図 3.6ではリスト表示されたセキュリティイベントと，それを時間軸に従って発生時期を示している．右に示すセキュリティイベントのリスト表示では発生時刻によって並び替えられていたとしても，各セキュリティイベントがどのようなタイミングで発生しているかを認識しにくい．図のように数件あるいは数十件であれば，発生時刻を読み取り，各セキュリティイベントの発生状況も把握できるが，セキュリティイベント数が多くなればネットワーク管理者にとって負担の大きい作業となってしまう．

3.2.3 全容把握のための情報要約技術の必要性

ここまでで述べたとおり，現在の NIDS が提供しているユーザインターフェースではセキュリティイベント発生状況の異常検知，そしてインシデント解析において十分

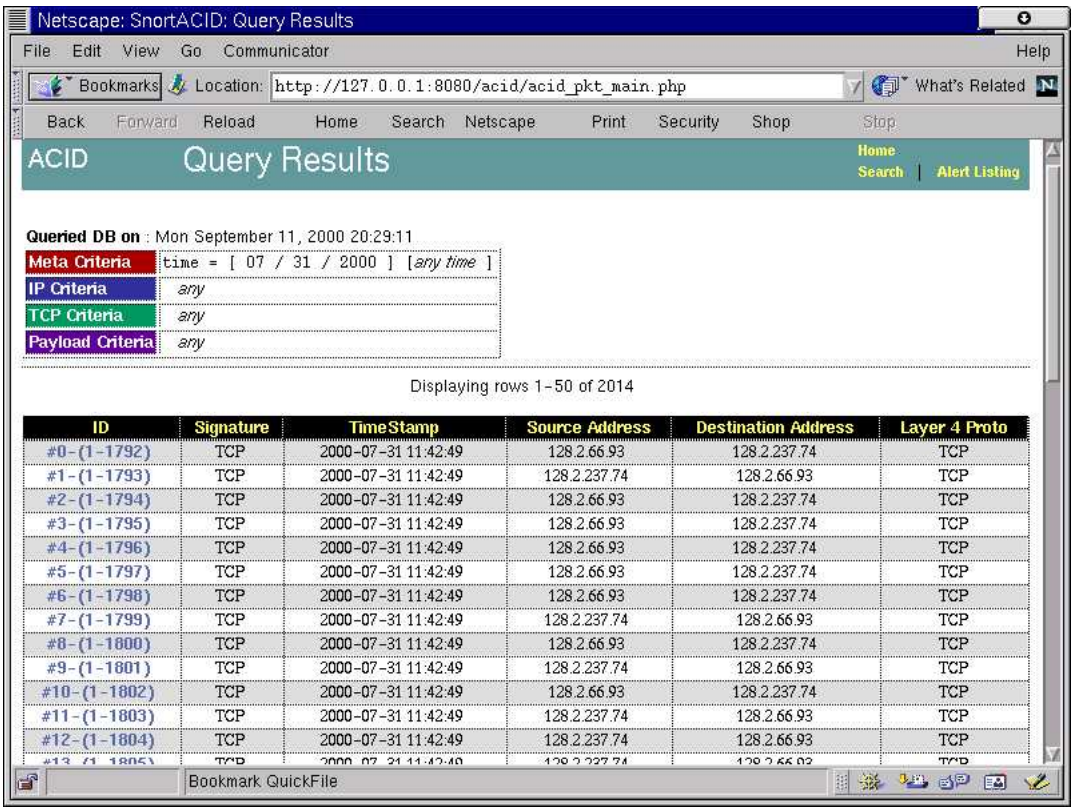

図 3.4: ACID を用いた Snort のログ表示インターフェース

な効果を発揮できないという問題点がある．より的確かつ迅速にセキュリティイベントの発生状況をネットワーク管理者に伝えるためには，ユーザインターフェースの改善が必須である．

ログのリスト表示によって発生状況の異常検知，関連セキュリティイベントの抽出が困難となるのは，ログの情報をほぼ未加工の状態で出力しようとしている点である．セキュリティイベントの発生件数がごく少なければ，リスト表示でも十分に状況把握が可能だが，発生件数の増加に伴って作業負担も増加する．ネットワーク管理者がユーザインターフェースによって認識できる情報量は有限であり，大量のセキュリティイベント発生状況を短時間で把握するためには，必要とされる情報を要約し，表示する仕組みが必要である．

また，グラフによる視覚化では，セキュリティイベントのログを活用できていない点も留意する必要がある．別種類のセキュリティイベントが同様の数量的変化をしたとしても，その変化が示す状況は，セキュリティイベントの種類によって異なる．情報の要約化を検討するためには，NIDS の特性を十分に考慮した上で，効果的な利用を実現するユーザインターフェースの設計が重要となる．

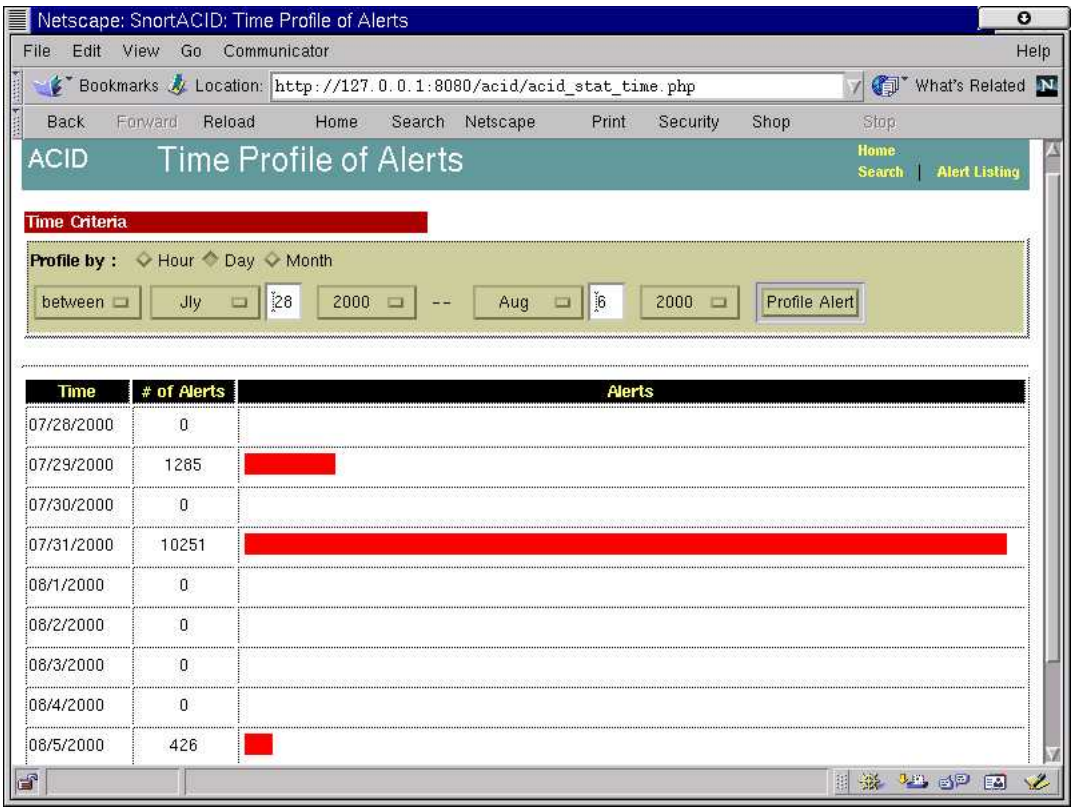

図 3.5: ACID を用いた Snort のログのグラフ化

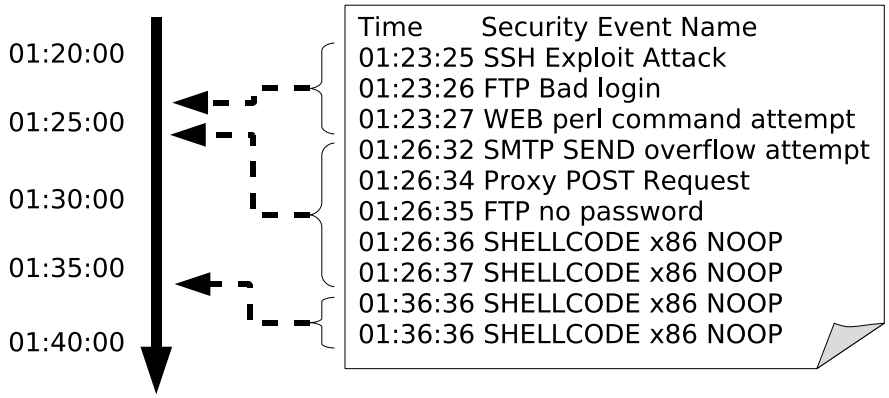

図 3.6: リスト表示されたログの時系列の可読性

3.3　まとめ

NIDS を運用する上での課題は 2 点にまとめられる．

1 つは，NIDS が検知する多くのセキュリティイベントは，単独でのリスク評価が困難となっている点である．評価できるセキュリティイベントでも，被害をおよぼすイ

ンシデントの発生後，早期に，かつ確実に検知されるセキュリティイベントとは限らない．そのため，NIDS をより効果的に運用するためには，新しいセキュリティイベントのリスク評価手法を検討する必要がある．

もう 1 つは，NIDS が出力するセキュリティイベント情報は全体の見通しが悪く，各セキュリティイベントの関連性や全体像の把握が困難な点である．インシデントの解析時には素早く関連するセキュリティイベントを抽出し，インシデントの正確な全容を理解しなければならない．セキュリティイベント発生状況の異常検知についても，全体を把握しなければ異常状態の発見は難しい．このような問題を解決するために，セキュリティイベントログを適切に要約しネットワーク管理者に伝達するユーザインターフェースが求められる．

第4章　運用効率化手法の提案

4.1　運用効率化のための戦略

本論文では従来のNIDS構成におけるセキュリティイベントの流れを示した図2.5をもとに，新しいセキュリティイベントの流れNIDSの構成モデルを提案する．新しいモデルには，多様な情報を収集し，セキュリティイベントのリスク評価に役立てる要素と，ログとして蓄積されたセキュリティイベントを要約し，発生状況の把握を容易化する要素を，それぞれ取り入れている．セキュリティイベント以外のトラフィックを監視し，従来のNIDS以上の情報を収集できる**Enhanced NIDS**，外部情報を用いてNIDSから出力されたセキュリティイベントのリスクを評価する**Security Information Manager**，そして，ユーザインターフェースとして蓄積されたセキュリティイベント情報を要約する**Log Aggregation System**の，3つの要素が不可欠である．各要素については第4.1.1項，第4.1.2項，第4.1.3項でそれぞれ述べる．

これらの要素を導入した場合のモデルの概要を図4.1に示す．Enhanced NIDS，Security Information Manager，Log Aggregation Systemは様々な実現手段が考えられる．従って，それぞれが単一の技術である必要はなく1つあるいは複数の技術を用いたシステムもありえる．このモデルに沿った技術を導入することによって，NIDS運用の全体的な効率化を目指す．

4.1.1　Enhanced NIDS

Enhanced NIDSは第3.1.4項で述べたセキュリティイベントがもつリスク評価に利用できる情報の少なさを解決するための要素として提案する．従来のNIDSは，セキュリティイベントのトラフィック発見を目的としている．しかし，トラフィックにはセキュリティイベントだけではなく，セキュリティイベントに関連するトラフィックも存在する．Ping of Death[24]のように1パケットしか送信されないセキュリティイベントもあるが，一般的にはセキュリティイベントの検知前後に検知対象とされないトラフィックを送受信しているセキュリティイベントが多い．Stateful Signature[25, 26]の技術は，このようなトラフィックも解釈することで，誤検知を削減している．Enhanced NIDSとは通常のセキュリティイベント検知だけではなく，関連するトラフィックもあわせて監視し，セキュリティイベントのより詳細な状況を把握するNIDSとする．把握された情報はセキュリティイベントの出力時にパラメータとして付与され，リスク評価に用いられる．

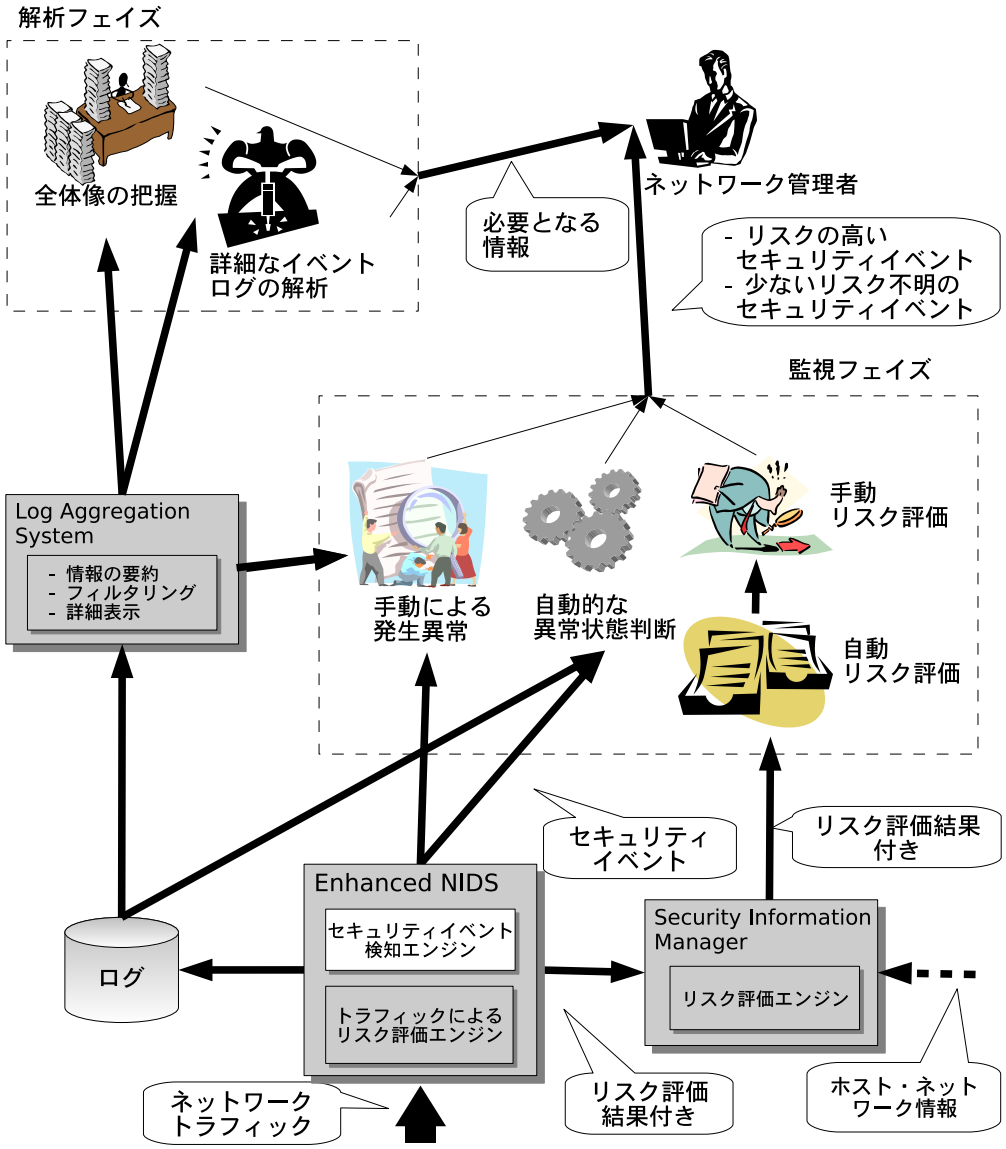

図 4.1: 本論文で提案する NIDS の構成モデル

4.1.2 Security Information Manager

第 3.1.4項の問題点を解決するための，もう 1 つの要素として，Security Information Manager（SIM）を提案する．NIDS によって得られたセキュリティイベントの情報と，NIDS 以外の手段で得られた情報を元に，セキュリティイベントのリスクを評価するシステムを SIM と呼ぶ．NIDS 以外から得られる情報とは，管理ネットワーク内のトポロジ情報，接続している各ホストのソフトウェア情報，Firewall などの他のセキュリティデバイスから発生したセキュリティイベントのログなどが挙げられる．また，過去に NIDS が検知したセキュリティイベントログ情報も利用できる．必要に応じて自動的にこれの情報を取得し，セキュリティイベントのリスク評価に用いる．評価の結

果は Enhanced NIDS と同様，セキュリティイベントに付与され監視や解析に利用される．Target-based IDS(TIDS)[27] も SIM の一形態である．多くの SIM 実装ではリスク評価機能の他にユーザインターフェースとしての役割も兼ねているが，本論文ではリスク評価機能のみに着目する．

4.1.3 Log Aggregation System

第 3.2.3項の問題点を解決するために Log Aggregation System を提案する．Log Aggregation System とは，多数のセキュリティイベントのログ情報を要約し，複数セキュリティイベントの長期的な発生状況を把握するためのシステムである．リスト表示によるセキュリティイベントログの状況把握には限界があるため，統計的処理，あるいは視覚的処理によって多数のセキュリティイベントの発生状況を素早く理解できるように処理する．また，インシデントの解析時には個別のセキュリティイベントログの詳細を検査しなければならないため，素早く目的のセキュリティイベント情報にアクセスできる仕組みも必要となる．

4.2 新しい NIDS 運用モデルに利用する手法

本節では，図 4.1で示した Enhanced NIDS，Security Information Manager，Log Summarizing System の各要素に，どのような手法を採用するのかについて詳細に述べていく．Enhanced NIDS として **Session based NIDS**，Security Information Manager として **OS based Risk Evaluation System**，そして Log Summarizing System として **Security Evnet Log Visualizer** を用いる．以上の技術を各機構において利用する．本節で各技術の概要を述べた上で，次章以降で各技術とその設計，実装について詳細に述べていく．

4.2.1 Session based NIDS

第 3.1節で述べた通り，従来の NIDS では迅速，かつ確実に検知したセキュリティイベントのリスクを評価するのが難しい．そこで，既存の NIDS が「セキュリティイベントを発生した事実」しか検知していない点に着目する．セキュリティイベントが発生した通信を，セキュリティイベントの検知後も継続的に監視することで，そのセキュリティイベントがどのような影響をおよぼしたかを判断できると考えられる．本論文では，このような手段によってセキュリティイベントによる影響を判断し，リスクを評価する Enhanced NIDS を **Session based NIDS** と呼ぶ．具体的には不正侵入の試みやスキャン行為，不審な挙動を監視し，そのような行為をうけたホストからの応答によって成否の判別ができるようになる．

例えば，あるソフトウェアのバッファオーバーフロー脆弱性を利用した不正侵入の試みでは，攻撃が成功すれば当該ソフトウェアは通常と異なる挙動を起こす可能性が

高い．これに伴って，通信の応答が変化する，あるいは応答すべき手順で応答を返さなくなるなどの状況が考えられる．調査行為ではより顕著に変化が現れると考えられる．調査行為は応答によって目標のソフトウェアの情報を得るため，応答を監視すれば成否の判断は容易である．また，応答を監視する技術であるため即時性も高い．異常な応答や正常な応答の規則を正確に示したシグネチャを用意すれば確実性も期待できる．Session based NIDS の詳細は第 5章で述べる．

4.2.2 OS based Risk Evaluation System

不正侵入の試みでは，目標とするソフトウェアが異なると攻撃が成功しにくくなり，リスクが低くなるという特徴について第 3.1節で述べた．この特徴を利用し，管理者が把握した管理ネットワーク内のホストが使用している OS 情報と，検知されたセキュリティイベント内容とを比較することで，リスクを評価する手法を OS based Risk Evaluation System と呼ぶ．本論文では各ホストの OS 情報を自動的に取得し，リスクが低いと判断する方式を提案する．OS based Risk Evaluation System は第 6章で詳しく述べる．

4.2.3 Security Event Log Visualizer

本論文ではセキュリティイベントログを要約する手法としてログ情報の視覚化を提案する．多数のセキュリティイベントログも可読性が悪く，インシデント発生時の全体的な状況把握や，関連するセキュリティイベントの抽出が困難であると，第 3.2節で述べた．このような問題点を解決するために，本論文ではセキュリティイベントログの視覚化によって情報を要約させる手法を Security Event Log Visualizer と呼ぶ．Security Event Log Visualizer は第 7章で詳しく述べる．

4.3 まとめ

第 3章で指摘した NIDS の運用における問題点を解決するために，本論文では攻撃だけではなく攻撃の応答も監視することでセキュリティイベントのリスクを評価する Session based NIDS，OS 情報に基づきリスクが低いと予想されるセキュリティイベントを除外する OS based Risk Evaluation System，多数のセキュリティイベントログを可視化し全体的な状況把握を容易にする Security Event Log Visualizer の 3 手法を提案する．

第5章　セッション追跡によるリスク評価型NIDS

5.1　はじめに

これまでで述べたとおり，既存のNIDSで大量に検知されるセキュリティイベントはリスクの評価にコストがかかる．そのため，適切にセキュリティイベントのリスク評価がなされない限り，リスクの高いインシデントを早期にかつ確実に発見するのは困難である．これでは，NIDS運用の目的が達成されない．

本章で解決すべき問題点は外部からの不正侵入の試み，調査行為のセキュリティイベントを，迅速かつ確実にリスク評価ができない点である．本章ではセキュリティイベントの検知後も当該トラフィックの監視を続け，その応答などからリスクを評価するSession based NIDSについて述べる．

5.2　関連研究

本論文で提案したモデルを構成する要素の1つである，Enhanced NIDSに関連する技術を関連研究として挙げる．各研究，技術ではセキュリティイベントと考えられるトラフィックだけではなく，関連するトラフィックも監視し検知精度の上昇やリスク評価に利用している．

5.2.1　Stateful Signature

通常のシグネチャ型NIDSではトラフィック中に含まれる特定のキーワードの発見によって攻撃を検知する．しかし，攻撃ではないトラフィックにも同じキーワードが偶然含まれる可能性があり，誤検知をおこす原因となる．これを解決するために，Stateful Signature[25]では単一のトラフィックだけではなく，TCPの同一セッションなどから複数のパケット，あるいはキーワードを検知対象にできる．これにより攻撃に関連する一連の動作を識別することが可能となり，誤検知の削減につながる．

5.2.2　NetSTAT

NetSTAT[28]はネットワークで観測されたセキュリティイベントやホスト上で発生したセキュリティイベントなど，複数の事実から侵入を発見する研究である．ホスト

の状態やネットワークトラフィックの情報，トポロジ情報などを組み合わせたルールを用いる事で，管理ネットワーク内で起きたセキュリティイベントを正確に検知できる．また，各アプリケーションの状態遷移も検知に利用できるため，汎用性が高く設計されている．

ただし，ネットワーク側は通信の開始から終了までの状態遷移，ホスト側は各セキュリティイベントでの状態を記録する必要がある．そのため広帯域なネットワークでは，監視ホストに高い演算能力と多くの記憶領域が要求される．また，トポロジの変更に伴い，その都度適用されるルールを変更しなければならない．多数のホストが動的に接続されるネットワークでは，ルール管理ホストに高い負荷がかかると予想される．さらに，各ホストにシステムを導入しなければならないため，不特定多数のホストが接続する場合，システムが機能しない可能性が高い．このため広帯域，または多数のホストが動的に接続するネットワークにおいては運用が困難であると考えられる．

5.2.3 継続的な通信の記録

NIDS の実装によっては攻撃そのものの検知だけではなく，その後のトラフィックも継続的に記録し続けることができる．この機能は Snort[13] や NFR[26]，NetScreenIDP[29] で実装されており，記録されたトラフィックをリスク評価の手がかりとして利用できる．しかし，確実にリスク評価するためには，検知された全てのセキュリティイベントを検査しなければならず，管理するネットワークの規模の拡大や NIDS が検知したセキュリティイベント数の増加によって運用コストが高くなってしまう．また，知識や経験の無いネットワーク管理者は通信の意味を理解できないため，攻撃を検知した後のトラフィックから正確なリスク評価をするのが難しい．さらに，豊富な知識があったとしても，人為的なミスにより誤判断，あるいは見逃しが起こる可能性がある．

5.3 要件定義

本節では NIDS が検知したセキュリティイベントのリスク評価における要件を定義する．要件の定義には NIDS の目的の 1 つである「リスクの高いインシデントの早期検出」，そして第 3.1.3項で述べたリスク評価の困難さによって引き起こされる弊害を参考とした．

即時性　リスク評価の作業負担が大きいければ，インシデントの発見が遅れてしまう．発見の遅れは被害の拡大に繋がるため，セキュリティイベント検知後の早急なリスク評価処理が求められる．

確実性　誤った結果が多ければリスク評価の信頼性が失われ，システムとしての利用価値も低下してしまう．したがって，評価結果に一定の信頼性が得られるシステムでなければならない．

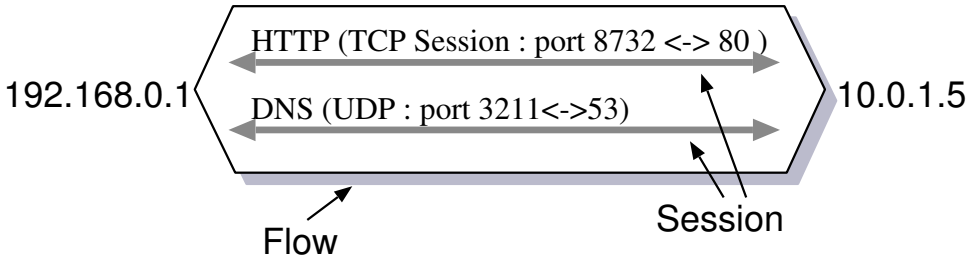

図 5.1: フローとセッションのモデル

規模性　評価の手法によっては管理ホスト数の増加に比例して，作業負担が増加してしまう．多くのホストで構成されているネットワークでの運用において，規模性は重要な要件となる．

多様性　多様性とはリスクを評価できるセキュリティイベントの多さをあらわす．いかに適切な評価が行えるとしても，一部の攻撃にしか対応できなければ，有用性が著しく低下してしまう．

5.4 解決手法

本章では攻撃のみならず，その応答も併せて監視する Session based NIDS を提案し，その概念について論じる．

5.4.1 フローとセッション

本論文における，フローとセッションという 2 つの概念を定義する．概要を図 5.1に示す．

2 つの IP アドレス間でのトラフィックは，その内容に関わらず全て**フロー**と定義する．各フローには 1 つ，あるいは複数のプロトコルを利用した通信が行われている．例えば，TCP の HTTP による Web リソースの取得，UDP による DNS 問い合わせ，ICMP のエコーリクエストなどが挙げられる．このような特定のサービスのための一連の通信を**セッション**と定義する．

ステートフル型の通信である TCP は 1 つの TCP セッションを本論文で定義するセッションとし，ステートレスである UDP は同一の送信元ポート番号，送信先ポート番号の組合せをセッションとする．ICMP はエコー要求やエコー応答などの関連性のある通信を 1 つにまとめ，セッションとする．その他の IP 通信では，プロトコルの種類毎にセッションとする．

5.4.2 Session based NIDS

本研究では迅速かつ確実なリスク評価のために，セッションに着目する．ネットワークを経由した攻撃の多くは，その攻撃に対する応答のトラフィックが存在する．例として不正侵入を試みた後にその成功を示す応答が返される，あるいは攻撃対象のホストに対して調査をした後に，その結果が返されるなどが挙げられる．多くの場合，これらの応答には攻撃の成否を判断する手がかりが含まれる．この情報をもとに NIDS が自動的に攻撃の成否を判断し，その情報をセキュリティイベントに付与する．管理者がセキュリティイベントを見る際，攻撃の成否の情報によって同種類のセキュリティイベントにおいてもリスクの順位付が可能となる．攻撃が成功し，リスクが高いと判断されたセキュリティイベントについてのみ緊急に対応することで，NIDS による監視運用コストの大幅な減少が期待できる．

本機構は NIDS の検知システム内において即時的にセキュリティイベントのリスクを評価する．別の手法として，関連する全ての通信の記録を 2 次記憶領域に保存し，後から応答を自動的に解析することで，セキュリティイベントをリスク評価することもできる．しかし，高帯域，あるいは多数のホストが接続しているネットワークにおいて，通信の記録は大量の記憶領域が必要となり，記憶領域が圧迫され記録に失敗する可能性がある．そのため，NIDS の検知システム内で評価を行い，結果のみを記録することで，記憶領域の圧迫を防ぐ．

このような動作機構を持つ NIDS を Session Based NIDS (以下，S-NIDS) と呼ぶ．

5.4.3 追跡型監視の提案

S-NIDS では主に，攻撃の通信が行われたホストからの応答を監視し，その応答によって攻撃の成功，あるいは失敗を判断する．そのために，以下に挙げる 2 つのシグネチャを用意し，攻撃とその応答を監視する．

1. 攻撃のトラフィックを示すシグネチャ
2. 攻撃の成功，あるいは失敗を示す応答トラフィックのシグネチャ

最初に全てのパケットを 1と比較する．攻撃が行われた可能性を示すパケットを検知した場合，そのセッションの応答に該当するパケットを 2と比較し，合致すれば攻撃が成功した可能性があると判断し，記録，通知を行う．その際，経過時間や該当する攻撃が必要とするセッション内でのパケット数に応じて，そのセッションに対する反応検知シグネチャの保持条件を定義することで，新たな誤検知を軽減する．

5.4.4 シナリオ

本項では，S-NIDS が有効に動作する例としてバックドアを用いた不正侵入を挙げる．バックドアは CodeRedII ワーム [30] が設置するプログラムとする．

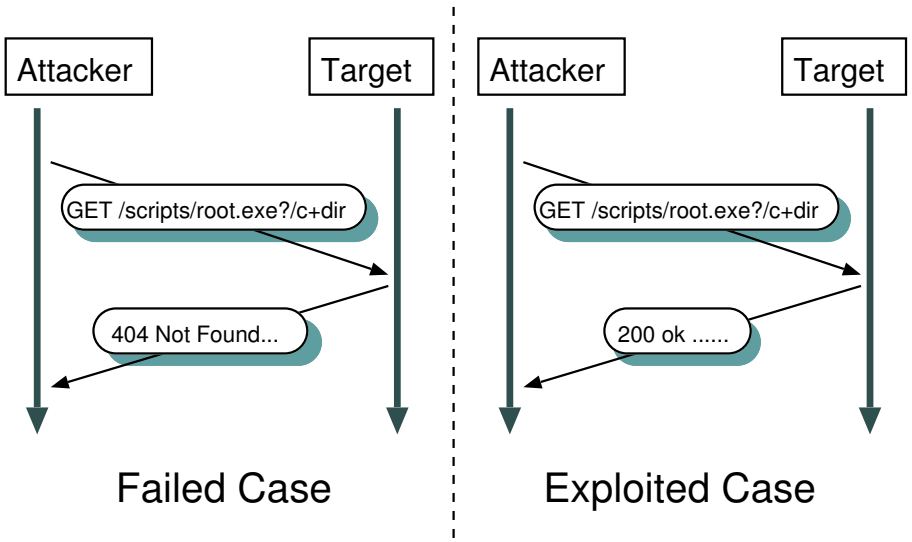

図 5.2: バックドアを利用した攻撃の成功例と失敗例

CodeRedII ワームは Microsoft 社の Web サーバである Internet Information Server（IIS）の脆弱性を利用し感染する．CodeRedII ワームは感染した後，感染コンピュータのハードディスクの複数箇所に “root.exe” というバックドア用のプログラムファイルをコピーする．さらに，IIS の設定を変更し，遠隔から任意のコマンドを実行可能にする．

ここで，任意のコマンドの実行例を図 5.2に示す．バックドアが設置されているホストに対しては，“GET /scripts/root.exe?/c+dir” というリクエストの送信によって，目標ホスト内部のディレクトリが参照できる．図 5.2の左部分は感染していないホストの例である．ホストに対するリクエストは HTTP[31] のファイル取得不能エラーである “404” などが応答として返され，攻撃は失敗する．一方で，右側は感染したホストに対する不正アクセスである．このケースではアクセスが成功するため，ファイル取得成功コード “200” と，実行結果であるディレクトリのファイル名一覧が返される．

図 5.3ではバックドアへのアクセスに対する，従来の NIDS と S-NIDS の検知手法を比較している．図のように，従来の NIDS では攻撃者が送信するトラフィックのみを監視していたため，“GET /scripts/root.exe” というリクエストは全て同様のセキュリティイベントとして検知する．S-NIDS では，送信されたリクエストを「攻撃があったことを示すパケット」として検出し，続けてその攻撃に対するホストの応答を監視する．ファイル取得成功コードの “200” が返された場合，root.exe へのアクセスが成功したと判断できる．以上の動作により，成功した攻撃と，失敗した攻撃を判別できるようになる．

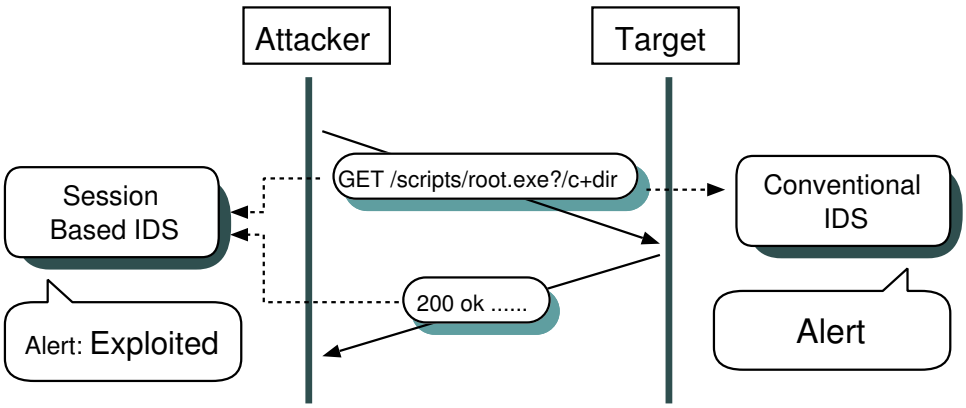

図 5.3: システム動作例

5.5 S-NIDS の設計

S-NIDS を実現するために必要となる全体，及び各部の設計について述べる．

5.5.1 トリガーシグネチャとトラッキングシグネチャ

S-NIDS はその挙動を定義されたルールにもとづいてセキュリティイベントを検知，評価するミスユース型 NIDS の応用である．ルールは不正侵入の試みや調査行為そのもののトラフィックパターンを示すシグネチャと，その応答から成否を判断するためのトラフィックパターンを示したシグネチャの 2 種類で構成しなければならない．以降は，前者を**トリガーシグネチャ**，後者を**トラッキングシグネチャ**と呼ぶ．

概要を図 5.4にて示す．トリガーシグネチャは追跡のきっかけとなるシグネチャであるため，全てのトラフィックと比較する．その後，以前に当該セッションでトリガーシグネチャが検知され，登録されたトラッキングシグネチャとトラフィック比較する．トラッキングシグネチャとして定義する応答トラフィックは，成功と失敗の双方が明らかであるものや，同じ結果でも複数のパターンが存在するセキュリティイベントが考えられる．よって，トラッキングシグネチャは複数登録できる必要がある．

図 5.4の例ではトラッキングシグネチャ A が検知されれば攻撃が成功したものと判断し，B が検知されれば失敗したと判断するルールとなっている．また，通信障害やアプリケーション異常によってシグネチャに一致するトラフィックが出現しない可能性もある．記憶領域の圧迫を防ぐために時間切れを設定し，長時間経過した後は未知の状態として記録に残す．時間切れまでの長さは S-NIDS を動作させるホストの環境やネットワークの環境に応じて変化する．さらにセキュリティイベント毎にも異なるので，柔軟に設定できる実装が必要である．

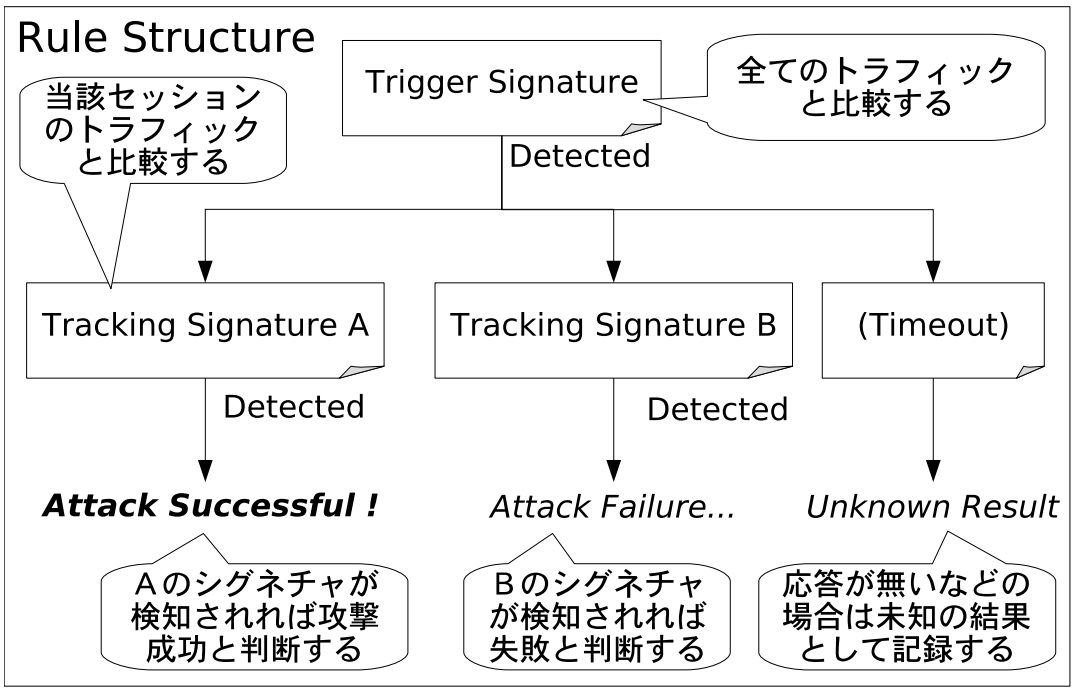

図 5.4: S-NIDS の検知ルールの構成

5.5.2 複雑なルールの記述

トラッキングシグネチャで監視すべき対象は攻撃対象ホストからの反応だけに限らない．S-NIDS では，基本的にリクエストをトリガーシグネチャとし，レスポンスをトラッキングシグネチャとするルールの記述を想定している．しかし，場合によってはリクエスト，レスポンス，リクエストのように連続したトラフィックが観測されて，初めて侵入とみなせる攻撃も考えられる．このようなインシデントに対応するため，トラッキングシグネチャが複数連続して記述できる仕様とする．

また，トラッキングシグネチャが攻撃者から攻撃目標に対するリクエストであるか，攻撃を受けたホストからのレスポンスのパケットであるかを判断しなければならない．各トラッキングシグネチャにはトラフィックの宛先が攻撃元か，あるいは攻撃先なのかを示す項目が必要となる．

トラッキングシグネチャでは当該セッション以外のトラフィックを検知できる実装が望ましい．コンピュータウィルスはホストへの感染後に攻撃元とは別のホストに対して感染活動を開始する．このようなセキュリティイベントでは，別ホストへのトラフィックを監視することでリスクの高いインシデントの発見となる．この他にも連続したスキャンの発見にも有効な機能となるため，当該セッション以外も追跡可能な仕様が望ましい．

5.5.3 ログの種別

S-NIDS は攻撃の成否の情報をセキュリティイベントのログに付与する．従来の NIDS は同種類のセキュリティイベントを全て一様に記録しているが，S-NIDS では検知された応答に応じたログを出力する必要がある．セキュリティイベントに付与すべき情報を後述する．

攻撃の成功　トラッキングシグネチャと比較し，攻撃成功と判断されたセキュリティイベントに付与する情報．

攻撃の失敗　トラッキングシグネチャと比較し，攻撃失敗と判断されたセキュリティイベントに付与する情報．

未知の応答や反応　トラッキングシグネチャに一致しない応答トラフィックや，応答が返されるはずのタイミングで何もトラフィックが流れないなど，想定されていない状況が発生したセキュリティイベントに付与される情報．

単独の検知　もともと応答などが期待されない攻撃や，不審なトラフィック，プロトコル異常などを示すシグネチャが検知された際に付与される情報．

5.6 S-NIDS の実装

第 5.5節の設計に従い，S-NIDS を実装した．開発環境は OS に Debian GNU/Linux 3.1 testing を，開発言語に C 言語を用いた．実装の際には [32] で述べられている，NIDS の構造や NIDS に必要となる機能，NIDS で検知するトラフィックの特徴を，S-NIDS の構造を検討する上で参考にした．また，[33] では実際の NIDS 運用に基づいたインシデント検出例などが述べられており，実装の上で参考にした．

5.6.1 実装の構成

S-NIDS の全体構成図を図 5.5に示す．S-NIDS の実装は主にトラフィック取得モジュール，解析エンジン，検知エンジン，出力エンジンの 4 つのグループとキュー管理機構，タイマー管理機構の 2 つのマネージャーによって構成される．

トラフィック取得モジュール　ネットワークデバイスを経由してネットワークのトラフィックを取得する部分．本論文の実装では libpcap[34] を利用している．デバイス毎に独立したスレッドで動作し，取得したトラフィックをパケット単位でキュー管理機構に渡す．さらに pcap 形式で保存されたファイルからの読み出しなど，ネットワークデバイス以外からの入力も可能となっている．

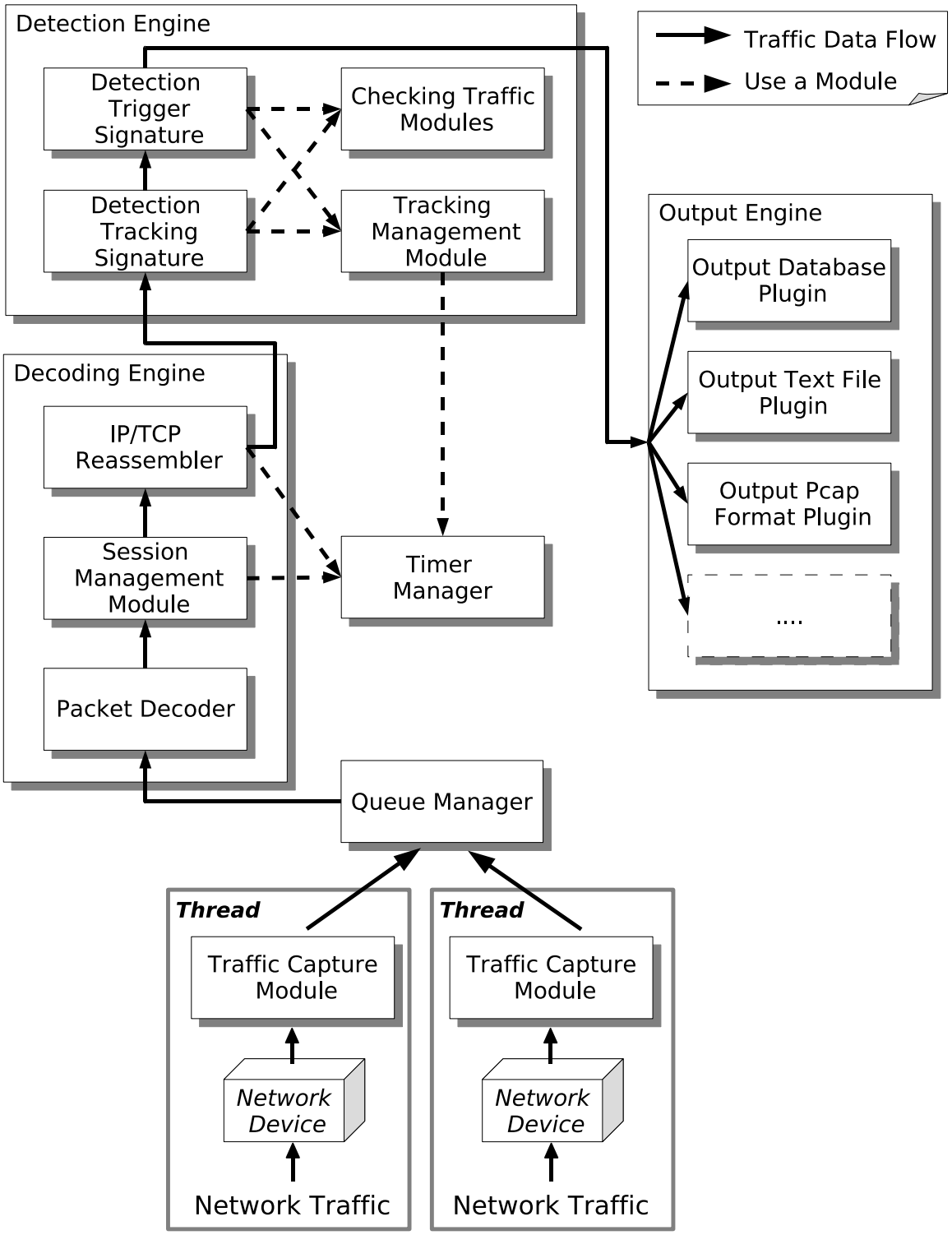

図 5.5: S-NIDS 全体構成図

キュー管理機構　トラフィック取得モジュールから渡されたトラフィックデータをキューイングし，解析エンジンに順次転送する機構．通常のネットワークトラフィックは発生頻度が不規則であり，急激なトラフィック増加が発生する可能性がある．また，解析や検知の処理，あるいは結果の出力に時間がかかってしまうと，その間のトラフィックを取りこぼしてしまう恐れがある．トラフィックの取りこぼしを最小限に防ぐため，キューに蓄えた後に順次解析する構造とした．

解析エンジン　検知エンジンでシグネチャとトラフィックを比較するために，キュー管理機構から渡されたパケット単位のトラフィックデータを正規化する．具体的にはパケットのヘッダ解析，フローやセッションの管理，そしてフラグメントされた IP パケットや TCP パケットを再構築する．厳密には，フローを検索した後に IP パケットのフラグメントを処理し，セッションを検索する．そしてセッションが TCP のセッションであり，パケットがフラグメント化されていれば再構築する．このようにして正規化されたトラフィックデータを検知エンジンに渡す．

タイマー管理機構　S-NIDS はトラッキングシグネチャやセッション情報など，保持すべきデータの内容が通常の NIDS と比較して多い．全ての情報を保持し続けると，一時記憶領域を圧迫してしまう．したがって，長時間観測されていないフローやセッション，あるいはトラッキングシグネチャを，それぞれある一定時間毎に破棄しなければならない．そのため，様々なモジュールから利用できる汎用的なタイマーを実装した．

検知エンジン　解析エンジンによって正規化されたトラフィックをシグネチャに基づいて検査する．検査すべきシグネチャは全てのトリガーシグネチャと，検知対象として登録されたトラッキングシグネチャである．トリガーシグネチャを検査した後にトラッキングシグネチャを検査すると，トリガーシグネチャの検知によって登録されたトラッキングシグネチャに対しても，同様のトラフィックデータで再び検査してしまうため．そのため，トラッキングシグネチャを先に検査している．

出力エンジン　検知エンジンによって検知されたセキュリティイベントをログとして出力する．出力形式は用途によって様々であるため，プラグインとして複数の出力手段を用意する．代表的なものとしてテキスト形式によるファイルへの書き込み，pcap 形式によるファイルへの書き込み，データベースへの書き込みを提供している．

5.6.2　ルール書式

図 5.6に S-NIDS が利用するルールの書式例を示し，記述すべき項目について述べる．

図 5.6では 1 行目がルール全体の基本設定となっている．各ルール内で最も上段のシグネチャがトリガーシグネチャに相当する．図 5.6では 2 行目（シグネチャa1）がこれにあたる．トリガーシグネチャは複数記述可能であり，同一の検知対象セキュリティイベントに対して複数パターンを登録できる．トリガーシグネチャの入れ子となってい

```
1| action (rule_name, protocol, sport->dport) {
2|   sig_type:direction:(signature a1){
3|     sig_type:direction:(signature a2);
4|     sig_type:direction:(signature a3);
5|   }
6| }
```

図 5.6: ルールの記述書式

るシグネチャがトラッキングシグネチャとなる．図 5.6では 3，4 行目（シグネチャa2, a3）が 2 行目のシグネチャに対するトラッキングシグネチャとなる．トラッキングシグネチャa2 と a3 にはそれぞれ異なるシグネチャが記述され，検知されたシグネチャの種別に応じた結果が出力される．シグネチャの記述については第 5.6.4項にて詳細に触れる．

まず，ルールの主な設定項目について説明する．

action　ルールの属性を指定する．攻撃として，シグネチャのパターンと一致するトラフィックを発見した際にセキュリティイベントとして検知する “alert”，特定のトラフィックをセキュリティイベントとして検知しないようにする “ignore” を指定する．

rule_name　ルールの内容を説明するメッセージを指定する．

protocol　レイヤー 4 のプロトコルを指定する．“TCP”，“UDP”，“ICMP” はそれぞれ文字で記述し，その他のプロトコルはプロトコル番号で指定する．

sport / dport　最初に検知すべき TCP，UDP パケットのポート番号を指定する．sport は送信元ポート番号，dport は宛先ポート番号をそれぞれ指定する．指定が無い項目は “any” とする．TCP，UDP 以外のプロトコルで指定する必要はない．

5.6.3 ルール書式の応用

図 5.6ではトリガーシグネチャの中に入れ子としてトラッキングシグネチャa1，a2 を並列して指定しているのみだが，トラッキングシグネチャの中に入れ子のトラッキングシグネチャを記述することもできる．図 5.7では連続したトラッキングシグネチャの記述例を示している．図中のルールの場合はシグネチャb1 を検知した後に，シグネチャb2 がトラッキングシグネチャとして当該セッションに登録される．そして，当該セッションのトラフィックと b2 が一致した場合はシグネチャb2 が破棄され，シグネチャb3

```
1| action (rule_name, protocol, sport->dport) {
2|   sig_type:direction:(signature b1){
3|     sig_type:direction:(signature b2){
4|       sig_type:direction:(signature b3);
5|     }
6|   }
7| }
```

図 5.7: 連続したトラッキングシグネチャの記述

```
sig_type:direction:( item1=value1; item2=value2; ...)
```

図 5.8: S-NIDS のシグネチャ書式

が当該セッションに登録される．このようにトラッキングシグネチャの多段指定にも対応し，より複雑なルールの設定が可能となっている．

5.6.4 シグネチャ書式

シグネチャで指定できるトラフィックの特徴は，トリガーシグネチャとトラッキングシグネチャで共通した項目となる．図 5.8に基本的なシグネチャの書式を示す．

最初にシグネチャの種別を示す *sig_type* を指定する．*sig_type* で指定できる種別を表 5.1に示す．本実装ではルール記述の自由度を上げるため，指定するシグネチャ種別はトリガーシグネチャ，トラッキングシグネチャに関わらず，その内容を指定できる．例えば最初のトリガーシグネチャに相当するシグネチャで exp（攻撃成功を意味するシグネチャ）も指定できる．攻撃とその応答の状態遷移は多様であるため，高い汎用性が必要になると考えられる．指定したシグネチャ種別はセキュリティイベントの検知後，ログとして出力する際に付加情報として記録されるのみである．

シグネチャ種別の後には攻撃元ホストから攻撃先へのトラフィックか，攻撃先から攻撃元への応答などのトラフィックか，あるいは当該セッション以外のトラフィックを検知すべきなのかを *direction* で指定する．*direction* で指定できる値を表 5.2に示す．攻撃元ホストから攻撃先へのトラフィックを “fwd”，攻撃先から攻撃元へのトラフィックを “bak” とする．さらに第 5.5.2項の設計を反映し，“new” と記述することで当該セッション以外のトラフィックを監視できる．“new” を指定した場合は，図 5.9の書式に従う必要がある．図 5.9に現れるフィールドの説明は以下である．

表 5.1: S-NIDS に用いられるシグネチャ種別

シンボル	説明
trg	“Trigger”, 攻撃そのもののトラフィックを示すシグネチャ
slnt	“Silent”, セッションの追跡には必要となるが，記録しないことを推奨するシグネチャ
solo	“Solo”, 単独でしか検知されない，応答が期待できない攻撃等のシグネチャ
exp	“Exploited”, 攻撃の成功を示すシグネチャ
fail	“Failure”, 攻撃の失敗を示すシグネチャ
ukwn	“Unknown”, 予期せぬ応答を示すシグネチャ
pass	“Pass”, これが検知された場合，後のシグネチャは検査しないようにするシグネチャ

表 5.2: シグネチャの方向 指定項目

シンボル	説明
fwd	“Forward”．攻撃者から攻撃対象のホストへ送られる，攻撃パケット.
bak	“Back”．行われた攻撃に対する反応・結果として，攻撃対象のホストから攻撃者に送信されるパケット.
new	“New Session”．当該セッション以外のトラフィックを指定できる.

shost / dhost　監視する別セッションの送信元ホスト，送信先ホストを指定する．特に指定しない場合は “any” とする．また，上位のシグネチャを検知した際のトラフィックから項目内容を引き継げる．$taddr は攻撃先 IP アドレス（Target Address）を$aaddr は攻撃元 IP アドレス（Attacker Address）をそれぞれ指定できる.

protocol　第 5.6.2項で説明している図 5.6の *protocol* と同様にレイヤー 4 のプロトコルを指定できる.

sport / dport　第 5.6.2項で説明している図 5.6の *sport*，*dport* と同様に TCP, UDP のポート番号を指定できる．上位シグネチャ検知時の情報を引き継ぐ場合は，$tport で攻撃先ポート番号（Target Port）を，$aport で攻撃元ポート（Attacker Port）を指定できる.

具体的には図 5.10のように記述する．図 5.10では，上位のシグネチャが検知された際の攻撃先ホストから任意の宛先に対してポート番号 1433 の UDP トラフィックを発見した場合に限り，シグネチャの内容とトラフィックを比較する.

```
sig_type:new(shost->dhost, protocol, sport->dport):(item1=value1; ...)
```

図 5.9: 別セッション監視シグネチャの書式

```
sig_type:new($taddr->any, UDP, any->1433):(item1=value1; ...)
```

図 5.10: 別セッション監視シグネチャの記述例

さらに，トラフィックのパターンを記述するシグネチャにおいて指定できる項目を表 5.3に示す．これらを 1 つ，あるいは複数指定することで様々なトラフィックのパターンを表現できる．

図 5.11ではペイロードのパターンを記述するための書式を示している．本実装でのペイロードとはレイヤー 5 以上のパケットデータを指している．パターンは “&” で区切ることによって複数指定でき，それぞれに検索開始点 (*start*) と検索終了点 (*end*) をバイト単位で任意に指定できる．ペイロードのパターンは ASCII 文字によって記述しする．ASCII 以外のデータを指定するには “|” で囲い，2 桁の 16 進数によって 1 バイト分のデータとして記述する．

図 5.12にペイロード部分の記述例を示す．図中では payload.nocase 項目を利用しており，アルファベットの大文字，小文字を区別せずに検知するよう指定している．1 つめのパターンでは “GET” という文字列と，16 進数で 20 と表される 1 バイト分のデータ (空白スペース) が先頭から 4 バイト目までの間に含まれるかを検査する．加えて 2 つめのパターンで “root.exe” という文字列がペイロードに含まれるか検査する．この両方が検知されることで当該項目が成立したと見なされる．

また図 5.13では libpcre[35] を用いた正規表現によるペイロードパターンの例を示している．“/” で囲まれている部分が正規表現の記述であり，右端にある “i” はアルファベットの大文字，小文字の区別をしないオプションを指定している．

5.6.5 ルールの記述例

ここでは root.exe バックドアへの侵入を検知するためのシグネチャを例として図 5.14に示す．図 5.14の例ではトリガーシグネチャとなる 2 行目で “root.exe?/c+dir” を含む TCP の http(ポート 80 番) へのトラフィックを検知する．その後，攻撃対象からの反応に HTTP のファイル取得成功コードの 200 を含む，‘‘HTTP/\d.\d 200 OK’’ の文字列

```
payload = [start:end,] "pattern1" & [start:end,] "pattern2...;
```

図 5.11: ペイロードパターンの書式

```
payload.nocase = 0:4, "GET|20|" & "root.exe";
```

図 5.12: ペイロードパターンの記述例

```
payload.pcre = "/GET[ ]+\S*root\.exe[ ]+HTTP\/\d.\d/i"
```

図 5.13: 正規表現によるペイロードパターンの記述例

が発見されれば，攻撃が成功したと判断する．4 行目の失敗を示すシグネチャにはペイロード部の指定に‘‘HTTP/\d.\d \d\d\d’’ とあり，\d は任意の ASCII 数字を示すため，HTTP の応答コードが 200 以外だったときに失敗と判断するトラッキングシグネチャとなっている．また，HTTP の応答コードが含まれない異常な応答や，応答が 300 秒以上なかった場合には未知の状況として検知するよう，5 行目，6 行目で指定している．

実際の実装では 5 行目，6 行目のシグネチャは多くのルールで同様の記述が必要となるため，S-NIDS 全体で初期値を設定できる仕様となっている．

5.6.6 トリガーシグネチャ検索

トリガーシグネチャは，常に同様のデータ群とパケットを比較し検索するため，ハッシュテーブルを使用する．ハッシュ検索に利用するキーを TCP,UDP ではポート番号，ICMP では ICMP コード，それ以外の IP 通信ではプロトコル番号とする．ハッシュの衝突が発生した場合は，リストとして保存していく．

```
1| alert ("root.exe probe", TCP, any->80) {
2|   trg:fwd:(payload="root.exe?/c+dir";) {
3|     exp:bak:(payload = 0:14, "HTTP/\d.\d 200 OK";);
4|     fail:bak:(payload = 0:11, "HTTP/\d.\d \d\d\d";);
5|     ukwn:bak:(ext.packet = any_pl;);
6|     ukwn:bak:(ext.timeout = 300;);
7|   }
8| };
```

図 5.14: root.exe バックドアを用いた攻撃のシグネチャ例

表 5.3: シグネチャ設定項目

シンボル	指定内容	説明
ip.saddr	IP アドレス	IP ソースアドレス (個別アドレス / アドレス空間)
ip.daddr	IP アドレス	IP デスティネーションアドレス (個別アドレス / アドレス空間)
ip.len	数値 (0〜2^{16})	IP パケット長
ip.ttl	数値 (0〜2^{8})	生存時間 (Time To Live)
tcp.seq	数値 (0〜2^{32})	TCP シーケンス番号
tcp.ack	数値 (0〜2^{32})	TCP 応答番号
tcp.win	数値 (0〜2^{16})	TCP ウィンドウサイズ
tcp.flags	別途説明	TCP フラグ
icmp.type	数値 (0〜2^{8})	ICMP 種別
icmp.code	数値 (0〜2^{8})	ICMP コード
payload.len	数値 (0〜2^{16})	ペイロード長
payload	別途説明	ペイロードのパターン
payload.nocase	別途説明	アルファベットの大文字，小文字を区別しないペイロードの
payload.pcre	別途説明	libpcre を用いた正規表現によるペイロードのパターン
ext.timeout	数値	トラッキングシグネチャがタイムアウトするまでの秒数
ext.packet	指定ワード	any:あらゆるパケット．any_pl:ペイロードを含むあらゆるパケット

5.6.7 セッション，トラッキングシグネチャ管理

セッション，トラッキングシグネチャ管理の実装について，図 5.15に概要を示す．

各セッション，トラッキングシグネチャは 1 つのフローに属するため，フローを検索した後，セッション，トラッキングシグネチャを検索するよう実装した．フローの検索は，取得したパケットの IP アドレスのハッシュによって行われる．ハッシュテーブルのスケールは監視するネットワークの規模と設定値によって異なる．ハッシュの競合に対処するため，そのハッシュ値に一致するフローをグループとして管理する．グループは連結リストによって保存され，検索する場合はこれを順番に検証していく．

発見された当該フローには，それぞれセッションのリスト，トラッキングシグネチャのリストが関連付されている．双方ともリストを順番に検査し，該当するものを検索する．セッションは発見されなかった場合，新しいものを追加する．トラッキングシグネチャの検索では，当該セッションのトラッキングシグネチャであるかを検査し，必要に応じて検索をスキップする．該当するトラッキングシグネチャが発見された場合

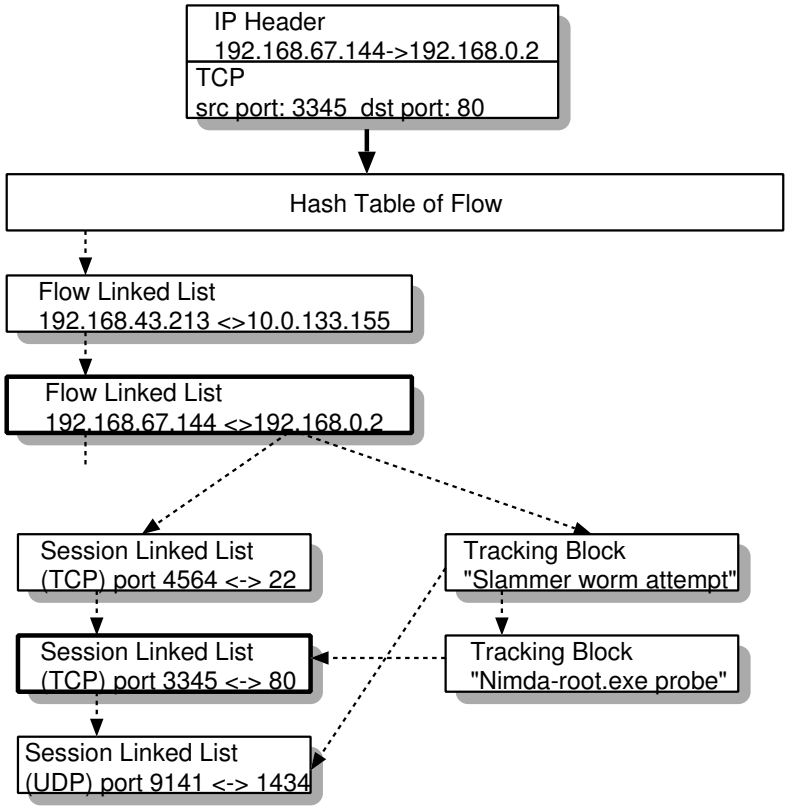

図 5.15: セッション，トラッキングシグネチャ検索動作概要

は，セキュリティイベントを記録し登録を削除する．

5.7 評価

幾つかのシグネチャを作成し，表 3.2と同様の環境において実験を行った．その結果をもとに，S-NIDS の評価を行う．評価は，実際にリスクを分類し運用コストの削減を実現できているか，分類した結果が正しいかについての定量評価と，他の手法と比較した定性評価を行った．

5.7.1 リスク評価

実際に取得したログから，応答の監視によってリスク評価が可能となることを検証する．

評価は実運用ネットワークを 2004/01/12 03:30:00 から 2004/01/16 20:20:00 まで監視することで行った．図 5.16，図 5.17，に評価に用いたシグネチャを示し，その結果を表 5.4，表 5.5にそれぞれ示す．

```
alert ("CodeRed II Worm", TCP, any->80) {
  trg:fwd:(payload = "\\root.exe";) {
    exp:bak !(payload = "404" | "403";);
  }
};
```

図 5.16: CodeRed II ワームのシグネチャ

表 5.4: CodeRed II ワームの検出結果

種別	件数	割合
検知数	571 件	
失敗判定数	544 件	95.271%
成功判定数	27 件	4.729%

以上の通り，攻撃されたホストの応答を監視することで危険性の高いセキュリティイベントを識別することを可能とした．表 5.4，表 5.5では共に 95%以上の失敗したと考えられる低リスクの攻撃と，5%以下の攻撃が成功した可能性のある攻撃に分類している．

また，トリガーシグネチャで検知した内容が誤検知である可能性を考慮した場合も，成功判定されたセキュリティイベントに対してのみ検知内容が誤検知であるかの調査を行えば良く，結果として運用コストは削減されていると言える．

5.7.2 検出精度

第 5.7.1項で使用した CodeRedII ワームのデータをもとに，検出精度を評価する．まず，失敗と判定された攻撃についての内訳を表 5.6に示す．

```
alert ("netbios_netbios-name-query", UDP, any->137) {
  trg:fwd:(payload="CKAAAAAAAAAAAAAAAAAAAAAAAAAAAA|00 00|";){
    exp:bak:(payload="CKAAAAAAAAAAAAAAAAAAAAAAAAAAAA|00 00|";);
    ukwn:bak:(ext.packet = any_pl;);
    fail:bak:(ext.timeout = 60);
  }
};
```

図 5.17: Windows netbios name 検索のシグネチャ

表 5.5: Windows netbios name 検索の検知結果

種別	メッセージ	件数	割合
検知数	59764 件		
失敗判定数	56838 件	95.104%	
成功判定数	2926 件	4.896%	

表 5.6: 失敗判別された攻撃の内容

応答	件数
404 Not Found	339
403 Forbidden	7
Timeout	198

404 と 403 のエラーメッセージによる失敗判定は，シグネチャの通りの挙動であるため，S-NIDS が正常に稼働していることを示している．Timeout はトラッキングシグネチャの時間切れ処理がなされたものである．トラッキングシグネチャの保持条件はペイロードを含むパケットが来るまで監視するものと設定されていた．そのため TCP セッションが異常終了，あるいは能動的な終了をする際にペイロードを含まない，RST や FIN によって TCP セッションの中断が行われており，パケット数のカウントが行われなかった．これは観測したネットワークが研究用ネットワークであったために，アドレスでアクセスを制限している，あるいは HTTP ではないプロトコルを使用していた，などの理由が考えられるが，S-NIDS の動作としては正しいと言える．

次に表 5.7で攻撃が成功したものの内訳を示す．HTTP のエラーコードである 400, 411 については明らかに攻撃失敗であることが分かる．301 は同一のホストに対する攻撃で，あらゆるリクエストに対して 301 を返すという設定になっているホストであるため，特にリスクは無いと判断できる．リスクとして考えられるのはコード 200 を応答しているホストだが，このホストでは全てのリクエストに対してコード 200 を返すという，RFC に準拠していない Web サーバが稼働していることを確認した．本来 NIDS は，ネットワーク毎の特異な状況を管理者が考慮し，それに応じた設定をしたうえで運用するため，このような例外は許容範囲であると考えられる．本ネットワークで運

表 5.7: 成功判別された攻撃の内容

応答	件数
400 BadRequest	8
200 Ok	5
301 Moved Permanently	12
411 Length Required	2

表 5.8: 定性評価

	従来の NIDS	継続的な通信の記録	TIDS	S-IDS
即時性	△	×	○	○
確実性	○	△	○	○
規模性	○	×	△	○
多様性	×	△	△	△

用する際は，このようなホストを *ignore* ルールを用いて無視する事で，さらにリスクの低いセキュリティイベントを取り除くことができる．

成功判定がでたセキュリティイベントには，全てに危険性が無かったため，これらは誤検知であると言える．表 5.7の結果をもとに，400，301，411 も失敗判定できるようシグネチャを変更することで，さらに誤検知を減らすことができる．

5.7.3 定性評価

S-NIDS を他のリスク評価手法と比較し，S-NIDS の定性評価を行う．第 5.3節で挙げている要件の４つを評価項目に組み込む．

1. 即時性
2. 確実性
3. 規模性
4. 多様性

1は速やかにセキュリティイベントのリスクを評価できるか，2はセキュリティイベントの評価をした結果が正しいと言えるかどうか，3は多数の内部ホストで構成されるネットワークにおいても運用のための手間が変わらないか，4は様々な攻撃に対応できるか，の以上４つを定性評価項目とする．また，比較する手法は従来の NIDS，第 5.2節で挙げている継続的な通信の記録，そして TIDS[27] の 3 つである．従来の NIDS は運用が効率化されている事を示すために挙げる．継続的な通信の記録は Enhanced NIDS に属する別の実装として挙げる．そして，TIDS は自動的なリスク評価の一手法として比較する．以下で評価について述べ，表 5.8においてその一覧をまとめる．

即時性

TIDS，S-NIDS はそれぞれ，NIDS の検知エンジンやセキュリティイベントの解析エンジンが自動的にリスクを評価する．これにより，各ソフトウェアが即座にリスク評価の情報を記録に残す，あるいは管理者への通知が可能となり，早急なインシデント

対応ができる．しかし，攻撃検知後に継続して通信を記録しても，リスク評価は管理者が 1 つずつセキュリティイベントを検査しなければならず，多くの手間がかかってしまう．また，従来の NIDS ではウィルスの感染活動は即時的にリスクが高いセキュリティイベントを検知できるものの，不正侵入については即時的に判断できないなどの制約がある．

確実性

TIDS はネットワークの全ホストについての完全な情報を取得することで，確実なリスク評価ができる．また，内部からの攻撃検知と S-NIDS はシグネチャに定められた通信のパターンを確実に検知する．そのため，適切なシグネチャが設定されていればリスク評価の確実性は高いと言える．しかし，攻撃検知後に継続して通信を記録しても，管理者が多くの通信プロトコルについて熟練した知識を持っていなければリスク評価は困難となる．また，人間が記録を検査することでリスク評価を行っているため，誤判断を起こす可能性が有る，誤判断は調査するセキュリティイベントの数に比例して多くなる場合が多いと考えられる．

規模性

TIDS はリスク評価を行うために，全ての内部ホストについてホスト情報を把握する必要がある．したがって，ネットワークの規模拡大に伴いホスト情報の把握が困難になり，リスク評価の弊害となる．攻撃検知後の継続的な通信の記録も，管理者が自らリスク評価をしなければならず，ホスト数の増加はセキュリティイベントの増加につながり，大幅に手間が増えてしまう．内部からの攻撃検知と S-NIDS はホスト数が増えても，常に同様のリスク評価の処理を行うため規模性について問題はない．

多様性

従来の NIDS は送信元が内部である不正侵入の試みや調査行為など，ごく一部の攻撃に対してのみリスク評価が有効である．そのため外部から内部への不正侵入や調査活動の評価ができない．TIDS はバッファオーバーフローによる不正侵入の試みについてはリスク評価が可能だが，OS やアプリケーションに依存しない，攻撃者による調査活動には対応できない．また，S-NIDS は一部の応答が期待できない攻撃については未対応となっている．攻撃検知後の継続的な通信の記録は，どのような攻撃に対しても記録が可能である．しかし S-NIDS と同様に，攻撃後に通信が発生しない攻撃も存在するため，これについてはリスク評価の手がかりを得ることができない．

5.8 今後の課題と展望

本節では S-NIDS の特性によって引き起こされる問題点と，S-NIDS の応用によって実現できる可能性について論じる．

5.8.1 保持情報量の増加

S-NIDS は検知を行うために，フロー，セッション，トラッキングシグネチャの情報を，それぞれ記憶しておかなければならない．そのため，S-NIDS は従来の NIDS と比較すると保持すべき情報量が増加する．また，情報量の増加は処理負担も増加させる可能性がある．これにより，ある条件下では S-NIDS が正常に動作しなくなり，脅威となる攻撃が検知できなくなる可能性がある．例として，特定の偽装したパケットを大量に送信される [36]，あるいは急速に感染を広めるワームの突発的な発生が挙げられる．

これは S-NIDS の構造によるものであり，アルゴリズムによる根本的な解決は困難であるが，今後のハードウェアにおける記憶保持，及び処理機能の改善によりある程度緩和可能であると期待できる．

5.8.2 通信の応答が期待できない攻撃への対策

S-NIDS を用いても，通信の応答が期待できない可能性のある攻撃について成否を判断するのは困難である．具体例として DoS 攻撃や，ステートレスな UDP を利用したウィルスの感染活動が挙げられる．ステートレスな UDP を利用したウィルスの感染活動は，ICMP Port Unreachable Error のパケット返答によって失敗を判定できるが，Personal Firewall が有効なホストでは ICMP を返さない可能性があり，正確に成否を判別できない．

また，応答が無い場合に成功したと判断される攻撃については，Firewall などで遮断されることで誤検知の発生が予想される．一部のホストに対する攻撃で失敗の判定を得られたとしても，特に堅牢に構築されたネットワークほど誤検知が多発し，別個にリスク評価をするコストが発生する可能性が高い．この問題についてはセッションの追跡とは別に，内部ホストの挙動を継続的に監視する，あるいは他の評価手法との併用によって解決できると考える．

5.8.3 プロトコルアノーマリ検知への応用

S-NIDS はシグネチャの記述方法により，プロトコルアノーマリ [37] の検知が可能となる．セッションを追跡しての監視が可能であるため，プロトコル違反のパケットに対して，プロトコル違反の応答がなされた場合，リスクが高いと判断できる．これによって，高リスクの通信を抽出し，シグネチャとして登録されていない攻撃の検出が容易になる．具体例としては，HTTP のリクエストは通常 ASCII コードのみの通信とされているが，この中にアセンブラコードが含まれる場合，攻撃の可能性があると考え，

当該セッションを監視し続ける．その応答に通常のエラーコードが含まれていれば正常な動作であると考えられる．しかし，エラーコードが含まれない，あるいは ASCII 以外の文字コードが含まれるなどの応答を確認することで正常な動作では無いと判断し，高リスクであると評価できる．S-NIDS は未知の攻撃にたいしてもリスク評価ができる可能性が高いため，幅広い応用が期待できる．

5.9 まとめ

本章では，検知したセキュリティイベントについてリスク評価ができない問題を解決するために，攻撃が成功したか，失敗したかによって応答が変化するセキュリティイベントがあることに着目した．そして，攻撃のあったセッションを追跡し，攻撃の検知だけではなく，その応答も監視することで，成功・失敗の判別をする Session based NIDS を提案し，実装した．そして，実ネットワーク上で実験を行った結果，Session based NIDS によって攻撃のリスク判別が行える事実を示した．

第6章　OSの種別を基本としたリスク評価システム

6.1　はじめに

NIDSが検知するセキュリティイベントの中でも，ソフトウェアの脆弱性を利用する不正侵入の試みはOSやアプリケーションの種類，バージョンによってその結果が左右される．例えば，ソフトウェア内のバッファ領域を越えてしまうバグ（バッファオーバーフロー）を利用した攻撃は任意のコードを実行するのに，当該ソフトウェアにおける関数アドレスなどの情報を把握している必要がある [38]．コンパイル時に定められる関数のアドレスなどはOSや構成環境によって変化する可能性があるため，脆弱性を持つバージョンの同一アプリケーションでも，OSによって攻撃手法が異なる．また，同一のプロトコルでサービスを提供するアプリケーションも複数の実装が存在するものが多い．プロトコルの脆弱性でない限り，アプリケーションの種類によってもバグやその脆弱性は全く異なる．そのためアプリケーションが異なれば，同一の攻撃手法を検知したセキュリティイベントでもリスクが異なる．

既存の主なNIDSはソフトウェアの情報をもとにリスクを評価する機能を持たない．本章では各ホストで利用しているOS情報の収集とリスク評価の自動化によって，NIDS運用モデルにおける監視フェイズの運用負担の軽減を目的とする．

本章ではホスト毎のソフトウェア情報としてOSの種類をDHCPのリクエスト情報から収集し，リスク評価に利用するシステムを提案，構築した．これにより，動的に利用ホストが変化するネットワークにおいても管理負担を増やすことなく，正確な情報を取得することに成功した．さらに，NIDSが取得したログのリスクを評価し，本システムの有用性を示した．

6.2　関連研究

内部ネットワークのトポロジ情報や機器の構成情報，各ホストで動作しているアプリケーションの情報などをもとにリスクを判別するシステムは，一般的にTarget-based IDS（TIDS）[27] と呼ばれている．TIDSはNIDSのように自らセキュリティイベントを収集する機能はないが，NIDSによって検知されたセキュリティイベントをネットワーク環境やホスト環境の情報を利用して，リスク評価するシステムである．

TIDSが有効に機能する例を示す．NIDSの検知結果を攪乱する方法の1つとして，IPヘッダに含まれるTTL（Time To Live:生存時間）を書き換える手法がある．TTL

は IP ネットワーク上でのループを防ぐために，ルータを経由するたびにその値を 1 ずつ減らし，0 になった時点でパケットを破棄するためにある．複雑な内部ネットワークトポロジをもつネットワークであれば，あえて短い TTL 値のパケットも攻撃時に送信することで，到達するパケットと途中で破棄されるパケットに分かれる場合がある．しかし，短い TTL のパケットを NIDS が等しく検知対象としてしまえば，NIDS の検知エンジンに無効な情報が渡されてしまい，結果として無効なセキュリティイベントと有効なセキュリティイベントを混在して検知してしまう．TIDS はネットワークのトポロジを正確に把握し，パケットの宛先から TTL が有効であるかどうか，すなわちパケットが到達可能であるかを判断することで，無効なセキュリティイベントを検知しないような処理が可能となる．

また，内部ネットワークで動作している各アプリケーションを事前に調査し，発見したセキュリティイベントと脆弱性情報を照らし合わせ，リスクの高いセキュリティイベントを検知するような実装もある．具体的な実装について，以下で述べる．

6.2.1 RNA sensor

RNA sensor[39] はネットワークトラフィックを監視し，内部ホストが送信するパケットに付随する各種ヘッダなどを分析することで，OS やソフトウェアのバージョンを判定する．この分析手法はパッシブフィンガープリンティング手法と呼ばれている．RNA sensor は得られた OS やソフトウェアバージョンを NIDS である Snort[12] のログと関連づけ，リスクを評価する TIDS である．ただし，内部ホストが何らかの通信をするまではホスト情報の収集ができない．攻撃を受けたホストが応答する種類の攻撃であればそのタイミングでホスト情報を取得できるが，ステートレス型の UDP などを用いた攻撃では応答を返さない可能性もあるため，確実性が保障しにくい．

6.2.2 nCircle

nCircle[40] は，それ自体が内部ネットワークに接続するホストを定期的に検査し，対象ホストの OS やソフトウェアの種類，そしてそれらがもつ脆弱性を調査できる．そのため，ソフトウェアの種類やバージョンよりも詳しい情報にもとづいた関連づけが可能となり，より正確なリスク評価ができる．ただし，定期的な検査によって情報を収集，更新しているため，検査のインターバルが長ければ正確性を欠いてしまう．また，ネットワークに接続するホストが動的に変更する環境であれば，IP アドレスとホスト情報の不一致が発生しやすくなり，正確性に影響を与えてしまう．

6.2.3 ArcSight

最も高機能な SIM の 1 つとして ArcSight[41] が挙げられる．ArcSight は NIDS や Firewall の他に，ルータやスイッチといったネットワーク機器からも情報を収集でき

る．収集した情報を，詳細な記述ができるルールに従って，セキュリティイベントのリスクを判定できる．また過去に発生したセキュリティイベントも評価のためのルールに記述できる．

ArcSight の欠点としては記述すべきルールが複雑化してしまう点である．多様なセキュリティデバイスから情報を収集するため，利用できる NIDS 以外からの情報はそのネットワークの構成によって様々である．また，接続しているホストの種類やネットワーク構成をもとに記述するルールもネットワークによって異なる．従って，ArcSight を有効に活用するためにはネットワーク管理者が自らルールを設定しなければならない．導入時に必要となるルールを多く記述する，あるいは運用状況に合わせてルールを編集することによって，ネットワーク管理者の負担が増加する可能性が高い．

6.3 現状の TIDS における課題

TIDS はホストが利用しているソフトウェア情報にもとづいてセキュリティイベントのリスクを評価するため，適切なソフトウェア情報を保持している必要がある．そのため，ネットワークに接続しているホストのソフトウェア情報を管理，更新する手段の検討が TIDS の課題の 1 つとなっている．

ソフトウェア情報の管理において，もっとも負荷が高くなると予想されるのは接続ホストの状況変化に伴うソフトウェア情報の更新作業である．ネットワークに接続されているホストが固定されていれば，責任者への問い合わせ，あるいは定期的な調査により，ソフトウェア情報の把握が可能となる．

しかし，ネットワークへの接続・切断が頻繁におこるホストがいる場合，ホストの接続状態に伴いソフトウェア情報も変化する．このような動的なホストは，一般的に Dynamic Host Configuration Protocol(DHCP)[42] によって，ホストの識別子である IP アドレスの割り当てを受ける．DHCP は，同一ホストであっても状態に応じて異なる IP アドレスが割り当てられる可能性がある．NIDS のログとの関連づけは基本的に IP アドレスを用いるため，各ホストの IP アドレスを管理者が把握しなければならず，管理者の負担となってしまう．利用ホスト数が増加すれば，管理者が各ホストの情報を管理する作業が困難となってしまう．また，不特定多数のホストが接続するネットワークや大規模ネットワークでは各ホストの情報をその都度取得するのは大幅な負担の増加となってしまう．

6.4 解決手法

本章では DHCP を利用した OS 情報の取得により，動的に接続するホストの OS 情報を正確に取得し，リスクが低いと考えられるセキュリティイベントを確定する手法を提案する．DHCP による OS 情報の取得と，OS 情報に基づいたリスク評価の手法について論じる．

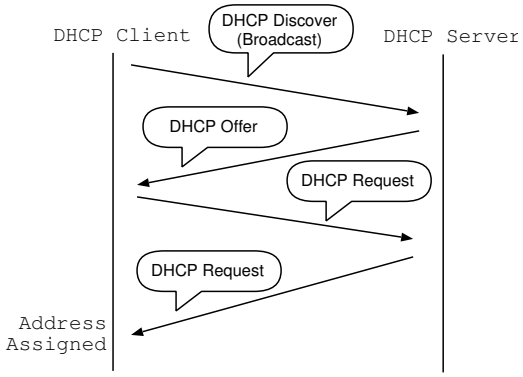

図 6.1: DHCP によるアドレス取得時のメッセージの流れ

6.4.1 DHCP を用いた OS 情報の収集

DHCP はネットワークへ接続したホストが DHCP サーバに問い合わせをし，適切なアドレスを割り当ててもらうプロトコルである．図 6.1に DHCP のアドレス取得時に送信するメッセージの流れを示す．接続当初はどの IP アドレス，あるいは MAC アドレスのホストが DHCP サーバなのかが分からない．そのため，クライアントとなるホストは接続後にブロードキャストによって DHCP サーバを探し出す DHCP ディスカバーと呼ばれるメッセージを送信する．これは同一のブロードキャストセグメントに接続している全ホストへ送信されるため，同一セグメントのトラフィックを監視していれば動的に接続するホストが送信する DHCP ディスカバーを確実に取得できる．

DHCP ディスカバーメッセージにはアドレスを取得するために，送信ホストの MAC アドレスやメッセージ識別のためのトランザクション ID などが含まれるが，その中に OS 毎の固有情報が含まれている．この特性を利用することで同一セグメント内のホストが使用している OS を特定できる [43]．本章では DHCP による OS 識別技術を利用し，NIDS が検知したセキュリティイベントのリスクを評価する手法を提案する．

6.4.2 OS 情報を用いた攻撃の効果推測

ホストが利用している OS の種類からリスクを評価できるセキュリティイベントは，主に 4 つに分類できる．

OS が持つ脆弱性の利用　1 つめは OS そのものが持つ脆弱性である．各 OS のネットワークスタックや提供している API の脆弱性を利用した不正侵入の試みは特定の OS のみにしか効果が無い．特に，様々な機能が OS に付随している Windows では画像処理における脆弱性 [44] をはじめとする様々な OS 固有の脆弱性がある．

特定の OS のみで動作する悪意のあるプログラム　悪意のあるプログラムであるコンピュータウィルスやマルウェアは OS 毎の実行形式となっているものがある．このよう

なプログラムは別の OS では実行形式が異なるため実行できない．実行不能である OS から悪意のあるプログラムの活動が検知されたとしても，誤検知であると判断できる．

特定の OS 上で動作するアプリケーション脆弱性の利用　複数の OS 上で動作するアプリケーションであったとしても，バッファオーバーフロー脆弱性のように利用にあたって OS の種類や CPU アーキテクチャに依存する不正侵入の試みがある．目標ではない OS 上で動作しているアプリケーションに不正侵入の試みた場合，侵入される可能性は低い．

特定の OS のみで動作するアプリケーションの脆弱性の利用　特定の OS においてのみ提供されているアプリケーションの脆弱性を利用した不正侵入の試みであれば，他の OS で動作するホストでは被害が発生しないと考えられる．

以上のセキュリティイベントについてはセキュリティイベントの送信元ホストや送信先ホストの OS 情報に基づき，リスクを判断できると言える．本章ではこのような特性に着目し，リスク評価システムを構築する．

6.5　設計

第 6.4節で述べた解決手法を実現するためのシステム構成，およびシステムに必要となる要素について述べる．各ホストの OS 情報に基づいたリスク評価システムの全体的な構成を図 6.2に示す．第 6.5.1項，第 6.5.2項，第 6.5.3項において，各部の詳細を述べる．

6.5.1　OS 情報の取得

内部ネットワークのセグメントには DHCP ディスカバーメッセージを受信し，動的に接続するホストの OS を判別する DHCP gazer を設置する．DHCP gazer は DHCP を利用し，OS を判別するパッシブフィンガープリンティングツールであり，[43] において実装されている，内部ネットワークに接続するホストは DHCP ディスカバーメッセージをブロードキャストパケットとして送信するため，接続するたびに DHCP gazer が OS の種別を判断し，その結果をデータベースに蓄積する．

ネットワークとの接続・切断を繰り返さない固定ホストについては頻繁に更新される可能性は低いため，定期的な管理者の調査によって各ホストの情報を正確なものに保てると考えられる．

本章では DHCP のメッセージによって OS の種類を判別しているが，今後さらに別の手法を用いて拡張する可能性がある．その場合は，DHCP gazer が利用している OS 情報を格納したデータベースを適宜更新すれば，本システムは透過的に OS の情報を扱うことができる．

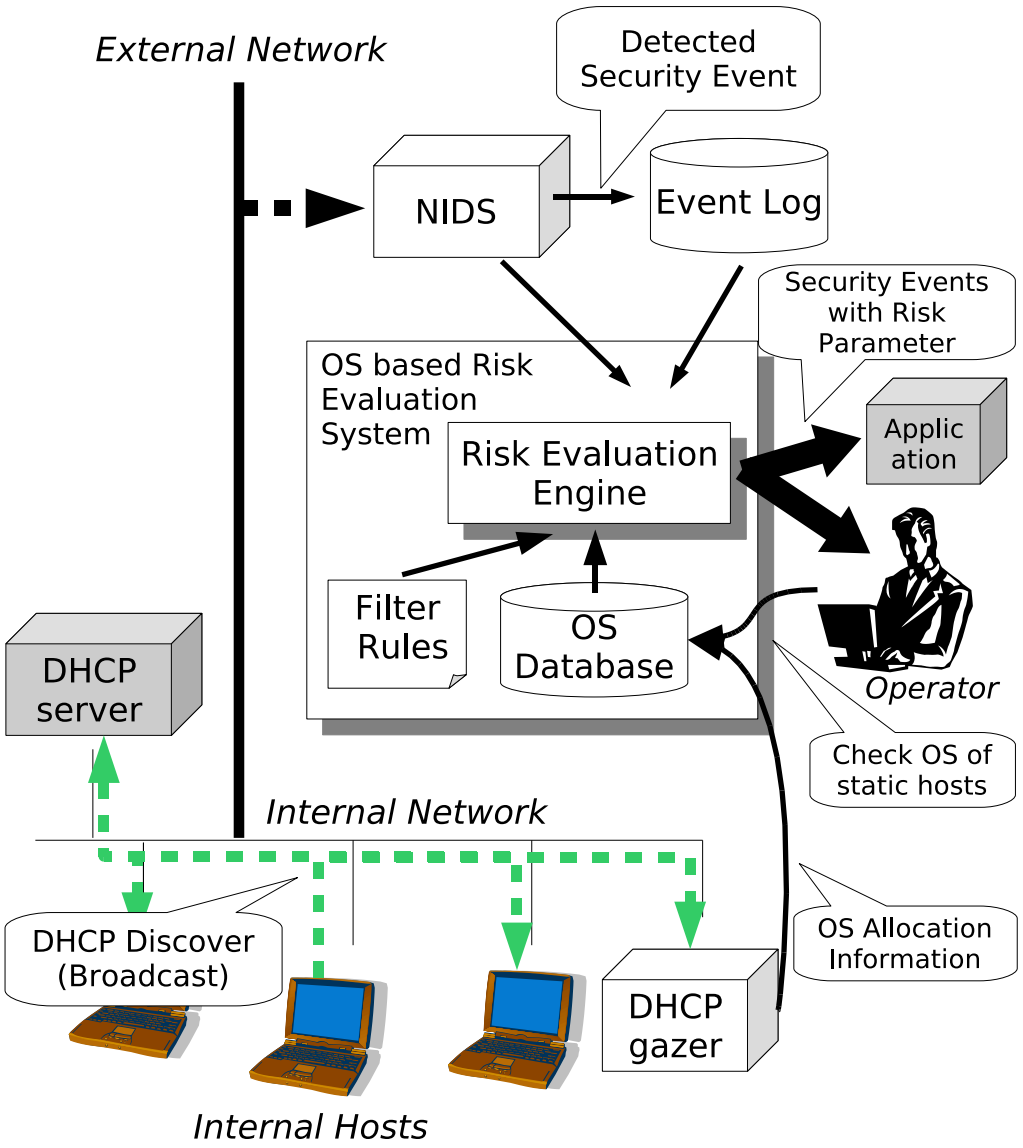

図 6.2: システム概要図

6.5.2 リスク評価ルール

OS 情報にもとづいたリスク評価には，どのセキュリティイベントがどの OS を利用しているホストに対してリスクが高いか，あるいはリスクが低いかを記述したルールが必要となる．リスクを評価すべきセキュリティイベントは多数存在するため，ネットワーク管理者は評価ルールを管理する必要がある．特に NIDS でセキュリティイベント検知に用いるシグネチャは，ネットワークの特性に応じて管理者が独自のルールを記述する場合が少なくない．そのため，テキストファイルなどによって容易に管理，編集が可能なルール情報を用いる必要がある．

ルールには，ある種類のセキュリティイベントについて送信元ホストおよび宛先ホストの OS，そしてリスク度合いの 3 項目が指定可能である必要がある．

6.5.3 リスク評価の実行

本システムは DHCP gazer により取得した各ホストが利用している OS の情報，NIDS が検知したセキュリティイベント，そしてリスク評価ルールの 3 つを用いてリスクを評価する．

NIDS で取得されたセキュリティイベントを随時リスク評価する仕様も必要だが，各ホストの環境変化が後になって判明した場合は，リスクの再評価が必要となり評価結果に不整合が生じる場合がある．よって，ホストの OS 情報を修正した後に，蓄積されたセキュリティイベントログを用いてリスク評価する要求も考えられる．そのため，NIDS で検知されたセキュリティイベント情報を直接受け取る方法と，NIDS によって蓄積されたログからセキュリティイベント情報を受け取る方法の 2 通りで動作する必要がある．

6.6 実装

第 6.5節で述べた設計を元に，OS based Risk Evaluation System(OSRES) を実装した．第 6.5.3項で述べた，即時的なセキュリティイベントのリスク評価と蓄積されたセキュリティイベントログからのリスク評価の両方を実現するために，本論文ではリスク評価のライブラリとして実装し，各種アプリケーションに対応させる手法を採った．実装には C 言語を用い，情報を蓄積するためのデータベースとしてフリーのリレーショナルデータベース実装である MySQL[45] を利用した．

6.6.1 システム構成

システムの全体的な実装概要を図 6.3に示す．本システムは定期的な調査による固定ホストの OS 情報と，DHCP gaezr を用いて取得した動的なホストの OS 情報を利用している．

リスク評価機構は libosres ライブラリとして実装し，各種アプリケーションが必要に応じてセキュリティイベントのリスクを判別できるようにしている．本ライブラリを NIDS が利用すれば，セキュリティイベントを検知した時点で，即時的にリスクを判定できる．ログとして蓄積されたセキュリティイベントを表示するインターフェースアプリケーションで利用すれば，任意のタイミングでセキュリティイベントをリスク評価できる．また，本論文ではホストの OS 情報を蓄積している MySQL データベースに，OS 情報を入力できるよう DHCP gazer に機能を追加した．

その他の詳細については，以下で述べる．

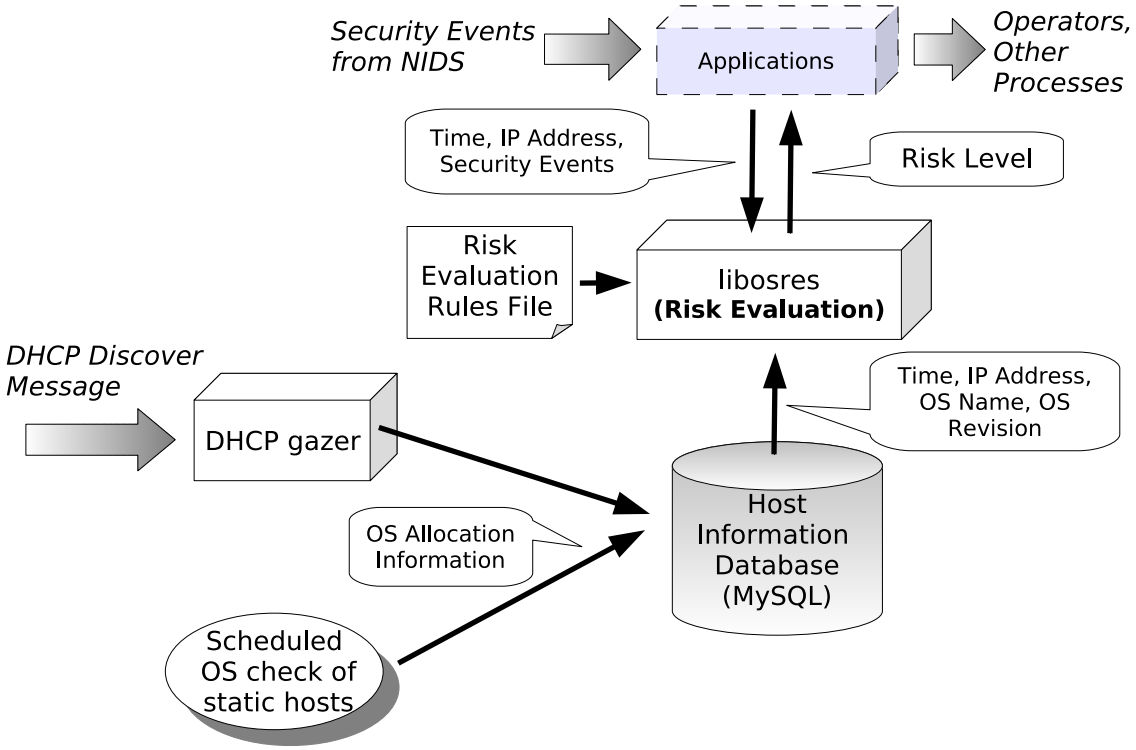

図 6.3: 実装概要図

signature_name, *src_os*: *src_rev*-> *dst_os*: *dst_rev*, *risk*

図 6.4: リスク評価用ルールの書式

6.6.2 リスク評価用ルール

OS 情報をもとにしたリスク評価で使用するルールを図 6.4 に示す．libosres が使用するルールは編集や管理の容易さからテキスト形式で記述している．セキュリティイベントの種類判別は各ルールに記述されたシグネチャ名を用いる．NIDS は，検知した内容が同一の事象であったとしても，実装によってセキュリティイベントの名称は異なる場合が多い．そのため，NIDS 毎にルールを用意する必要があり，アプリケーション側で適宜使用するルールファイルを変更すれば，様々な実装に対応できる．

図 6.4中の *src_os*，*src_rev* にはそれぞれセキュリティイベントの送信元ホストの OS 名とそのリビジョン，*dst_os*，*dst_rev* には送信先ホストの OS 名とそのリビジョンを記述する．現在の実装において指定できる OS 名と種類を表 6.1に示す．任意の OS を指定した場合は*(アスタリスク) を記述することで，全ての OS やリビジョンが一致する．

```
MS-SQL version overflow attempt, *:* -> Windows:*, R
MS-SQL version overflow attempt, *:* -> *:*, S
```

図 6.5: Slammer worm を対象としたリスク評価用ルールの記述例

最後に *risk* では，そのセキュリティイベントや OS の組み合わせの場合にリスクが高い，あるいは低いことを指定する．リスクが高い場合は “R” (Risky) を，低い場合には “S” (Safety) を，不明な場合には “U” (Unknown) を指定する．ルールは上から順番に検査されるため，明示的にリスク不明を示したい場合は “U” を記述する．全てのルールに一致しないセキュリティイベントは自動的に Uknown，すなわちリスク不明と判断される．

図 6.5に評価用ルールの記述例を示す．図中では，Snort[12] が Slammer worm[1] を検知したセキュリティイベントのリスクを評価するルールを記述している．Slammer worm が感染する MSSQL[46] は Windows OS 上でのみ動作する SQL サーバのアプリケーションである．そのため，Slammer worm が送信するエクスプロイトコードは Windows 以外の OS に対して無効であり，リスクが低いと判断できる．逆に，Windows を使用しているホストにエクスプロイトコードが送信されれば，他の OS に送信されるより感染する可能性が高いと言える．

図 6.5は，Slammer worm が Windows のホストに対してのみ有効である，という事実をもとにして記述されているルールである．1 行目はエクスプロイトコードが送信したホストが Windows を利用していればリスクが高いと判断するルールである．判別すべきセキュリティイベントの送信先ホストが，Windows を利用している OS だった場合は，1 行目で評価処理を停止する．2 行目で送信先ホストに任意の OS を指定することで，Windows 以外の OS はリスクが低い，という設定を記述している．

6.6.3 libosres

libosres はアプリケーション側から渡されるセキュリティイベントの情報と，データベースを用いて取得する各ホストの OS 情報，そして評価用ルールに基づきリスクを判別する．libosres を使用するための基本的な流れは以下の通りである．

1. ライブラリの初期化
2. ルールファイルの指定
3. ホスト情報データベースへの接続パラメータの設定
4. 評価処理の準備
5. 評価処理関数の呼び出し

表 6.1: リスク評価用ルールで指定できる OS 名

OS 名	リビジョンの種類	説明
Windows	98,Me,2000, XP,2003,*	Microsoft Windows[4]
Mac OS X	8,9,10.0.X,10.1.X, 10.2.X,10.3.X,*	Apple Macintosh[47]
Linux	2.2,2.4,2.6,*	Linux[48]
FreeBSD	4.X,5.X,6.X,*	FreeBSD[49]
NetBSD	2.0,*	NetBSD[50]
Sun Solaris	8,*	Solaris[51]
3Com embedded	*	3Com 社製製品のネットワーク接続機器
Bay Networks embedded	*	Bay Network 社製製品のネットワーク接続機器
Cisco embedded	*	Cisco Systems[52] 社製製品のネットワーク接続機器
Cisco IOS	12.X,*	Cisco Systems[52] 社のネットワーク接続機器で使用されている OS
Cisco IP Phone	CP-7920,CP-7940,CP-7960,CP-7940G,CP-7960,*	Cisco IP Phones[53]
D-Link embedded	*	D-Link 社製製品のネットワーク接続機器
Dell embedded	*	Dell 社製製品のネットワーク接続機器
Enterasys embedded	E9.0.7.0,*	Enterasys 社製製品のネットワーク接続機器
Extremeware	6.2.2,*	Extreme 社製製品のネットワーク接続機器で使用されている OS
Foundry embedded	*	Foundry 社製製品のネットワーク接続機器
Foundry IronWare	*	Foundry 社製製品のネットワーク接続機器で使用されている OS
Hitachi embedded	*	Hitachi 社製製品のネットワーク接続機器
HP embedded	*	HP 社製製品のネットワーク接続機器
Juniper JUNOS	*	Juniper 社製製品のネットワーク接続機器
Megras embedded	*	東京特殊無線社製 RFID リーダ Megras[54]
Motorola VxWorks	*	
Netscreen ScreenOS	*	Netscreen[29] で使用されている OS
NIB embedded	*	
Okidata embedded	*	OKI データ社製製品のネットワーク接続機器
Polycom embedded	*	Polycom 社 [55] 製テレビ会議システム
RCA embedded	*	
Tandem Tandem NSK	*	
Xerox embedded	*	Xerox 社製製品のネットワーク接続機器

6. 終了関数の呼び出し

具体的な libosres の利用方法については付録 Aで説明する．

6.7 評価

OSRES の有用性を示すために，リスク評価によって判別された低リスクセキュリティイベントが全体に占める割合を示す．

NIDS が検知したセキュリティイベントを本実装によって評価した結果を表 6.3に示す．表ではセキュリティイベントの種類，検知数，本実装によってリスクが低いと判断されたセキュリティイベント数，そしてリスクが低いと判断されたセキュリティイベントが占める割合となっている．セキュリティイベント検知のための環境は第 3.2.1項で述べたものと同様であり，セキュリティイベントは 2005 年 12 月 23 日 0 時 0 分から同日 23 時 59 分までの 24 時間となっている．表 6.2には内部ネットワークに接続しているホスト数を OS 別に分類している．評価にはリスクを判断できるセキュリティイベントを事例としてとりあげた．

低リスクと評価されたセキュリティイベントの割合はセキュリティイベントによって様々であるが，各項目は平均して 50%以上がリスクの低いセキュリティイベントであると判断された．また，表 6.3の最下段にはここに挙げたセキュリティイベント数の合計，リスクの低いセキュリティイベント数の合計，リスクの低いセキュリティイベントが全体に占める割合を示している．リスクが低いセキュリティイベントは全体の 78.79 パーセントであり，当該セキュリティイベントについては解析すべき件数が従来の 5 分の 1 に削減された．この事実から，本システムは実ネットワーク環境においてセキュリティイベントのリスクを判断し，解析すべきセキュリティイベントの削減を実現していると言える．

6.8 今後の課題と展望

OSRES が持つ問題点と，今後の発展に必要となる要素について述べる．

6.8.1 仮想 OS への対応

本章では，動的ホストについても即時的に使用している OS を判別し，リスク評価に利用するシステムを構築した．DHCP での OS 判定率は高いが，各ホスト上で仮想 OS を利用している可能性に留意しなければならない．仮想 OS とは，ホストで動作している OS 上で仮想的に動作している OS である．実際にホストが使用している OS とは異なる種類である場合がほとんどであり，仮想 OS が外部と通信していればそのホストに対するセキュリティイベントのリスクは，本実装が判定した結果とは異なる可能性がある．同一ホスト上から 2 種類以上の OS のトラフィックが流れる場合，ネットワーク

表 6.2: 評価期間に内部ネットワークへ接続していたホストの OS 種別による分類

OS 種別	台数
3Com embedded	1
Bay Networks embedded	4
Cisco embedded	11
Cisco IOS	2
Cisco IOS (12.X)	9
Cisco IP Phone (CP-7920)	1
Cisco IP Phone (CP-7940)	8
Cisco IP.Phone (CP-7940G)	4
Cisco IP.Phone (CP-7960)	7
Cisco VoIP Phone	1
D-Link embedded	1
Dell embedded	1
Enterasys embedded (E9.0.7.0)	1
Enterasys embedded	1
Extremeware(6.2.2)	1
Foundry embedded	5
Foundry IronWare	5
FreeBSD	60
FreeBSD (4.X)	17
FreeBSD (5.X)	32
Hitachi embedded	1
HP embedded	1
Juniper JUNOS	1
Linux	57
Linux(2.4)	3
Linux(2.6)	1
Mac OS X	20
Mac OS X (10.3.X)	5
Megras embedded	1
Motorola VxWorks	1
NetBSD	23
NetBSD (2.0)	5
Netscreen ScreenOS	1
NIB embedded	1
Okidata embedded	1
Polycom embedded	5
RCA embedded	1
Sun Solaris	1
Sun Solaris(8)	3
Tandem Tandem NSK	1
Unknown OS	53
Windows	7
Windows (2003)	1
Windows (XP)	146
Xerox embedded	2

経由でホストが使用している OS や仮想 OS を正確に判定するのが難しい．ゆえに，正確にセキュリティイベントのリスク評価をするためには，仮想 OS の利用をネットワーク管理者が把握している必要がある．

6.8.2 アプリケーション情報によるリスク評価への対応

本実装では OS 情報を利用したセキュリティイベントのリスク評価を実現しているが，他にもアプリケーションの種類や，アプリケーションのバージョン情報を利用してリスク評価ができると考えられる．アプリケーションの種類やバージョンによっても成功する攻撃の種類が限定される場合が多い．現在は OS 情報のみを利用しているためリスク評価できるセキュリティイベントの種類は限定されているが，アプリケーション情報の利用によりその幅を広げられると考えられる．

表 6.3: セキュリティイベントのリスク判別結果

セキュリティイベント名	(a)	(b)	(c)
(http_inspect) IIS UNICODE CODEPOINT ENCODING	67	1	1.49%
Attempted Sun RPC high port access	4	4	100.00%
Attempted Sun RPC high port access	425	421	99.06%
BLEEDING-EDGE EXPLOIT MS04-007 Kill-Bill ASN1 exploit attempt	21	8	38.10%
CISCO IOS exploit tool (shadowchode)	3	3	100.00%
MS-SQL probe response overflow attempt	42	5	11.90%
MS-SQL version overflow attempt	12419	10510	84.63%
NETBIOS DCERPC IActivation little endian bind attempt	49	8	16.33%
NETBIOS DCERPC ISystemActivator bind attempt	68	12	17.65%
NETBIOS DCERPC Remote Activation bind attempt	49	8	16.33%
NETBIOS SMB Session Setup AndX request unicode username overflow attempt	132	39	29.55%
NETBIOS SMB Session Setup NTMLSSP unicode asn1 overflow attempt	188	144	76.60%
NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit attempt	8	2	25.00%
NETBIOS SMB-DS repeated logon failure	420	420	100.00%
NETBIOS SMB-DS Session Setup AndX request unicode username overflow attempt	2210	883	39.95%
NETBIOS SMB-DS Session Setup NTMLSSP unicode asn1 overflow attempt	443	341	76.98%
RPC DCOM exploit/ Blaster Worm Attack	8	1	12.50%
WEB-IIS view source via translate header	22	14	63.64%
WIN-RPC ISystemActivator Interface Detected (135/tcp)	69	13	18.84%
WIN-RPC Messenger service (messenger) Interface Detected (high port udp)	5220	4458	85.40%
WIN-RPC Plug and Play service (services.exe) Interface Detected (445/tcp)	54	11	20.37%
WIN-RPC Remote Management Interface Detected (135/tcp)	40	7	17.50%
WIN-RPC SAM access (lsass.exe) Interface Detected (139/tcp)	6	2	33.33%
WIN-RPC SAM access (lsass.exe) Interface Detected (445/tcp)	20	9	45.00%
WIN-RPC Server service (lsass.exe) Interface Detected (139/tcp)	8	6	75.00%
WIN-RPC Workstation service (lsass.exe) Interface Detected (139/tcp)	1	1	100.00%
合計	21996	17331	78.79%

(a): 検知されたセキュリティイベントの件数
(b): リスクが低いと判断されたセキュリティイベント数
(c): 検知数に対して，リスクの低いセキュリティイベントが占める割合

6.9 まとめ

本章では，NIDS が検知した不正侵入の試みが目標となるホストの OS によってリスクが異なる点に着目し，OS 情報を利用したセキュリティイベントの評価システムを提案，設計した．動的なホストの OS 情報を収集するために DHCP gazer を利用し，様々なアプリケーションからセキュリティイベントの評価ができるライブラリ libosres を実装した．そして，実ネットワークで検知されたセキュリティイベントと内部ホストの OS 情報をもとにセキュリティイベントのリスクを評価し，解析すべきセキュリティイベント数が削減できることを示した．

第7章　セキュリティイベント・ログの可視化手法

7.1　はじめに

NIDS によって出力される多数のセキュリティイベントはログとして蓄積され，参照する際にはさらに膨大な数量となる．膨大なセキュリティイベントを逐次的に調査し，発生状況を把握するのは負担が多く，迅速な把握が難しい．本章では，ログ情報を要約し視覚的に表現することによって，ネットワーク管理者がセキュリティイベント発生状況を迅速に把握できるようになることを目的とする．

7.2　関連技術・研究

本節ではセキュリティイベントのログ視覚化にあたり，一般的な視覚化手法，および関連研究の特徴と問題点について言及する．

7.2.1　一般的な情報視覚化手法

分野に関わらず多数の情報を要約する代表的な手法としてグラフ化が挙げられる．いくつかの実装 [23, 56] では，セキュリティイベントの発生件数をグラフとして表示する．しかし，セキュリティイベント発生状況を把握するためには，十分な手段であるとは言い難い．具体的な問題点について以下で述べる．

円グラフ，帯グラフ　円グラフによって視覚化する場合，インシデントの種類毎の発生件数をグラフの要素にする事が考えられる．しかし，円グラフはそれぞれの発生件数と全体の発生件数を比較する視覚化であり，根本的にインシデントの種類によってインシデントの合計発生件数がもつ意味は異なってくるため，あまり意味をなさない．さらに，発生件数の多いインシデントが目立つばかりか，発生件数の少ないインシデントの存在が見えなくなってしまうため，セキュリティイベントの視覚化としては適さない．

折れ線グラフ　折れ線グラフでは時系列に沿って発生件数の推移を表示する視覚化ができる [23]．これはセキュリティイベントの発生状態を的確に表しているように見える．しかし，第 2.7節での述べたように，NIDS が検知するセキュリティイベントの内容は様々である．複数種類のセキュリティイベントをまとめて折れ線グラフ化したとしても，急激にセキュリティイベントの検知数が増減したという事実以外は，把握できない．また，種類毎に分けてグラフ化するとグラフ数が多くなり，一度に状況を把握したい場合にはグラフの領域が狭くなる．すると，各グラフの読み取りが困難になり，やはり正確な状況把握は難しくなる．

分布図　分布図は存在の有無を示す事を目的としたグラフである．例えば，横軸を時間経過，縦軸をセキュリティイベントの種類とすることでセキュリティイベントの発生周期を的確に表現できる．しかし，発生件数の変化について表すことができないため，いつ・どのくらいの数のセキュリティイベントが発生したかを判断ができない．また，分布図は 2 つの連続性を持ったデータをもとにグラフの表示位置を決定するため，それぞれが全く異なる性質を持つセキュリティイベントについての表現には適さない．

7.2.2 SnortView

SnortView[57] は snort で取得したセキュリティイベントログ情報を発生日時で並び替えし，表示する機能を有する．幾つかのパターン分析機能も備えている．ただし，並び替えたセキュリティイベントを発生した時間の間隔とは関係なく，ただ順番に表示しているだけなので，各セキュリティイベントの時間間隔を認識しにくい．また，SnortView は各セキュリティイベントを要約せず，個別に表示する仕様となっており，一括して表示される情報が制限されてしまう．これは論文中でも問題点として指摘されており，規模性に支障をきたすと考えられる．

7.2.3 tudumi

tudumi[58, 59] は周期的なパターン検出を目的として構築された機構である．年，月，週，日のような周期を柔軟に表現し，視覚化によってネットワーク管理者がパターンを認識しやすいような処理を施している．ある曜日やある時間帯だけに発生するセキュリティイベントを特定するために効果的な研究であるが，解析すべきセキュリティイベントの数が増えると表示が煩雑になり，規則性の認識が難しいと考えられる．これもまた SnortView と同様に，規模性に支障をきたすと考えられる．

7.2.4 NIDS RainStorm

NIDS RainStorm[60] は，学内ネットワークのような広範囲ネットワークにおいて，異常を検知するために開発された視覚化技術である．特に不審なトラフィックなどの検

知を目的としたNIDSのセキュリティイベントログを利用し，ネットワークの規模が大きくても，セキュリティイベント発生状況を把握できるような視覚化が実現されている．このシステムの最大の問題は，セキュリティイベント情報の解析に時間を要するため，即時的な処理ができない点にある．リスクが高いと考えられるセキュリティイベントの発生時における，迅速なログの解析は当然の要求であり，セキュリティイベントログの視覚化として十分な機能を提供しているとは言い難い．

7.3 設計

本節では視覚化システムにおける要求事項や必要機能を定義し，それらをもとにシステム全体を設計する．

7.3.1 要求事項

A. Komlodi らの研究 [61] では，複数のNIDSオペレータからNIDSが検知したセキュリティイベントのログ視覚化に対する要求を調査している．これを参考とし，本論文でのセキュリティイベントログの視覚化における要求事項を述べる．

要約化　多数のセキュリティイベント情報を迅速にネットワーク管理者へ伝えるためには，情報を要約し必要となる情報を理解できる機能が求められる．特に中規模，大規模ネットワーク全体の状況を把握するためには必須な機能であると言える．本論文では中規模ネットワークをIPv4アドレスのクラスC程度，大規模ネットワークをクラスB程度として議論する．

簡易性　ユーザインターフェースに表示する内容を複雑にしてしまうと，伝えるべき情報が伝わらなくなる恐れがある．同様に，視覚化の仕様として定めていたとしてもネットワーク管理者が直感的に理解できる表示でなければ，誤解を生みやすい．直感に反する視覚化はネットワーク管理者の利便性を損ない，システムの有用性を下げてしまう．また操作設計の観点からも，複雑な操作を必要としない実装が望ましい．

柔軟性　特定のセキュリティイベントに関連する情報を取得するために，セキュリティイベントの種類やIPアドレスなどの項目を条件とした，フィルタが必要となる．簡単な項目による条件でもある程度は有用性を高められるが，視覚化システムにおいてはより柔軟な条件を指定できることが望ましい．

応答性　情報の要約や視覚化が，高速に処理できる必要がある．充実した視覚化が可能だとしても，情報の取得や表示処理が長時間かかってしまっては，ネットワーク管理者が感じる利用負担が大きくなってしまう．視覚化の際に，時間のかかるデータの正規化が必要な場合も，同様に迅速な情報へのアクセスが阻害されてしまう．NIDSが

出力したセキュリティイベントのログを即時的に解析する機能が視覚化システムには必要である．

即時性　バッチ処理などによって，定期的にセキュリティイベントログを加工し，その結果をもとに視覚化しているシステムでは，最後にバッチ処理された時間と，実際に視覚化表示をした時間で，差が生じる．緊急時に必要なセキュリティイベントログを参照できないなどの弊害が考えられるため，実時間的なセキュリティイベントログの処理が必要となる．

規模性　即時性と関わるが，ネットワークの規模に応じて利用が困難になってはならない．クラス B 程度の規模で運用されているネットワークも多数存在する．このようなネットワークを監視している NIDS を有効に利用するためには，規模に左右されにくい視覚化機能が求められる．

7.3.2　具体的な必要機能

第 7.3.1項で述べた要求事項を踏まえ，本論文で実現する視覚化機構に組み込むべき機能を後述する．

全体像の把握　複数セキュリティイベントの全体像把握が，情報要約の最たる目的である．多数のセキュリティイベントが，どのようなタイミングで，どのくらい発生しているかを把握することで，セキュリティイベントの異常発生などを発見する．関連研究でも全体像を迅速に見渡せるための機能が必要であると述べられており [61]，NIDS の運用において欠かせない要件と言える．

時系列にそった表現　ログ情報を要約し特定の期間中に発生したインシデントの数だけを表そうとすると，そのインシデントにどのような発生傾向があるのかを掴めない．時系列にそってセキュリティイベントを見ることで，インシデントが集中的に発生したのか分散的に発生したのか，分散的であればそれは定期的か不定期的であるか，あるいはどの時間帯にどのような発生件数の差があるか，発生開始と発生終了がいつか，などを明らかにできる．

項目毎の分類　NIDS では不正侵入の試みをはじめ調査行為や DoS 攻撃まで様々な種類のセキュリティイベントを検知する．複数種類のセキュリティイベントはそれぞれが完全に独立した別事象である場合もあるが，互いに関連性を持つ可能性もある．これは互いの発生頻度や発生時間を比較し，関連性を発見できる．また，ある 1 つの送信元 IP アドレスからのインシデントを 1 つにまとめて視覚化するだけではなく，インシデントの種類毎に分類して比較することによって，そのソース IP アドレスの行動パターンを分析できる．よって，セキュリティイベントの種類および IP アドレスといっ

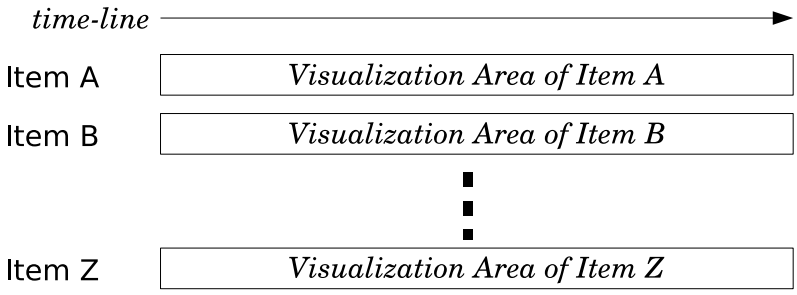

図 7.1: 視覚化画面の設計

た各セキュリティイベントが持つ情報によってセキュリティイベントを分類し，分類したセキュリティイベント毎の発生状況を比較できる機能が必要となる．

量的変化の表現　第 2.6節で述べたように，セキュリティイベントの量的変化を確認することで，注目すべきリスクの高いセキュリティイベントを発見できる可能性がある．量的変化を適切に認識するためには，どのようなタイミングでセキュリティイベントの増減が起こったかを表現する必要がある．

少数セキュリティイベント発生の表現　状態異常の発見のためには量的変化の認識も重要だが，これによって少数セキュリティイベントの発生を覆い隠してしまう恐れがある．特に棒グラフ，折れ線グラフなどを用いた場合は大量発生している時間帯と少数しか発生していない時間帯を同時に表示することによって，少数セキュリティイベントが認識できないほど微細な表現となってしまう．例え少数であってもセキュリティイベントが発生した事実はパターンの理解や異常検知の判断につながると考えられる．

7.3.3　視覚化画面の設計

ユーザインターフェースに表示する視覚化画面の設計概要を図 7.1に示す．本論文で提案する視覚化システムは，指定した一定期間中のセキュリティイベントログを取得し，項目毎に視覚化するセキュリティイベントを分類する．そして項目毎に視覚化領域を設け，その発生状況を画像として出力する．図中では左側が各項目の項目名および項目の説明，右側が各項目毎の視覚化領域となっている．セキュリティイベントログの取得時には必要に応じた条件を指定できるよう実装する．分類するための項目としてセキュリティイベントの種類，送信元 IP アドレス，送信先 IP アドレス，そして送信元 IP アドレスと送信先 IP アドレスの組み合わせの 4 種類を基本とする．NIDS のセキュリティイベントログでは特に，IP アドレスは連続的な数値としてではなく，離散的な宛先情報としての役割が強い．したがって，セキュリティイベントの種類と同様に離散的な項目として扱う．

本実装では 1 つの領域に複数の項目を表現する視覚化手法を採らず，項目毎に視覚化領域を設ける．これは，セキュリティイベントログの増加にともない項目数も多くなりやすいためである．NIDS RainStorm のような 1 つの視覚化領域に複数種類のセキュリティイベントを表現するためには，色などによってセキュリティイベントを区別しなければならない．例として，セキュリティイベントの種類毎に色を割り当てて表示させるとしても，NIDS で検知されるセキュリティイベントの種類数は 50〜100 件以上となる．そのため，各セキュリティイベントの色が似通ってしまい，表示されているセキュリティイベントの種類を把握するのが困難になる．本実装では，項目毎に視覚化領域を設けるているため，どのセキュリティイベント，あるいは IP アドレスのセキュリティイベントログが視覚化されているかを瞬時に判断できる．

視覚化領域では横軸を時間軸とし，左から右に移るに従って時間の経過を表している．各視覚化領域の開始時間，終了時間，倍率はそれぞれ一定であるため，複数項目の発生状況比較によって項目毎の関連性を発見できる．視覚化の具体的なアルゴリズムについては第 7.3.4項で詳細に述べる．

7.3.4 視覚化アルゴリズムの設計

本項では，第 7.3.2項で述べた必要機能を実現するための，視覚化アルゴリズムを設計する．視覚化の説明に必要となる各要素を表 7.1に示す．本実装はある一定期間のセキュリティイベントログを抽出し，これを視覚化する．この期間の時間的な長さを T とする．視覚化に利用できる領域は限界があるため，視覚化領域の横幅が n ピクセルの時 T を n 個に細分化し，細分化された期間長を Δt とする．本実装では細分化された期間を用い，横幅 1 ピクセルの領域をそれぞれ基準として視覚化する．細分化された期間中に発生するセキュリティイベント数を E とし，先頭から i 番目の期間中に発生したセキュリティイベント数を E_i とする．また，E_1 から E_n までの間で最もセキュリティイベント発生数が多い E を，特に E_m とする．c は 1 以上の E の個数とする．$\bar{E}$ はセキュリティイベントが発生していない期間を除いた E による平均値である．他に，各視覚化領域の縦幅を h ピクセルとする．図 7.2に視覚化領域を細分化した概念図を示す．期間 T 中に発生したセキュリティイベントログの発生時間に応じて，1 から n 番目まであるいずれかの期間に振り分ける．そして，先頭から $i(i \in N, 1 \leq i \leq n)$ 番目に振り分けられたセキュリティイベントログの件数が E_i となる．

定常的に発生しているセキュリティイベントであっても，なんらかの理由で突発的に発生頻度が変化する場合がある．例えば集中的に調査行為を受ける，ウィルスの流行により特定のセキュリティイベントが急増する，などが挙げられる．そのため，発生状況が定常的な部分と突発的に変化する部分との差異を表すために $\bar{E}$ を用いる．$\bar{E}$ に近かったとしても c が小さければ強調すべきであり，逆に c が大きければ定常的な状態に近いと判断できる．$\bar{E}$ に近いセキュリティイベント数の強調は H_l と H_u の値によって調整している．H_l が大きければ $\bar{E}$ 近傍でも強調され，H_u が大きければ逆に目立たなくする．H_l と H_u の決定は式 7.1に示す通りである．

表 7.1: 視覚化アルゴリズムに用いる要素

T	視覚化する期間長（$T = n\Delta t$）
Δt	細分化した期間長
n	細分化した期間の個数，かつ視覚化領域の横幅（単位 ピクセル）
E	Δt 中に発生したセキュリティイベント数
E_i	先頭から i 番目の Δt 中に発生したセキュリティイベント数
E_m	E_1 から E_n の中で最もセキュリティイベント数が多い期間のセキュリティイベント数
c	1 以上の E の個数
h	各視覚化領域の縦幅（単位 ピクセル）
$\bar{E}$	0 の E を除外した，セキュリティイベント数の平均（$\bar{E} = \frac{\sum_{i=1}^{n} E_i}{c}$）

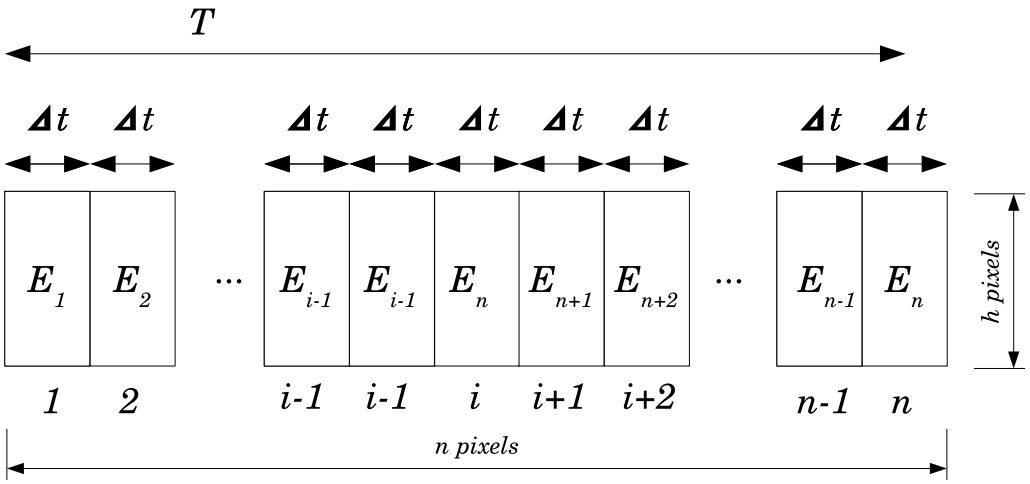

図 7.2: 視覚化する期間の分割

$$
\begin{aligned}
H_l &= log(n) - log(c) \\
H_u &= log(c)
\end{aligned}
\tag{7.1}
$$

次に少数セキュリティイベント発生の認識を容易にするために，E_i の周辺の平均をとる．式7.2に示しているとおり，周囲 4 つの期間と E_i の平均をとる．横幅 1 ピクセルの領域を基準に視覚化するため，周囲の期間でセキュリティイベントが発生していないと 1 ピクセルのみの表示となり，ネットワーク管理者が利用する際に見逃しやすくなってしまう．必要機能として量的変化だけではなく，発生パターンの認識を認識するために，少数しか発生していないセキュリティイベントも見逃さないようにする必

要がある．周囲との平均をとることで，1 つの期間でしか発生していないセキュリティイベントも認識が容易となる．E_i と，その周囲とで得られた平均を A_i とする．

$$A_i = \frac{\sum_{j=i-2}^{i+2} E_j}{5} \qquad (3 \leq i \leq n-2) \tag{7.2}$$

A_i を得られた後，最終的な視覚化における強調度を決定する．強調度は 0 から 1 の間の割合として求める．i 番目の期間の強調度 P_i は，式 7.3によって算出する．$\bar{E}$ 以上と未満によって算出方法が異なり，それぞれ $\bar{E}$ から E_m までの割合と 0 から $\bar{E}$ までの割合を求め，その後 H_u，H_l と乗算する．そして，双方最後に $(H_u + H_l)$ によって除算し，$(H_u + H_l)$ に対する割合 P_i を求める．

$$P_i = \begin{cases} \dfrac{\dfrac{(A_i - \bar{E})H_u}{E_m - \bar{E}} + H_l}{H_u + H_l} & A_i \geq \bar{E}\text{の場合} \\ \dfrac{A_i Hl}{\bar{E}(H_u + H_l)} & A_i < \bar{E}\text{の場合} \end{cases} \tag{7.3}$$

P_i によって，各細分化された期間の視覚化について図 7.3に示す．各 Δt にはそれぞれ幅 1 ピクセル，高さ h ピクセルの視覚化領域が割り当てられている．本実装ではこの視覚化領域の中心から $[\frac{hP_i}{2}]$ ピクセルだけ，上下に向かって色を描画する．P_i が 0 でない限りはセキュリティイベント発生の見落としを割けるために，かならず 1 ピクセル以上は描画する．この方法にもとづき，全ての Δt を描画することによって量的変化の発見やパターン発見を実現する．

7.4 実装

第 7.3節の設計をもとに，Security Event Log Visualizer として **Bishop** を実装した．本節では Bishop の詳細な実装について述べる．

7.4.1 システム構成

Bishop では視覚化した情報を表示するためのユーザインターフェースとして，Web ブラウザを利用している．Hyper Text Markup Language (HTML)[62] を表示する Web ブラウザは様々な OS において実装されているため，管理者は OS を意識せずに Bishop を使用できる．Bishop の具体的なシステム構成を図 7.4に示す．Bishop は HTTP サーバから実行される Common Gateway Interface (CGI) として動作する．実装には C 言語を用い，描画ライブラリとして GD Graphics Library (libgd)[63] を利用した．C 言

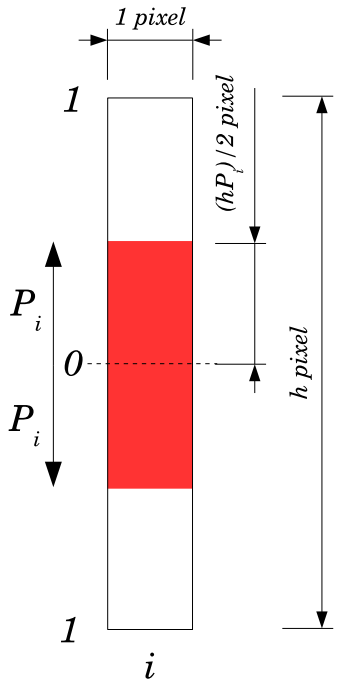

図 7.3: 計算結果を用いた視覚化

語の標準出力によって HTML を生成し，視覚化した画像情報を libgd によって GIF イメージとして出力する．HTML 中に生成したイメージを表示させる Uniform Resource Identifiers (URI) を記述し，テキストによるセキュリティイベント名などの情報と画像情報を同時に出力する．

Bishop はインターフェースの視覚化エンジンの他には，主にデータ取得エンジンと拡張機能管理機構が動作している．データ取得エンジンは 1 つ，あるいは複数のデータベースやファイルから NIDS が出力したログのデータを取得する．ログの格納状態は NIDS の実装や出力先によって様々であるため，特定形式を取得するためのプラグインを必要に応じて用意する．これによって様々な形式のログを視覚化するための抽象化がなされる．本論文執筆時点では，snort が MySQL[45] に出力したセキュリティイベントログおよび，S-NIDS が MySQL に出力したセキュリティイベントログを取得するためのプラグインをそれぞれ実装している．

取得したデータは特定の形式に変換され，拡張機能管理機構に渡される．Bishop における拡張機能とは取得したセキュリティイベントログに，視覚化を補助するための付加情報を与える仕組みである．例えば，本論文執筆時点では検出されたセキュリティイベントログが外部ネットワークから内部ネットワークへのセキュリティイベント，内部ネットワークから外部ネットワークへのセキュリティイベント，内部から内部への

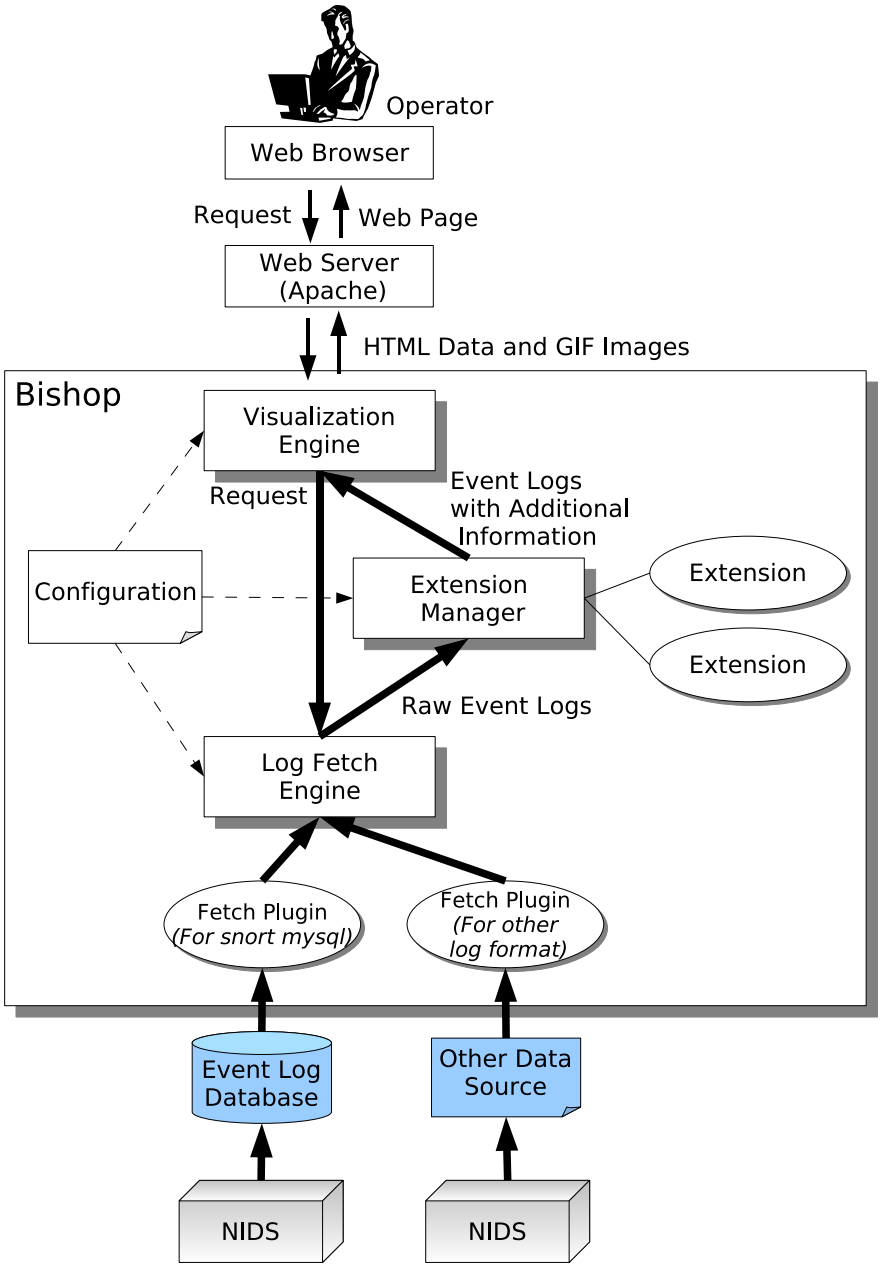

図 7.4: 視覚化システムの構成概要図

セキュリティイベント，外部から外部へのセキュリティイベントのいずれであるかを判別し，結果を各セキュリティイベントログに付加する拡張機能を実装している．第3.1節でも述べたとおり，NIDS が検知するセキュリティイベントはその送信元と送信先によって意味が異なるため，それぞれを分けて考える機能が必要となる機能だと言える．このような送信先，送信元の分類に限らず，OSRES で得られた結果などを付与する機能を組み込むことで，より複雑な状況を把握可能になると考える．

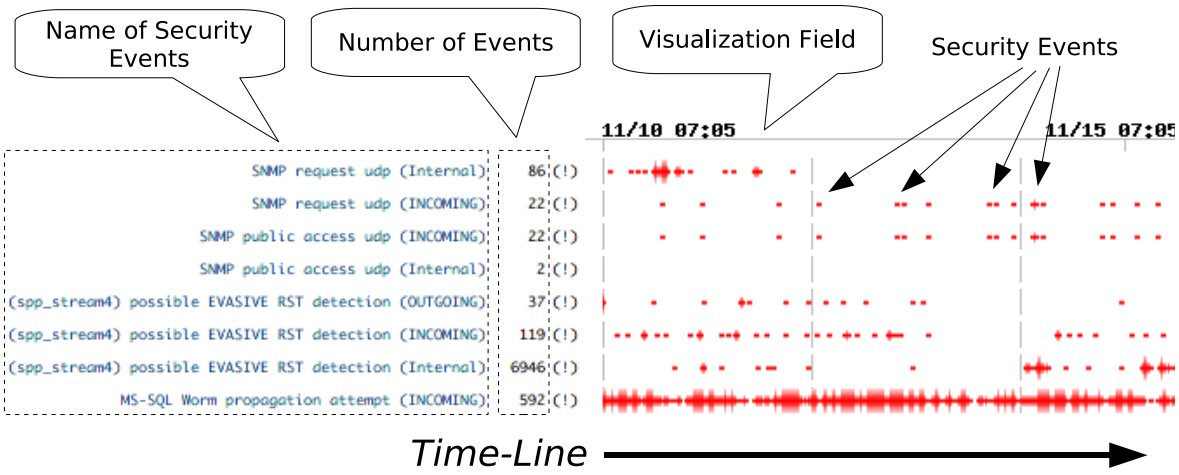

図 7.5: ユーザインターフェースの表示例

7.4.2 インターフェース説明

Bishop によって視覚化されたユーザインターフェースの一部を図 7.5に示す．図では右側がセキュリティイベントの名称，中央が表示されている範囲でのセキュリティイベント検知総数，そして左側が実際の視覚化した画像を示している．色づけされている各点がセキュリティイベントの発生を表しており，視覚化部分は左から右に移動するにつれ時間が経過していることを示している．縦幅が広い部分ほど，同一セキュリティイベントにおいて相対的発生頻度が高い部分となっている．そのため，発生数の高い部分が強調され，発生頻度が少なくなれば目立たなくなっており発生頻度が変化を認識しやすい表示となっている．

視覚化の条件指定についても利用の容易さを考慮している．ユーザインターフェースが Web ブラウザなので各項目名がリンクとなっている．リンクは当該項目のみを表示するような条件が指定されているため，クリックすれば当該項目に関連するセキュリティイベントログだけが視覚化される．また，項目名の横にあるエクスクラメーションマークは，当該項目を除外するための条件を指定したリンクとなっている．また，視覚化された画像の一部をクリックすることによって表示期間を縮小し，より詳細に当該部分を調査できる．

図中の例では，セキュリティイベントの送信元 IP アドレスと送信先 IP アドレスによって分類する拡張機能も実装している．外部ネットワークから内部ネットワークへのセキュリティイベント（InComing），内部ネットワークから外部ネットワークへのセキュリティイベント（OutGoing），内部から内部へのセキュリティイベント（Internal），外部から外部へのセキュリティイベント（External）のいずれであるかを判断し，項目の分類に利用している．InComing，OutGoing，Internal，External の中から必要なものだけを抽出しての視覚化も可能となっている．

7.5 評価

本節では Bishop の NIDS 運用における有用性を示すため，実ネットワークにおいて検知されたセキュリティイベントログの分析事例を第 7.5.1項と第 7.5.2項で述べる．また，第 7.5.3項で各視覚化手法との比較を，第 7.5.4項で応答時間の計測結果について論じる．

7.5.1 セキュリティイベント異常発生の発見の実例

Bishop を用いたセキュリティイベント異常発生状態の発見と，その追跡過程を実例により示す．図 7.6は 7 日分のセキュリティイベントログを処理し，視覚化したユーザインターフェースの一部である．ここで，“SHELLCODE x86 NOOP” というセキュリティイベントの発生頻度が，著しく変化しているのが分かる．“SHELLCODE x86 NOOP” は x86 アーキテクチャの CPU における NOOP(NO OPeratoin)，すなわち「何もしない」という命令コードが含まれるトラフィックを，検知したセキュリティイベントである．x86 アーキテクチャを対象とした攻撃トラフィックに頻繁に含まれる傾向がある．しかし，FTP によるファイル転送時などでも，似たようなトラフィックが多々発生するため，通常は誤検知が多くリスクの判別は困難である．Bishop では，“SHELLCODE x86 NOOP” のような通常はリスク判断が困難なセキュリティイベントでも，注目すべき発生状況としてネットワーク管理者が認識しやすくなっている．

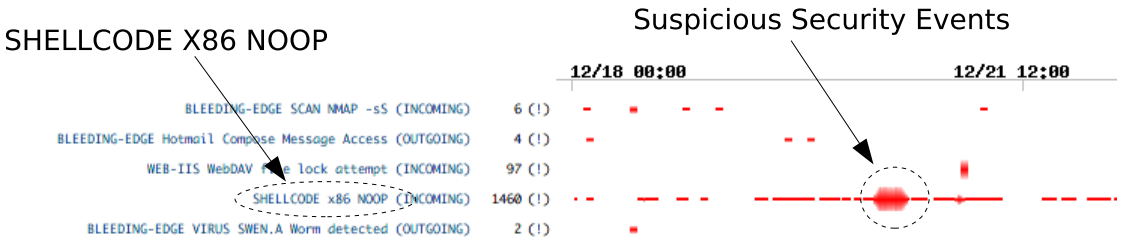

図 7.6: セキュリティイベント異常発生の追跡例 1

注目すべきセキュリティイベントを発見した後には，より詳しい情報を取得できる．図 7.7では，条件によるフィルタによって “SHELLCODE x86 NOOP” のセキュリティイベントのみを視覚化している．これは図 7.6の左側にある項目名をクリックすればフィルタされ，必要に応じて解除できる．項目が 2 つあるのは拡張機能によって，INCOMING のセキュリティイベントと Internal のセキュリティイベントが分類されているためである．

図 7.7で視覚化されている，注目すべきセキュリティイベント発生箇所をクリックすることによって，図 7.8の表示へと移行する．図 7.8は図 7.7と比べて視覚化するためのセキュリティイベントログ取得期間を短くしている．クリックを繰り返すことで，より目的のセキュリティイベントに対して細かくフォーカスできる．

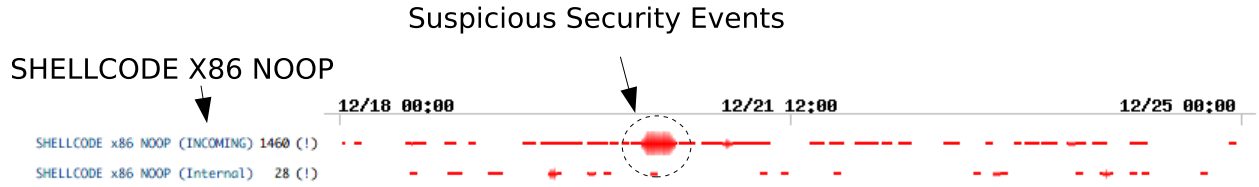

図 7.7: セキュリティイベント異常発生の追跡例 2

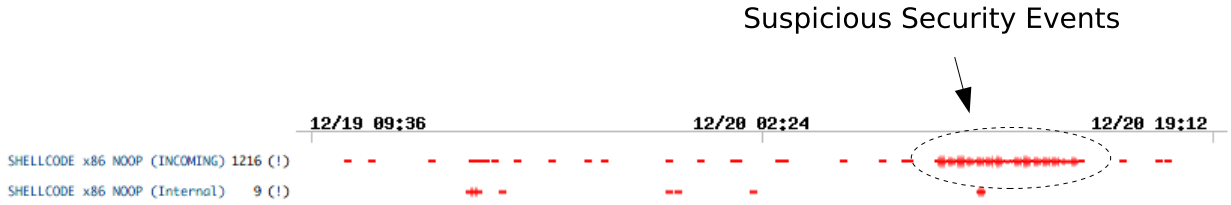

図 7.8: セキュリティイベント異常発生の追跡例 3

図 7.9では図 7.8と同じ条件で，分類するための項目を変えている．図 7.8はセキュリティイベント毎に視覚化していたが，図 7.9は送信元 IP アドレスと送信先 IP アドレスの組み合わせによって分類している．4 項目に分割されたため，やや強調度は低くなっているが，視覚化部分では不規則なセキュリティイベントの発生を見て取れる．ここで各項目に注目する．この 4 項目では全て同一の送信元 IP アドレスとなっている．そこで，次はこの IP アドレスによってフィルタする．

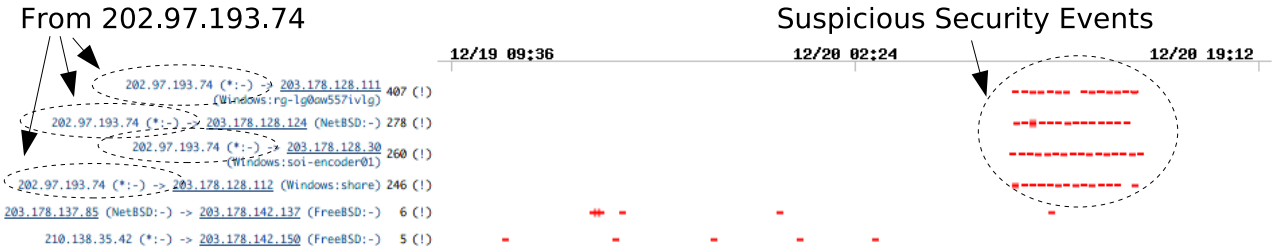

図 7.9: セキュリティイベント異常発生の追跡例 4

送信元 IP アドレスを条件に指定して視覚化するセキュリティイベントログをフィルタしたのち，他に条件として指定していたセキュリティイベントの種類を条件から外す．さらに，分類する項目をもう一度セキュリティイベントの種類に戻したのが図 7.10である．ここで発生しているセキュリティイベントの種類は “SHELLCODE x86 NOOP” の他に “BARE BYTE UNICODE ENCODING”，“WEB-MISC WebDAV search access”

の項目が見られる．図 7.10によって，送信元 IP アドレスが他の攻撃や調査をしていた事がわかり，“SHELLCODE x86 NOOP” も悪意を持った攻撃である可能性が極めて高い事実を示している．

図 7.10: セキュリティイベント異常発生の追跡例 5

7.5.2 関連性のあるセキュリティイベント抽出の実例

図 7.11では，関連性があると考えられる複数のセキュリティイベントの抽出例を示している．図での視覚化はある 24 時間中において検知されたセキュリティイベントから，特定の内部ホストにが送信先ホストとされているセキュリティイベントのみを視覚化している．表示されているセキュリティイベントは不正侵入の試みと調査行為であり，送信先は内部ホストとなっている．

図中の中央上部分では，“NETBIOS SMB Session Setup NTMLSSP unicode asn1 overflow attempt” と “NETBIOS SMB Session Setup AndX request unicode username overflow attempt” の 2 種類のセキュリティイベントが集中して発生しているのがわかる．“NETBIOS SMB Session Setup NTMLSSP unicode asn1 overflow attempt” は Windows の ASN.1 ライブラリが持つ脆弱性である MS04-007[64] を利用した攻撃，“NETBIOS SMB Session Setup AndX request unicode username overflow attempt” は RealSecure/BlackICE[65] がそれぞれ持つ脆弱性 [66] を利用した攻撃である．調査した結果，この 2 種類のセキュリティイベントは，1 つの外部ホストから集中的に攻撃されている事実が明らかになった．また，図の右部に注目すると，“NETBIOS SMB-DS IPC$ unicode share access”，“BLEEDING-EDGE EXPLOIT LSA exploit”，“NETBIOS DCERPC ISystemActivator bind attempt”，“WIN-RPC LSA DS access (lsass.exe) Interface Detected (445/tcp)”，“NETBIOS SMB-DS Session Setup AndX request unicode username overflow attempt”，のセキュリティイベントがそれぞれ時間的に近接しているのが分かる．これもまた同様に，1 つの送信元 IP アドレスからの調査，攻撃である事実が判明した．

このように時系列に沿った視覚化により，表示している内容から関連性のあるセキュリティイベントの発見を実現した．リスクの高いセキュリティイベントの発見時には，関連するセキュリティイベントを調査しインシデントの全容を把握するためにインシデントを構成するセキュリティイベントを特定する．各セキュリティイベントの関連

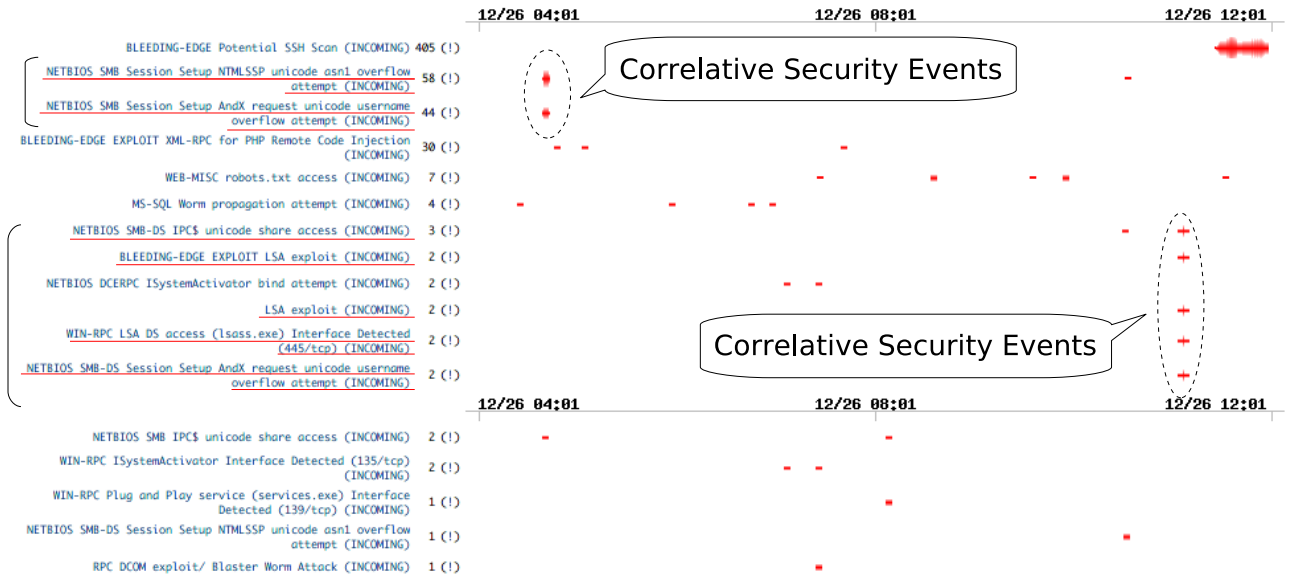

図 7.11: 関連性のあるセキュリティイベントの特定例

表 7.2: 他実装との比較

	Bishop	SnortView	NIDS RainStorm	tudumi	ACID
要約化	○	×	○	○	△
柔軟性	△	×	○	×	○
即時性	○	○	×	○	○
規模性	△	×	○	△	×

性をネットワーク管理者が認識しやすい形で提供することにより，インシデントの解析作業を簡易化していると言える．

7.5.3 他実装との比較による定性的評価

本論文で実装した Bishop と，関連研究で挙げた SnortView, NIDS RainStorm, tudumi とを比較する．さらに多く使われている実装として ACID とも比較する．比較する項目は要約化（多数の情報が要約されているか），柔軟性（要求に応じたフィルタなどができるか），即時性（NIDS が検知したセキュリティイベントログをすぐに利用できるか），そして規模性（中，大規模ネットワークでも利用可能か）の 6 点とする．比較結果をまとめて表 7.2に示し，詳細について後述する．応答性（高速にユーザインターフェースを表示するか），簡易性（直感的な表現かつ簡単な操作か）についても比較するのが望ましいが，論文中に応答性に関する議論が無く公開されていない実装も多いため，今回は比較を見送る．

要約化　Bishop，NIDS RainStorm，tudumi はそれぞれ情報の要約化を主たる目的として設計・実装されており，それぞれ要約化が実現できている．RainStorm は大規模ネットワークでの発生したセキュリティイベントを 1 画面に要約しており，tudumi もいくらかの情報をまとめて表示している．Bishop も多数のセキュリティイベントの発生状況をまとめており，要約化されていると言える．SnortView は視覚化はしているものの，情報の要約は機能は充実しておらず不得手である．ACID は一部情報を要約しているが，一般的なグラフ化手法であるためリスク判別などには利用しにくい．

柔軟性　本論文執筆時点の Bishop では視覚化するセキュリティイベントログのフィルタ条件は簡単にしか指定できず，ネットワーク管理者の要望に応じた複雑なフィルタは実装されていない．SnortView も複雑なフィルタ設定などは提供されておらず，柔軟性に欠ける．ACID は詳細なフィルタ条件を指定できるため，柔軟性には優れる．

即時性　NIDS RainStorm はセキュリティイベントログを予め要約，加工しなければ利用できないため，即時性に劣る．その他の Bishop，SnortView，tudumi，ACID はその都度必要なログ情報にアクセスしている．

規模性　要約化の項でも述べたが，NIDS RainStorm は大規模ネットワーク監視を目的に設計されているため，規模性に優れる．Bishop は中規模ネットワークにおいては問題なく利用できるが，大規模ネットワークでは表示速度などに支障をきたす可能性がある．SnortView は情報を要約していないため中規模ネットワークでも運用が難しいと考えられる．論文中でも規模性への対応を今後の課題としている．tudumi も論文中に示されているユーザインターフェースを見る限り，項目数があまりにも多くなると表示されている内容の判別が困難になると考えられる．

7.5.4　表示時間の計測

ユーザインターフェースとして利用する際の即時性，そして規模性について定量的に評価する．視覚化期間をランダムに変更し，画面表示を 1500 回試行した．処理したセキュリティイベントログ件数と処理に要した時間の関係を図 7.12に示す．システムの特性上指定した期間ではなく，ログの件数によって処理時間が影響を受けやすい．そのため横軸は処理したセキュリティイベントログの件数，縦軸は要した時間（秒）となっている．図中の Visualization の点 (x) はデータベースからのデータ取得処理，Fetch Data の点 (+) は視覚化と表示処理の，処理件数と処理時間を表している．それぞれ，Bishop の開始時，データ取得終了時，終了時に gettimeofday 関数を実行し，その差分をとった．そして，データベースには合計約 97,000,000 件のセキュリティイベントデータが格納されているものを利用した．また，評価に用いた環境を以下に示す．

- CPU: Intel(R) Xeon(TM) CPU 3.06GHz
- ディスク: SCSI HDD (ext3 ファイルシステム)

- メモリ: 2076852 kB (2GB)
- OS: Debian/GNU Linux 3.1
- HTTP サーバ: Apache 1.3.33
- SQL サーバ: MySQL 4.1.11
- NIDS: snort 2.4.3

図 7.12の計測結果から，データ取得処理と視覚化および表示処理はセキュリティイベントログ件数によって変化していると分かる．また，各処理における 1 秒あたりの処理件数の平均は，データ取得処理が 32,639 件，視覚化および表示処理が 844,047 件であった．図 7.12中の f1(x)，f2(x) は式 7.4の通りとなっている．x は横軸の処理件数を表し，f1(x)，f2(x) の結果は縦軸の処理時間 (秒) に対応する．

$$\begin{aligned} f1(x) &= \frac{x}{32639} \\ f2(x) &= \frac{x}{844047} \end{aligned} \tag{7.4}$$

処理時間の平均により，データ取得処理には 10,000 件あたり約 0.3063 秒程，視覚化および表示処理には 10,000 件あたり約 0.0118 秒程度要しているのが分かった．図 3.1で示したネットワークでは，2005 年 1 月 1 日から同年 12 月 25 日において 1 日平均 68401 件のセキュリティイベントが発生していた．68401 件を表示するために必要なのは推定約 2.17 秒，2 倍の 136802 件でも約 4.35 秒である．ストレスのないユーザインターフェースとは言いにくいが，解析などの作業には差し支えないと考えられる．

7.6 今後の課題

7.6.1 データ取得の高速化

第 7.5.4項では，日単位での計測を基準に表示時間の可用性を述べたがより月単位，あるいは年単位の解析には数 10 秒，あるいは数分の時間を要する．発生状況の異常を発見するためには，長期間にわたる発生状況を調査しなければならない場合もある．現実装における処理時間の課題はデータ取得部分であり，これは MySQL による処理が大部分を占めている．よって，データベースにより高速なシステムを採用する，あるいはハードウェアのスペックアップによってより高速化させる手段が挙げられるが，これには高額なソフトやハードが必要となる場合が少なくない．この他には，ある期間以上過去のセキュリティイベントログを定期的に要約する方法が考えられる．どの手法を採るべきかは，今後検討の必要がある．

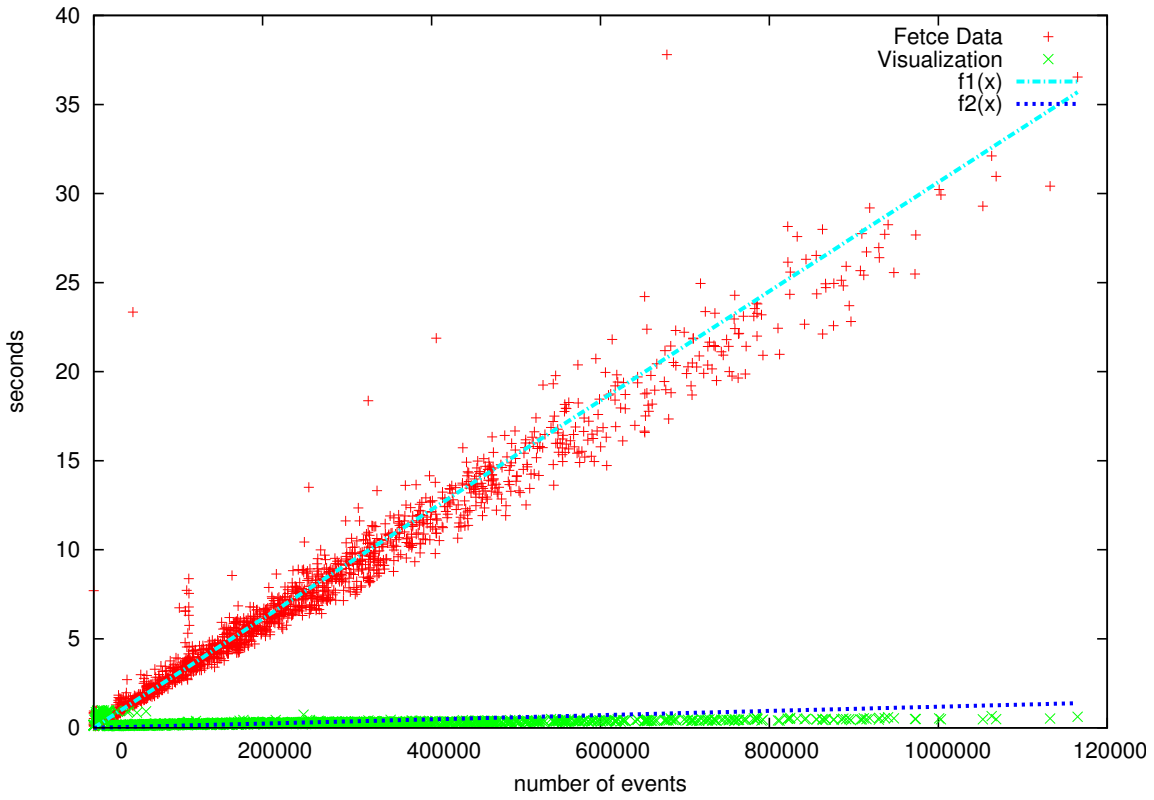

図 7.12: ユーザインターフェースの表示時間計測

7.6.2 全体的な状況把握の効率化

第 7.6.1項とも関わるが，本システムではセキュリティイベントの種類や IP アドレスを項目として扱っている．そのため，長期間の解析になると項目数が急激に増えてしまい発生したセキュリティイベントの全体的な状況把握を阻害してしまう．そのため，さらに全体的な把握に特化した視覚化を考案し，ある一定以上の項目を表示する際には任意の視覚化手法が使えるなどの機能が必要となると考えられる．

7.7 まとめ

本論文では，NIDS のログ視覚化をするにあたっての要求を定義し，それを解決するシステムの実装を行った．このシステムによって，インシデントの発生件数の差に左右されずインシデントを視認，比較できるようになった．そのため，インシデント発生状況の傾向分析の手間を大幅に減らし，誤検出の特定や危険性の高いインシデントの容易な発見を実現した．

第8章　運用効率化の実現

Session based NIDS，OS based Risk Evaluation System，Security Event Log Visualizer について，それぞれ第5章，第6章，第7章で述べた．本章では，この３つの技術を実際に導入した構成，および導入に必要となる実装の拡張について述べる．

8.1　各システム導入時の全体的な構成

各システムの導入構成を図8.1に示す．図中に示す各システムやデータベースはそれぞれ独立しており，システムを１つのホストに集中させる構成，あるいは複数のホストに分散させる構成のいずれでも構築できる．

図8.1中の構成ではEnhanced NIDSとしてSession based NIDSを導入しているが，従来のNIDSも並行して運用している．NIDSは実装によって処理性能や検知できる対象が異なる．処理性能は検知エンジンのハードウェア化や実装手法によってトラフィックの取得やシグネチャとの比較処理速度に差が出やすい．また，検知できるセキュリティイベントも様々であり，特にプロトコル異常のセキュリティイベントは実装によって検知できるセキュリティイベントが大きく異なる．このような理由からNIDSは１つに限らず，複数並行して運用する可能性がある．

NIDSは検知したセキュリティイベントをログとしてデータベースに蓄積する．セキュリティイベントログの解析用のBishopは適宜必要な期間を指定し，抽出したセキュリティイベントログを視覚化する．Bishopによってセキュリティイベント発生状況の全容把握を容易化しており，ネットワーク管理者がセキュリティイベントの異常発生検知やインシデントの全体像を把握するための負担を軽減している．

一方で自動的なセキュリティイベントの異常発生検知，および単独でリスクが低いと考えられるセキュリティイベントの除外を実行するためのソフトウェアが必要となる．本論文ではセキュリティイベントのリスク評価結果とセキュリティイベント異常発生の検知により，必要と判断されるセキュリティイベントの情報をネットワーク管理者に通達するソフトウェアとして**Risk Check and Alert Script**(RICAS)を実装した．RICASの詳細は第8.2節で述べる．RICASはlibosresを利用したリスク評価機能も提供している．

同環境では内部ネットワークでDHCP gazerを動作させる．DHCP gazerは動的にネットワークへ接続するホストのOSも網羅的に把握し，その内容をデータベースに格納する．固定ホストの利用OS情報はネットワーク管理者によって定期的に更新される．格納されているOS情報をもとにlibosresがセキュリティイベントのリスクを判断

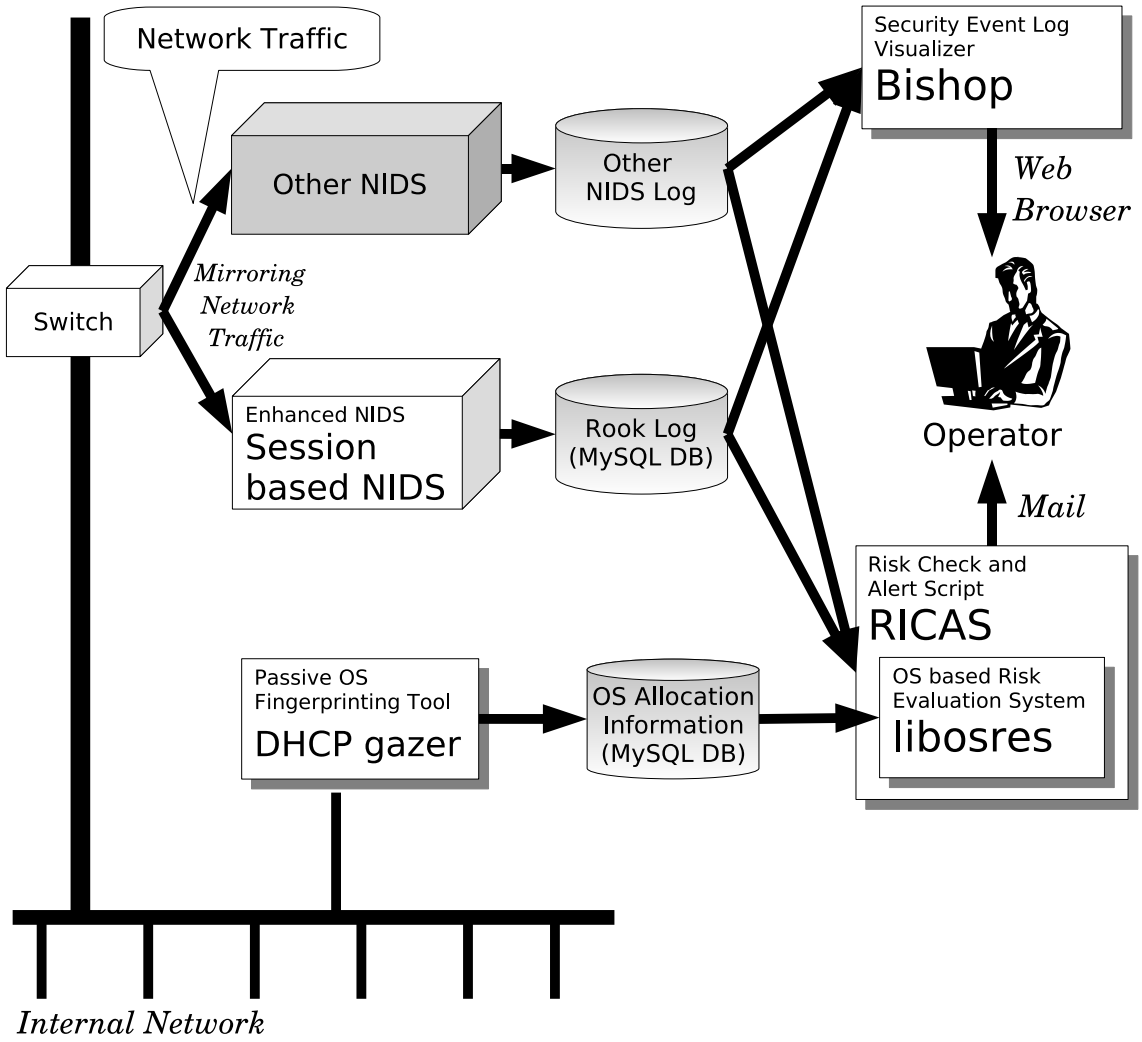

図 8.1: 各システムを導入した NIDS とその周辺の構成

する．

8.2 伝達用ソフトウェアの実装

第 2.6節で述べた監視フェイズにおけるセキュリティイベントの利用方法のうち「セキュリティイベントに付随する情報を利用した自動的なリスク判断」「セキュリティイベント発生状況の異常にもとづいた自動的なリスク判断」を実行し，その結果からネットワーク管理者へ伝達すべきセキュリティイベントを送るソフトウェアとして RICAS を実装した．RICAS は PHP[67]4.3.10-16 によって実装した．定期的にデータベースをポーリングし，最近発生したセキュリティイベントの情報を取得する．RICAS の動作は単独のセキュリティイベント評価と定められた閾値をもとにした発生頻度検査ある

いはその両方を同時に処理する．RICAS はネットワーク管理者に通知すべきセキュリティイベントの発見により，セキュリティイベントの情報をある程度集約した上でネットワーク管理者にメールで送信する．

RICAS は単独のセキュリティイベントをリスク評価するためにセキュリティイベントのパラメータ，S-NIDS によるリスク評価結果を利用している．セキュリティイベントのパラメータと S-NIDS によるリスク評価結果はそれぞれデータベースに格納されている．これらのパラメータと予め指定した条件とが一致すればネットワーク管理者に通知する．OS 情報を元にしたリスクを評価するためには libosres を使用する必要があるが，libosres は C 言語向けに開発したライブラリであるため PHP から関数を呼び出せない．そこで，標準入力からパラメータを渡すと標準出力からリスク評価結果を返すプログラムを作成し，PHP スクリプトから別プロセスとして呼び出す手法を採った．セキュリティイベント発生状況の異常の検知は，セキュリティイベントの種類毎に閾値を設定し指定した時間内に閾値を超える数のセキュリティイベントが発生すれば通知する．

8.3 本研究の評価

本節では，NIDS 運用の効率が本研究で提案したシステムの導入によって改善されていることを示す．

8.3.1 リスク評価可能項目の比較

表 8.1に従来の NIDS が検知した各セキュリティイベントの評価の可能性と，S-NIDS と OSRES を利用した評価の可能性とを比較している．着目すべきは，外部からの不正侵入の試み，外部からの調査行為，そして内部および外部からのプロトコル異常である．

不正侵入の試みと調査行為は第 5章で述べたとおり，Session based NIDS によって大部分のセキュリティイベントが評価可能となる．さらに不正侵入の試みや調査行為に対する応答がない，あるいは応答が不定であっても，OSRES を用いてリスク評価できるセキュリティイベントがある．評価した結果についても第 5.7.1項や第 6.7節で述べた通りセキュリティイベントのリスクを評価し，検知したセキュリティイベントの大部分がリスクの低いセキュリティイベントであった事実を示しており，従来の NIDS と比較すればリスク評価可能性があるセキュリティイベントが増えている．

第 5.8.3項でも述べた通り，S-NIDS は応用されたルールの記述によってプロトコル異常とその応答を監視できる．従来の NIDS が検知するプロトコル異常は，発生頻度の変化や解析フェイズにおける手がかりの取得程度にしか利用できなかった．S-NIDS はプロトコルに異常が起きたトラフィックに加えて，その応答もプロトコル異常が起きているか調査し，双方に異常があれば注目すべきセキュリティイベントとして高リス

表 8.1: NIDS が検知できるセキュリティイベントとその特性

セキュリティイベントの種類	送信元	従来の NIDS	S-NIDS，OSRES の利用
不正侵入の試み	内部	○	○
	外部	×	○
調査行為	内部	○	○
	外部	△	○
プロトコル異常	内部	×	△
	外部	×	△
不審なトラフィック	内部	×	×
	外部	×	×
悪意のあるプログラムの活動	内部	○	○
	外部	×	×
ポリシ違反	内部	○	○
	外部	○	○
サービス妨害攻撃	内部	○	○
	外部	○	○

クの判定を示せる．このような仕組みを利用し，プロトコル異常についても自動的にリスク評価できるセキュリティイベントの幅を広げている．

従来の NIDS ではネットワーク管理者が判断，解析しなければならないセキュリティイベントが多く，管理者は負担を下げるために解析するセキュリティイベントを限定しがちであった．そのため，NIDS の目的の 1 つである「リスクの高いインシデントの早期発見」を達成するためには，自動的にリスク判別できるセキュリティイベントの種類を増やしネットワーク管理者の負担を軽減することが重要となる．これまでで述べたとおり，本研究によって NIDS が検知したセキュリティイベントのリスク評価を従来より容易にしている．これにより「リスクの高いインシデントの早期検出」の達成を促している．

8.3.2 全体的なセキュリティイベント把握の容易化

Bishop は視覚化処理によって多数のセキュリティイベントログを要約し，Web ブラウザを通して閲覧できる．Web ブラウザを表示しているモニターの解像度が 1024 × 768（XGA）の場合，Web ブラウザを画面上に最大で表示すれば約 100,000 件のセキュリティイベントログを約 4,5 画面あまりで表現できる．第 7.5.4項で示した通り 500 台程度のホストが接続するネットワークで 1 日あたりに発生するセキュリティイベントログは約 70,000 件程度であり，その表示速度も実用範囲内である．約 70,000 件のセキュリティイベントログをリスト表示によって全体像を把握するのは，膨大な作業負担である．また，一般的なグラフ化手法ではセキュリティイベント発生状況の把握が困難

な点については第 7.2.1項で述べた通りである．本論文で実装した Bishop を用いることで，従来手法に比べてセキュリティイベントの異常発生を検知するための負担が大幅に軽減される．また，第 7.5.2項では，時間軸に沿った視覚化により関連性があると考えられるセキュリティイベントの特定を容易にしている実例を示した．

以上の成果により，手動によるセキュリティイベントの異常発生検知，そして解析時に関連するセキュリティイベントの抽出を容易にし，リスクの高いインシデントの早期検出とインシデント発見後の調査の負担を軽減することができた．

8.3.3 導入によるコストと効果の変化

本論文で提案したシステムの導入によって全体的にかかるコストと軽減されるコストについて論じる．それぞれをまとめて表 8.2に示す．

導入時のコスト 各システムを導入するために新たに必要となるコストは各ソフトウェアのインストール，データベースの設定，そして運用方針に基づいた自動通知項目の設定が挙げられる．各ソフトウェアのインストールは Session based NIDS，DHCP gazer，Bishop，RICAS，そして libosres である．各ソフトウェアはそれぞれ 1 つずつインストールされていればよく，内部ホスト全てにソフトウェアをインストールする必要は無い．運用方針に基づいた自動通知項目の設定は，管理ネットワークによって基準や対象となるセキュリティイベントが異なる．特にポリシ違反に属するセキュリティイベントは組織によって全く異なるため，運用方針を定めた上でリスクが高いと判断する基準を設定する必要がある．

監視コスト 監視フェイズでは多数のセキュリティイベントログの全体像把握作業が容易になり，さらに手動によってネットワーク管理者が自らリスク評価をしなければならないセキュリティイベントが減少する．従来の NIDS はリスク評価作業と全体像の把握作業が運用コストの大部分を占めていたため，この 2 つによって軽減されるコストは大きい．また，監視フェイズで新たに必要とされる運用コストは無い．

解析コスト 従来の NIDS ではリスク不明なセキュリティイベントが多く，このようなセキュリティイベントを解析しない運用体制も多くあると考えられる．自動的にリスク評価できるセキュリティイベントが増えることで，高いリスクと判断されたセキュリティイベントが以前より増加し解析作業が増える可能性は高い．しかしリスクの高いセキュリティイベントの発見は NIDS の目的に則しているため，このコストは必然であると言える．また，解析における関連セキュリティイベント発見のための作業は負担が低下しており，無駄な運用コストが増えることは無い．

調整コスト 調整フェイズでは自動通知項目の調整をする必要がある．リスクが高いと判断されたセキュリティイベントでも，被害を起こさないインシデントもある．そのため，ネットワーク管理者に通知するための閾値を解析から得られた結果にもとづ

表 8.2: 新たに発生するコストと軽減されるコストの比較

フェイズ	追加されるコスト	軽減されるコスト
導入	各ソフトウェアのインストール 自動通知項目の設定	
監視		多数のセキュリティイベントログの全体像把握 手動によってリスク判断
解析	従来は無視されていたセキュリティイベントの解析	関連するセキュリティイベントの発見作業
調整	自動通知項目の調整	
その他	固定ホストの OS 情報管理	

いて調整する．しかし，このような作業は従来の NIDS ではシグネチャそのものの調整という形で取り組まれてきたため，特別に運用コストが増加するわけではないと考えられる．

その他のコスト　NIDS での各運用フェイズの他に，定期的な固定ホストの OS 情報を管理するコストが発生する．DHCP gazer により動的なホストの OS 情報は自動的に更新されるが，固定ホストについてはネットワーク管理者が自ら調査し，更新しなければならない．しかし，そもそもネットワーク管理者は管理ネットワーク上で固定的に利用されるホストを把握している場合が多く，大幅な運用コスト増加には繋がらないと考えられる．

以上の通り全体的に作業負担を低下させ，追加される運用コストについても運用を阻害するものは無い．すなわち，本論文によって NIDS の運用を効率化できたと言える．

8.4 まとめ

Session based NIDS，OS based Risk Evaluation System，Security Event Log Visualizer によって第 4章で述べた本論文で提案する NIDS 構成の各技術を実現できた．そして，Risk Check and Alert Script を実装したことによって，モデル中にある 2 つの自動処理も実行可能となり，新しい NIDS の構成モデルを具体化できた．また，本研究で提案したモデルと，それを構成する技術を導入することによって NIDS 運用の効率が改善できる事を論じた．

第9章　結論

9.1　まとめ

インターネット上の脅威によって引き起こされる被害を最小限に防ぐためには，インシデントが発生する可能性を認識し，インシデントの発見，事後対応の準備が不可欠である．NIDS はトラフィックを監視し，インシデントの一部であるセキュリティイベントの発見，そしてインシデント発見後の調査に必要となる情報を収集する．NIDS はインシデントの発見，事後対応のために有用なツールだが，従来の NIDS は適切に運用するための負担が大きく，リスクの高いインシデントの早期検出やインシデント発見後の調査が難しくなっている．

NIDS の運用が難しい理由は大きく分けて 2 つある．1 つ目は，NIDS が検知したセキュリティイベントに付随する情報だけではリスク評価が困難な種類のセキュリティイベントが多い点である．特に外部からの不正侵入の試みや外部からの調査行為は，重大なインシデントに発展する可能性を示唆しながらも，実際のリスクは不明である．そのようなセキュリティイベントのリスクは，目標ホストの実地調査など，負担の大きい手段によってしか確認できない．これは，NIDS が検知したセキュリティイベントに付随する情報だけでは，そのセキュリティイベントによってどのような影響がおよぼされたか，あるいはセキュリティイベントがどのような状況を示しているかを，正確に判断できないためである．これによって，リスクが不明なセキュリティイベントが多数検知され，運用を妨げている．2 つ目は，NIDS が検知するセキュリティイベントの数は膨大であり，多数のセキュリティイベントログによって全体的なセキュリティイベントの発生状況を把握するのが難しい点である．セキュリティイベントの発生頻度や発生パターンに変化があれば，単独ではリスクが不明だとしても注目すべきセキュリティイベントとなる．また，インシデント解析時には関連するセキュリティイベントを抽出しなければならないが，多数のセキュリティイベント群の中でどのセキュリティイベントが関連しているかについては，全体像が把握できていなければ判断が難しい．筆者らが NIDS を運用している，500 台あまりのホストが接続されたネットワークでは，一日あたり平均約 65,000 件程度のセキュリティイベントが検知されている．この 65,000 件以上のセキュリティイベントを，ネットワーク管理者が限られた時間で全て把握するのは不可能であり，全体把握によって得られる情報を活用できていない．

このような運用上の課題を解決するために，本論文では新しい NIDS の構成モデルを提案した．従来の NIDS が検知・記録対象としないネットワークトラフィックからリスク評価に利用できる要素を抽出する “Enhanced NIDS”，ホスト情報やネットワーク環境情報を自動的に取得し，リスク評価に利用できる “Security Information Manager”，

そして膨大な数のセキュリティイベントログを要約し，セキュリティイベントの発生頻度や発生パターンの変化をネットワーク管理者が容易に認識できるようにする “Log Aggregation System”．以上の 3 つの要素によって，新しい NIDS のモデルが構成される．本論文では，このモデルを実際に構築するために，3 つの技術を設計，実装した．

まず，“Enhanced NIDS” として **Session based NIDS** を設計，実装した．本論文では，不正侵入の試みや調査行為は，その結果が成功か失敗かによって，応答が変化する種類のセキュリティイベントがある点に着目した．Session based NIDS は不正侵入の試みや調査行為そのもののトラフィックだけではなく，その応答についても監視し，不正侵入の試みや調査行為の成否を判別できる．この技術によって，不正侵入の試みや調査行為のリスクが評価可能となった．

そして，“Security Information Manager” として **OS based Risk Evaluation System** を設計，実装した．OS based Risk Evaluation System は DHCP を利用することで，動的にネットワークに接続するホストについても，そのホストが利用している OS の種類を取得する．収集した OS 情報と NIDS が検知したセキュリティイベントの種類を比較することで，不正侵入の試みの一部をリスク評価可能にした．

さらに，“Log Aggregation System” として，多数のセキュリティイベント情報を要約し，視覚化する **Security Event Log Visualizer** を設計，実装した．Security Event Log Visualizer は迅速なセキュリティイベントの発生頻度，発生パターンの把握を可能とした．関連するセキュリティイベントの抽出も容易に行えるようになり，多数のセキュリティイベントログを扱うのが困難である問題点を解決した．

本論文では，以上の技術をモデルにそって実ネットワークに導入した．その際，自動的なセキュリティイベントのリスク判別，通知を行う Risk Check and Alert Script を実装した．そして，本論文で提案したモデルによって NIDS を運用することで，従来と比較し，運用負担を減らせることを示した．

9.2 今後の課題

9.2.1 各モデル間での情報交換フォーマットの必要性

本論文は第 4.1節で述べたモデルを実現するために S-NIDS，OSRES，Bishop をそれぞれ採用した．しかし，このモデルは各要素毎に複数の技術を併用できるのが望ましい．特に SIM の部分については OS 情報だけに限らず，アプリケーション情報やトポロジ情報，あるいは他のセキュリティデバイスのログなどの同時に利用することで，より他種類のセキュリティイベントをより正確にリスク評価できるようになると考える．今後，このようなモデル間の接続性やセキュリティイベント情報の受け渡しの透過性を提供するためのフレームワークを考案していく必要がある．

9.2.2 より効率的なセキュリティイベントの利用

本論文では，関連するセキュリティイベントの抽出などによって，インシデントの発見や解析を容易化した．しかし，ネットワーク管理者が自らインシデントの発見作業を行うことに変わりはない．さらに，インシデントであるか否かの判断は，ネットワーク管理者の知識や運用経験に左右される部分が多く，NIDS を効率的に運用できるネットワーク管理者は限られてしまう．

この問題を解決するためには，インシデントの発見までを自動化する手法を確立する必要がある．複数のセキュリティイベントや，他のセキュリティデバイスからの情報，そして攻撃の成否情報などを利用し，インシデントを発見する手順を自動化できれば，知識や経験が少ないネットワーク管理者でも NIDS 運用を効率化を支援できると考える．

謝辞

本論文の作成にあたり，御指導いただきました慶應義塾大学環境情報学部教授村井純博士，並びに同学部教授徳田英幸博士，同学部助教授楠本博之博士，同学部助教授中村修博士，同学部助教授高汐一紀博士，同学部専任講師湧川隆次博士に感謝します．

そして，学部在籍中に最も御世話になった慶應義塾大学政策メディア研究科 白畑真氏に深く感謝致します．氏は特にセキュリティの分野に精通されており，研究室に入った直後から，研究や論文執筆などにおいて多くの御指導を頂きました．氏無くしてはこの卒業論文は為しえませんでした．改めて感謝申し上げます．次に，慶應義塾大学政策メディア研究科 南政樹氏に感謝致します．氏には論文誌への投稿や研究の進め方について，様々なご指導を頂きました．氏に勧められ，初めて投稿した国際学会は残念ながら不採録でしたが，その後の研究活動を進める上で大変貴重な体験となりました．本論文の執筆でも，お忙しいところ絶えずご指導いただき，感謝の念に尽きません．

また，慶應義塾大学環境情報学部村井研究室のメンバーである小原泰弘氏，海崎良氏，橋本和樹氏，佐川昭宏氏，久松剛氏，堀場勝広氏，三島和宏氏，小椋康平氏，遠峰隆史氏，大藪勇輝氏，中村友一氏，山本雄大氏，松谷健史氏，金井瑛氏，空閑洋平氏，奥村佑介氏，本多倫夫氏，中島智広氏，尾崎隆亮氏，佐藤龍氏そして東京大学 情報理工学系研究科電子情報学専攻 青山・森川研究室研究員助手 今泉英明博士に感謝いたします．特に小原氏には学部四年間に渡り，論文執筆や研究発表の際に様々な御助言をいただきました．また，金井氏には卒論以外の論文でも，執筆にあたり様々なご協力を頂きました．深く感謝いたします．

スターバックス湘南台イトーヨーカドー店にも感謝致します．執筆に行き詰まった時などは度々立ち寄らせて頂き，居座ること 2 時間，3 時間は珍しくありませんでした．特に提出締切が迫った時期には，2 日に 1 回以上立ち寄らせていただきました．非常に迷惑この上ない客だったと思いますが，それでもなお暖かい笑顔で迎えてくださった店員の皆様に感謝いたします．

執筆活動を続けるにあたり，精神状態を良好に保つため，多くの音楽に御世話になりました．その音楽を作曲された菅野よう子氏，岩崎琢氏，梶浦由記氏，服部隆之氏，川井憲次氏，すぎやまこういち氏，佐藤直紀氏，田中公平氏，植松伸夫氏，Andrew Lloyd Webber 氏，鷺巣詩郎氏，平沢進氏，千住明氏，七瀬光氏，久石譲氏，辻陽氏，保刈久明氏，羽毛田丈氏，James Horner 氏，John Williams 氏，桜庭統氏，椎名豪氏，溝口肇氏，光宗信吉氏にそれぞれ感謝致します．

最後に，大学４年間に渡る生活で，私を支え続けてくれた父と母，そして細井ゆりな氏に感謝します．

参考文献

[1] CERT Advisory CA-2003-04 :MS-SQL Server worm. *http://www.cert.org/advisories/CA-2003-04.html*, Jan 2003.

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

[3] 独立行政法人 情報処理推進機構. 国内・海外におけるコンピュータウィルス被害状況調査. *http://www.ipa.go.jp/security/fy15/reports/virus-survey/documents/2003_calc_model.pdf*, Apr 2004.

[4] Microsoft Corporation. Microsoft Windows. http://www.microsoft.com/.

[5] TIME TO LIVE ON THE NETWORK. Avantgarde. *http://www.avantgarde.com/xxxttln.pdf*, Nov 2004.

[6] ISO Standards. ISO/IEC 17799:2005, Information technology – Security techniques – Code of practice for information security management.

[7] Kevin Mandia, Chris Prosise. INCIDENT RESPONSE. Jul 2003.

[8] Bruce Schneier. Secrets and Lies: Digital Security in a Networked World. *John Wiley & Sons*, Aug 2000.

[9] ハニーネットプロジェクト. ハニーネットプロジェクト 汝の敵を知れ:セキュリティ脅威者の分析. **毎日コミュニケーションズ**, Jun 2005.

[10] L. T. Herberlein, G. V. Dias, Karl N. Levitt, Biswanath Mukherjee, J. Wood, and D. Wolber. A network security monitor. In *IEEE Symposium on Security and Privacy*, pages 296–305, 1990.

[11] WheelGroup Corporation. NetRanger, 1994.

[12] Martin Roesch. Snort. *http://www.snort.org*, 12 1998.

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

[14] Yoann Vandoorselaere. Prelude-IDS - The Hybrid IDS framework. *http://www.prelude-ids.org/*.

[15] 竹森敬祐，三宅優，中尾康二，菅谷史昭，笹瀬巌. Security Operation Center のための IDS ログ分析支援システム. **電子情報通信学会論文誌**, *Vol.J87-A, No.6*, pages 816–825, Jul 2004.

[16] Jonathan B. Postel. RFC 821: Simple Mail Transfer Protocol. *http://www.ietf.org/rfc/rfc821.txt*, Aug 1982.

[17] R. Nelson. Some Observations on Implementations of the Post Office Protocol (POP3). *http://www.ietf.org/rfc/rfc1957.txt*, Jun.

[18] ASCII Table and Description. *http://www.lookuptables.com/*.

[19] IETF. Request For Comments. *http://www.ietf.org/rfc.html*.

[20] win-rpc.rules for Snort. *http://www.snort.gr.jp/MLarchive/snort-users-jp/2004-June/001050.html*, Jun 2004.

[21] Bleeding Edge of Snort. *http://www.bleedingsnort.com/*, Oct 2004.

[22] InterSect Alliance Pty Ltd. RazorBack. *http://www.intersectalliance.com/projects/RazorBack/index.html*, Feb 2001.

[23] Roman Danyliw. Analysis Console for Intrusion Databases. *http://www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html*, Sep 2000.

[24] CERT Advisory CA-1996-26 Denial-of-Service Attack via ping. *http://www.cert.org/advisories/CA-1996-26.html*, Dec 1996.

[25] Juniper Networks Inc. Stateful signature detection. *http://www.juniper.net/products/intrusion/detection.html#Stateful_Sign_Det*.

[26] Network Flight Recorder. *http://www.nfr.com*.

[27] Joel Snyder. Target Based IDS. *http://infosecuritymag.techtarget.com/ss/0,295796,sid6_iss306_art540,00.html*, Jan 2004.

[28] Giovanni Vigna,Richard A. Kemmerer. NetSTAT: A Network-based Intrusion Detection Approach. *ACSAC*, 1998.

[29] Juniper Networks Inc. NetScreen IDP 100/500. http://www.netscreen.com/pdf/NS5000DS_datasheet.pdf.

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

[31] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, Request for Comments 2616. *ftp://ftp.isi.edu/in-notes/rfc2616.txt*, Jun 1999.

[32] Stephen Northcutt, Donald McLachlan, Judy Novak. Network Intrusion Detection: An Analystś Handbook (2nd Edition). *Paperback*, Sep 2000.

[33] Mark Cooper, Stephen Northcutt , Matt Fearnow , Karen Frederick. Intrusion Signatures and Analysis. *Person Education*, Jan 2001.

[34] TCPDUMP/LIBPCAP. *http://www.tcpdump.org/*.

[35] Philip Hazel. PCRE - Perl Compatible Regular Expressions. sep 1997. http://www.pcre.org/.

[36] Sniph. Snot. *http://www.sec33.com/sniph/*, 2001.

[37] Erwan Lemonnier. Protocol Anomaly Detection in Network-based IDSs. *Defcom 28th*, Jun 2001.

[38] Pierre-Alain FAYOLLE, Vincent GLAUME. A Buffer Overflow Study - Attacks & Defenses. *ENSEIRB Networks and Distributed Systems 2002*, 2002.

[39] Sourcefire, Inc. Sourcefire RNA Sensor. 2003.

[40] nCircle Network Security, Inc. nCircle. 2003.

[41] ArcSight. http://www.arcsight.com/.

[42] R. Droms. RFC 2131: Dynamic Host Configuration Procol. *http://www.ietf.org/rfc/rfc2131.txt*, Mar 1997.

[43] 白畑　真，土本 康生，村井　純. DHCP を用いた受動的フィンガープリンティング手法の提案と実装. **第 *111* 回マルチメディア通信と分散処理研究会**, Feb 2003.

[44] Microsoft Corporation. Microsoft security bulletin ms04-028: Buffer overrun in jpeg processing (gdi+) could allow code execution (833987). *http://www.microsoft.com/technet/security/bulletin/MS04-028.mspx*, sep 2004.

[45] MySQL AB. MySQL. http://www.mysql.com/.

[46] Microsoft Corporation. Microsoft SQL Server. http://www.microsoft.com/sql/.

[47] Apple Computer. Mac OS. http://www.apple.com/.

[48] Linus Torvalds. Linux. http://www.kernel.org/.

[49] The FreeBSD Project. FreeBSD. http://www.freebsd.org/.

[50] The NetBSD Foundation. NetBSD. http://www.netbsd.org/.

[51] Sun Microsystems. Sun Solaris. http://www.sun.com/software/solaris/.

[52] CISCO SYSTEMS. http://www.cisco.com/japanese/warp/public/3/jp/.

[53] Cisco Systems. Cisco IP Phones. http://cisco.com/.

[54] 東京特殊電線株式会社. MEGRAS. http://www.totoku.co.jp/products/rfid/index.html.

[55] Polycom, Inc. POLYCOM. http://www.polycom.com/home/.

[56] Circuits Maximus. Snort Report. *http://www.circuitsmaximus.com.*

[57] Hideki Koike, Kazuhiro Ohno. SnortView: Visualization of Snort Logs. 2004.

[58] Tetsuji Takada, Hideki Koike. Tudumi: Information Visualization System for Monitoring and Auditing Computer Logs. July 2002.

[59] 高田 哲司, 小池 英樹. ログ情報の視覚化による不正侵入検知手法の提案. **コンピュータセキュリティシンポジウム***'98*, **情報処理学会**, *pp.153-158*, Oct 1998.

[60] Kulsoom Abdullah, Chris Lee, Gregory Conti, John A. Copeland, John Stasko. IDS RainStorm: Visualizing IDS Alarms. *Workshop on Visualization for Computer Security (VizSEC '05)*, Oct 2005.

[61] A. Komlodi, J. Goodall, and W. Lutters. An information visualization framework for intrusion detection, 2004.

[62] W3C. HTML 4.01 Specification. *http://www.w3.org/TR/html4/*, December 1999.

[63] Boutell.Com, Inc. GD Graphics Library. *http://www.boutell.com/gd/.*

[64] Microsoft Corporation. Microsoft Security Bulletin MS04-007ASN.1 Vulnerability Could Allow Code Execution. *http://www.microsoft.com/technet/security/bulletin/MS04-007.mspx*, Jun 2004.

[65] Internet Security Systems. RealSecure/BlackICE. http://www.iss.net/find_products/network.php.

[66] eEye Digital Security. RealSecure/BlackICE Server Message Block (SMB) Processing Overflow.
http://www.eeye.com/html/Research/Advisories/AD20040226.html, Feb 2004.

[67] Rasmus Lerdorf. PHP: Hypertext Preprocessor, 1995.

付 録A　libosresの使用方法

ライブラリ初期化　libosresを初期化するための関数を表A.1に示す．初期化関数によって返されるOSRES構造体はデータベースへのアクセスするためのパラメータや，高速にホスト情報を参照するためのデータ構造が含まれる．OSRES構造体はlibosresで提供している関数を利用する際，常に必要となる．

ホスト情報データベースの指定　libosresがOS情報を格納したデータベースに接続するためのパラメータ指定関数を図A.2に示す．本実装ではMySQLデータベースサーバに接続するためのホスト名，ユーザ名，パスワード，データベース名，接続ポート番号をそれぞれ指定する．

ルールファイルの指定　libosresではNIDSに応じたルールファイルを指定する必要がある．表A.3に示す関数によって，ルールファイルを読み込み，評価処理に利用する．

評価処理の準備　大量のセキュリティイベントを評価する場合，その都度データベースにOSの情報を問い合わせると処理時間が大きくなってしまう．表A.4に示す関数を呼び出し，一定期間内のOS情報を最適化して保持することで，処理を高速化する．

評価処理関数の呼び出し　これまでの準備がが正常に実行されていることで，評価処理の関数を呼び出せる．表A.5に示した関数によってリスクを評価し，結果を返値として返す．

終了関数の呼び出し　最後にlibosresのために確保した記憶領域やデータベースとの接続を解放するために，表A.6に示す終了処理の関数を呼び出す．

表 A.1: libosresの初期化関数

概要	`OSRES * osresInit ()`
説明	libosresを初期化する
引数	無し
返値	初期化に成功した場合はOSRES構造体の実体を返す．失敗した場合はNULLを返す．

表 A.2: OS 情報データベースへの接続用パラメータの設定

概要	`int osresConnectDB (OSRES * osres, char * dbHost, char * dbUser, char * dbPass, char *dbName, char * dbPort);`
説明	libpawn で使用するリレーショナルデータベースへアクセスするためのパラメータを指定する.
引数	dbHost は MySQL サーバが動作しているホストのアドレス，dbUser には MySQL サーバに接続するためのユーザ名，dbPass にはユーザ名に対応するパスワード，dbName にはホスト情報を蓄積しているデータベース名，dbPort には MySQL サーバのポート番号をそれぞれ指定する.
返値	MySQL サーバに正常に接続できた場合は 0 を，項目異常や接続失敗の場合には-1 を返す.

表 A.3: リスク評価用ルールの指定関数

概要	`int osresSetRule (OSRES * osres, char * rulePath)`
説明	リスク評価の際に用いるルールファイルを指定する
引数	rulePath にルールファイルのパスを指定する.
返値	ファイルのオープンに失敗した，あるいはルールファイルの書式が誤っていた場合は-1 を，それ以外は 0 を返す

表 A.4: 評価処理の最適化関数

概要	`int osresSetSpan (OSRES * osres, time_t start, time_t end)`
説明	リスク評価する可能性のある時間範囲を指定する.
引数	unixtime 形式で start 開始時間を，end に終了時間を指定する.
返値	最適化処理に失敗した場合は-1 を，それ以外は 0 を返す.

表 A.5: リスク評価実行関数

概要	`int osresEval (OSRES * osres, char * sig_name, struct sockaddr_in * ip, time_t time);`
説明	セキュリティイベントのリスクを評価する関数.
引数	sig_name にルールのシグネチャ名，ip に当該ホストの IP アドレス，そして time にセキュリティイベントが発生した時刻を引数として与える.
返値	リスクがあると判断された場合，定数として定義されている OSRES_EVENT_RISKY を，リスクが低いと判断された場合 OSRES_EVENT_SAFETY を，不明な場合は OSRES_EVENT_UNKNOWN をそれぞれ返す. エラーが発生した場合は OSRES_ERROR を返す.

表 A.6: 終了処理関数

概要	`int osresDestory (OSRES * osres);`
説明	OSRES 構造体を破棄し，評価処理を終了する関数.
引数	osres に破棄する OSRES 構造体を指定する.
返値	正常に処理が終了した場合は 0 を，異常が発生した場合は-1 を返す.