卒業論文　　2003年度 (平成15年度)

ポリシに基づいたネットワーク
セキュリティイベント対応システムの構築

慶應義塾大学 環境情報学部

氏名：白畑　真

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

南 政樹

平成15年12月29日

卒業論文要旨 2003 年度 (平成 15 年度)

ポリシに基づいたネットワーク セキュリティイベント対応システムの構築

論文要旨

インターネットの利用範囲の拡大に伴い、ワームの感染やサーバへの侵入といったコンピュータセキュリティインシデントの影響が拡大している。2001 年に発生した CodeRed ワームは、ファイアウォールによって保護されているはずのネットワーク内部で感染活動を拡大した。また、毎日新たなソフトウェアのセキュリティ上の欠陥が発見されている。

このように、セキュリティインシデントが予想外の箇所で発生することは希ではない。従って、あらかじめセキュリティインシデントが発生することを想定した体制作りが必要になる。

本研究の目的は、セキュリティインシデントが発生した際に、管理者の負担をかけずに、迅速に問題に対処することである。

既存のセキュリティインシデント対応体制はシステム化されていないため、個別の事象ごとに管理者による対応を要している。その結果、セキュリティインシデント対応は高い運用コストを要している。実時間性のある対応を行うためには 24 時間 365 日の対応体制で運用を行う必要がある。また、対処を人間が行うため、コンピュータによる対応よりも即時性に欠ける。

本研究では、このセキュリティインシデント対応の問題を解決するために、統合的なインシデント対応モデルを提案し、モデルに従ったシステムの設計、実装を行った。

本システムでは、管理者の対応手順をインシデント対応ルールとして記述する。そして、各種デバイスからセキュリティイベントを広く収集し、これらのイベントデータをルールに基づいて評価する。その際、必要に応じてセキュリティインシデント対応の実施や、管理者への通知を実現できた。

本システムにより、ルールに基づくセキュリティイベント処理が実現した。これにより、インシデントレスポンス管理者の負担が軽減され、人間による対処よりも迅速な対応が実現した。

キーワード

1. インターネット，2. セキュリティ，3. インシデントレスポンス，4. IDS

慶應義塾大学 環境情報学部

白畑 真

Abstract of Bachelor's Thesis

Academic Year 2003

A Policy-based Automated Handling Mechanism for Network Security Events

In this thesis, I propose the framework that response to the security incident at a moment's notice without operation cost.

Computer security incidents are increasing by expansion of the Internet. The infection of the CodeRed worm was expanded internal network that Protected by t@CAEH[in 2001. The new security holes are also discovered everyday.

These cases are indicates that security incidents are often occurred at an unexpected point. Building an assumption framework for these incidents is needed in advance.

Existing action to these incidents is not systematized. Thus, operators are needed individual correspondence. Therefore the correspondence of these incidents requires high operational cost. Real-time correspondence requires 24/7 operation by human. This is inferior to by computer in instantaneously.

Synthesizing model of correspondence the security incidents is being proposed for resolution of this problem. This research proposes the design and implementation based on this model.

This system describes an operator's correspondence procedure as an incident correspondence rule. Security events are widely collected from various devices. These event data is evaluated based on a rule. In the evaluation, this system worked correspondence to the security incident if necessary, notified the security event to the operator.

Therefore, this system has achieved to security event processing based on the rule. This system reduces the cost of correspondence by incident response operator, achieves more quick correspondence than by human.

Keywords :
1. Internet, 2. Security, 3. Incident Responce, 4. IDS

Keio University , Faculty of Environmental Information

Shin SHIRAHATA

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景

1990 年代半ば以降、インターネットの急激な普及は、社会全般に大きな変化をもたらした。インターネットの利用範囲は、当初の学術分野から商業、初等中等教育、医療、行政など幅広い分野へと拡大し、今日の重要な社会インフラとなった。

インターネット上ではクラッキング、ウイルスやワームの拡散、DDoS(Distributed Denial of Service) 攻撃などネットワークセキュリティに関連する事象が発生しており、インターネットインフラストラクチャの信頼性を脅かしている。

実際、2001 年に発生した CodeRed[1] ワームの感染が拡大した事件では、安全だと思われていたネットワークにおいても被害が続出した [2]。

従来から、インターネットから内部ネットワークを保護するために、外部ネットワークと内部ネットワークの境界においてファイアウォールを用いていた。しかし、CodeRed 事件では、ノートパソコンなど、ネットワーク外部でワームに感染したホストが内部ネットワークに持ち込まれることにより、ファイアウォールによるフィルタリングとは無関係に被害が拡大した。

この事件は、セキュリティ上の問題は想定外の箇所から発生するため、事前対策だけでは完全にセキュリティを確保できないこと、そして問題が発生した後の対策を準備することの重要性を示している。

1.2　セキュリティインシデントの種類とその対応

本研究におけるセキュリティインシデントは、“RFC2350 - Expectations for Computer Security Incident Response” に従い、コンピュータの動作あるいはネットワークセキュリティを危険に晒す、あらゆる悪影響を及ぼすイベントをセキュリティインシデントと定義する。

セキュリティインシデントは、一般的に次の種類に分類できる。

守秘性喪失

情報の守秘性が失われるケース。例えば、パスワード攻撃が成功し、暗号化されたパスワードファイルの内容が解読されたケースや、Web サイトに設置された CGI(Common Gateway Interface) [3] にセキュリティホールがあり、個人情報が漏洩したケースがある。

完全性侵害

情報に対する完全性 (integrity) が侵害されるケース。rootkit[4] やトロイの木馬 [5] によるコマンドの置き換えや、Web ページの内容の改竄といったケースがある。

サービス不能攻撃 (Denial of Service)

処理能力を超える大量のアクセスを故意に行い、サービスの提供を妨害する行為である。一般的にはサーバに対して想定を超えたデータを送信し、サービスを不能にする行為と、ホストやネットワークに対して処理能力を超えたパケットを送りつけ、意図的に輻輳を引き起こす行為に分類できる。

濫用

サービス、システムもしくは情報の濫用行為で、ポリシで許容されない利用によって悪影響を及ぼす行為である。具体的には、メールサーバが第三者中継を許可しているため、意図していない UCE(Unsolicited CommercialEmail) を中継するケース、ルータにおいて ICMP Redirected broadcast を許可しているため、Smurf 攻撃の増幅器 [6] となっているケースがある。

システム破壊

システムの破壊を意図した行為。データベースやディスクの内容の消去や破壊に関したケースがこれにあたる。

セキュリティインシデントは、嫌疑がかけられている事象から、影響が確定した段階の事象まで、さまざまな段階の事象を意味する。インシデントレスポンスを行う上では、インシデントの抑止にとどまらず、当該事象の発生の有無を明らかにすることを含め、それぞれに適切な対処が求められる。

1.3 本論文の目的

ネットワークのセキュリティを維持するには、セキュリティインシデントの発生を前提とした対応策が必要である。同時にインシデントを早期に発見し、迅速に対処できるフレームワークが必要である。

本論文では、ネットワーク管理者が監視および対応しているセキュリティインシデントに対して、あらかじめ記述されたルールに従い、定義された対応、もしくは管理者へ通知するシステムを提案し、本システムの有効性を証明するために実装を行う。

これにより、セキュリティインシデント対応の運用コストを低減し、管理者の負担を軽減する。また、24 時間 365 日の有人運用を行えないネットワーク環境であっても、常にセキュリティインシデントに対応できるようにする。また、インシデント対応に要する時間を短縮し、管理者が対応するよりも迅速に対応できるようにする。

本研究の実現手法としては、ネットワークに接続された様々な機器からログデータや侵入検知データといったセキュリティイベントを収集し、ルールに基づいて収集したデータを一元的に評価する。評価結果に応じて、セキュリティイベントを管理者にメールで通知したり、インシデント対応を支援するプログラムを実行できる。

セキュリティイベントは、ファイアウォールや IDS といったセキュリティデバイス、ルータやスイッチなどのネットワークデバイス、Web サーバや DNS サーバといったサービスホストなどから収集する。

本研究で想定する環境は、ファイアウォールや IDS などのセキュリティデバイスが設置されている中規模以上のネットワークである。また、多数のエンドユーザが存在しており、数人程度の管理者がネットワークを運用していることを想定している。

1.4 本論文の構成

本論文は第 2章において、ネットワークセキュリティインシデントとセキュリティイベントの概要、および想定するセキュリティインシデントレスポンスの枠組みについて述べる。

第 3章において、本研究の要素技術であるセキュリティインシデント情報の収集技術、およびインシデント情報の交換技術について述べる。インシデント情報の収集技術では、ファイアウォールや IDS、IDP といったセキュリティイベントのデータソースを取り上げ、インシデント情報の交換フォーマットである IDMEF および IODEF、SNML について述べる。

第 4章においては、関連研究として AirCERT、swatch、ArcSight の紹介と、それらが抱える問題点や今後の要求などを整理する。

第 5章では、セキュリティインシデントの特性を明らかにするネットワーク環境情報について述べる。

第 6章では、問題点の解決・今後の要求を満たすシステムの設計を述べ、第 7章で具体的な実装方法に関して述べる。

さらに、第 8章において、構築されたシステムが提示した要件を満たし、既存の問題を解決しているか、求められる性能と比較し、十分な性能を達成しているかを軸に評価する。

最後に第 9章で本研究の成果と今後の課題をまとめる。

第2章　インシデント対応

2.1　インシデント対応

2.1.1　インシデントの定義

インシデントという単語はコンピュータセキュリティに限らず、他の分野で利用されている。例えば航空の分野では、航空法第 76 条の 2 に定められている「航空事故が発生するおそれがあると認められる事態」を指す。

一般的にコンピュータセキュリティインシデントとは、コンピュータやネットワークに関係するセキュリティ上の事象であり、例えばコンピュータへの侵入、サービス妨害、機密漏洩、攻撃準備行為等を指す。本研究では、具体的な被害の発生の有無とは関係なく、ネットワーク上を通じて意図的にセキュリティ侵害を行う行為をセキュリティインシデントと定義する。

また、セキュリティ侵害の発生に伴う要素である個々の事象をセキュリティイベントと定義する。セキュリティインシデントは単なるイベントデータではなく、複数のイベントを組み合わせたデータ集合として捉えることができる。従って、単一のセキュリティインシデントは、複数の攻撃手法、攻撃元、日時等といった一つ以上のセキュリティイベントから構成されるといえる。図 2.1にセキュリティインシデントとセキュリティイベントの関係を示す。この図では、一連のセキュリティインシデントは単一もしくは複数のセキュリティイベントから構成されていることを示している。

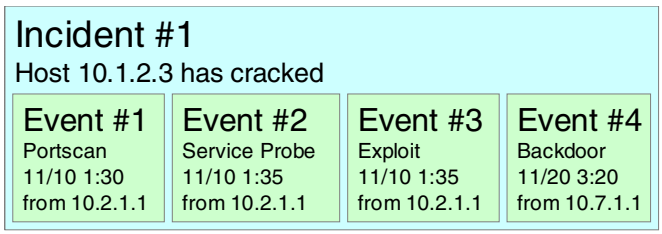


図 2.1: セキュリティインシデントとセキュリティイベント

2.1.2　インシデントレスポンスの定義

セキュリティインシデントレスポンスは、コンピュータセキュリティインシデントが発生した際のさまざまな対応策を包括的に意味する単語である。

インシデントレスポンスは、すでに発生したセキュリティインシデントの対処だけでなく、原因究明や関係者との調整、問題の再発防止など、セキュリティインシデントに関連した対応であると定義する。

2.1.3 インシデントレスポンスの必要性

コンピュータセキュリティインシデントには適切な対応が必須である。

攻撃者に管理権限を詐取されたホストをそのまま放置すれば、他のホストに対する攻撃の踏み台として悪用される恐れがある。そして、踏み台となったホストが外部のサイトを攻撃することで、インシデントがさらに拡大し、攻撃者の身元の追及がますます困難となる。また、組織の信頼失墜や管理不備に対する賠償請求などの法的リスクも想定される。

同様に、自サイトに対して攻撃が行われる場合も深刻である。侵入を受けたホストが踏み台とされてしまい、サイト内部の他のホストに対して攻撃するケースも想定される。例えば、外部との通信が比較的自由に行えるが重要性が高くないサーバを通じて、外部との直接通信が禁じられている極めて重要なサーバが攻撃される場合である。

このように、セキュリティインシデントに適切に対応しなければ、被害が拡大してしまうこととなり、全容の把握がより困難となる。従って、想定されるインシデントに対して判断方針と手順といったポリシと手続きを具体的に決めておくといった、事前対策が鍵となる。

2.2 インシデントレスポンスの手順

Best Current Practice として発行された RFC 2350 - Expectations for Computer Security Incident Response[7] によると、インシデントレスポンスには、インシデントトリアージ、インシデントコーディネーション、インシデントの解決という三種類の活動が含まれる。

一方、CERT/CC が発行している Handbook for Computer Security Incident Response Teams (CSIRTs)[8] によると、インシデントレスポンスサービスには、通常トリアージ機能、対処機能、広報機能、フィードバック機能が含まれるとしている。

Handbook for Computer Security Incident Response Teams においては、読者としてセキュリティインシデントレスポンスの専門チームである CSIRT の運用者を想定している。一方、Expectations for Computer Security Incident Response は CSIRT に対する一般的なインターネットコミュニティの期待を表明するという形をとっており、インシデントレスポンスを俯瞰的に説明している。

本研究においては、Expectations for Computer SecurityIncident Response に基づき、インシデントレスポンスの手順を次の五項目に分類した。このうち、インシデントトリアージ、インシデントコーディネーションの二種類については、RFC2350 の定義にそのまま従う。インシデントの解決については、短期的な視点に基づいたインシ

デントへの対処と、長期的な視野に基づいたインシデントに対するフォローアップの二種類に分割して扱う。

加えて、インシデントの発見もインシデントレスポンスの手順の一部として扱う。インシデントの発見は、厳密な意味でのインシデントレスポンスには含まれないが、インシデントレスポンスの根源となる手順であるため、広い視点でインシデントレスポンスを検討する上では欠かせない手順だからである。

- インシデントの発見
- インシデントトリアージ
- インシデントへの対処
- インシデントコーディネーション
- インシデントに対するフォローアップ

これらの各手順におけるセキュリティイベントとセキュリティインシデントの取り扱い、および各手順における管理者の運用コストの低減策について議論する。

2.2.1 インシデントの発見

セキュリティインシデントの発見は、管理者がシステムを運用する過程で得られるセキュリティイベントから発覚する場合と、利用者や外部のサイトからの問い合わせによってインシデントが発覚する場合がある。

前者の場合、ファイアウォールや IDS といったネットワーク機器のログやトラフィックデータ、ホストのサービスのログなどがセキュリティイベントとして利用できる。しかし、これらのイベントデータはシステム運用の過程で得られたデータに過ぎず、必ずしもインシデントを意味するわけではない。たとえ IDS のログのようなセキュリティインシデントの発見を目的としたデバイスであっても、セキュリティイベントである攻撃と考えられるトラフィックを検知しただけである。

また、後者の場合、組織内外からセキュリティインシデントに関する照会があったとしても、その裏付けとなる証拠を確保できるまでは、その時点では単なるセキュリティイベントである。

これらの理由から、インシデントの発見段階では得られたデータだけに基づいて、無条件にセキュリティインシデントであると判断することは困難である。そのため、セキュリティイベントの解釈は、インシデントレスポンスを実施する管理者に委ねられている。

2.2.2 トリアージ

セキュリティイベントに対するトリアージは、リスクに基づく対応の優先度付けと、各サイトのインシデントに対応するルールを実行する手順である。

トリアージ (triage) はフランス語の trier(選び出す、選り分ける) から来た言葉といわれる。現在ではトリアージは一般に医学で用いられている用語で、限られた環境の中で、できるだけ多くの人々を救命するため、緊急度の高い患者を優先的に対応するという意味で用いられている。

ポートスキャンやサービスプローブといった脅威に直接結びつかないセキュリティイベントには低い優先度で対応し、Blaster ワームの感染活動のような緊急対応が必要なイベントに対しては高い優先度で対応するといった優先度付けを行う手順がインシデントトリアージである。

インシデント対応においては、第2.2.1節で述べたように IDS のアラートや Web サーバのアクセスログなどのセキュリティイベントや、外部のサイトからの報告に対する事実確認と、その脅威の評価が必要である。

トリアージの基準となる要素は常に変化する。新種のコンピュータウイルスやワームの発生や、新たな脆弱性情報の公開、ネットワーク内部へのノートパソコンの持ち込みなど、リスク要因が動的に変化する。このような状況下において一貫したトリアージを行うためには、トリアージを行う主体を一元化し、常に同じ基準を適用することが望まれる。

ネットワークが社会の広い範囲で利用されるようになった今日、セキュリティインシデントに対応する管理者は、一日数百万エントリを超えるセキュリティイベントの中から本当に重要なイベントを選別し、インシデントの全体像をつかむ必要がある。IDS などによって生成されるセキュリティイベントには、単なる ping の応答のようなリスクが低いイベントから、プログラムのバッファオーバーランを狙った攻撃コードの実行といったリスクの高いイベントまで、さまざまな種類が含まれている。

従って、管理者は得られたセキュリティイベントに対してトリアージを実施し、リスクの低いものを対象から除外する、直近の攻撃の動向などの情報に注目する、といったふるい分け作業が必要になる。

IDS からアラートを受けた場合や、利用者や外部のサイトからセキュリティに関する連絡を受けた場合には、それが本当に発生したものなのかどうかについて、内容に関する裏付けとなるデータを調査する。また、連絡を受けた内容については、必ずしもセキュリティインシデントではなく、単なる事故や障害などに起因する問題である可能性も視野に入れつつ、検討する必要がある。

インシデントトリアージには、次のような留意事項がある。

- 内容の事実確認
- 影響範囲の検討
- セキュリティイベントデータの正規化

いったんトリアージフェーズに入ったセキュリティイベントに対しては、後から容易に参照できるようにするため識別子を付与することが望ましい。識別子を付与しておけば、個々のセキュリティイベントを容易に識別できるようになる。また、単一の

セキュリティインシデントにおける複数のセキュリティイベントの相関関係を明らかにする際にも役立つ。

2.2.3 インシデント対応

インシデントの分析

真に対応する必要性があると認められたセキュリティイベントは、セキュリティインシデントとして扱われる。個々のセキュリティインシデントに対してもっとも適切な対応策を講じるためには、そのインシデントがどのような性質を持っているのかを見極めなければならない。

インシデントを特徴づける要素としては、以下の項目が考えられる。

- 攻撃元
- 攻撃先
- インシデントが発生した日時
- 攻撃手法
- 影響範囲
- 被害発生の要因
- 講じうる対応策
- 被害拡大の可能性

上記の要因から、次の手順においてどのような対応策を選択するのかを決定する。

また、複数のセキュリティイベントがどのようにインシデントに関連しているかについても、時系列や攻撃対象のデータ、攻撃者の意図の推測などに基づき見極める必要がある。

組織内におけるインシデントの分析

自組織内においてインシデントが発生した場合、インシデントに関わる立場が被害者、加害者のいずれかであっても、その組織内のホストに関して詳細な調査ができる可能性がある。

例えば、あるホストが踏み台になった場合、ログファイルやバックドアプログラム、トロイの木馬といったファイルが残されている可能性がある。また、ウイルスに感染した場合であれば、感染したウイルスのファイルを入手できればその種類を同定できる場合が多い。

このように、インシデントに関連したホストが自組織内であった場合には、利用しているソフトウェア環境やシステム構成を把握することが容易である。そのインデントが発生したホストの利用形態など、利用環境を知ることにより、インシデントの原因や影響範囲などを推測するのに役立つ。

この際の留意点としては、インシデントに対する迅速な対応と詳細な分析を両立さるために、証拠となりうるデータの収集と保存作業を優先させるべきである。証拠となりうるデータを収集しておけば、後に法的措置を講じる際にそのデータを活用できる可能性がある。Guidelines for Evidence Collection and Archiving [9] によれば、ホストにおいてメモリ上のデータ等の揮発性の高いデータからディスクといった揮発性の低いデータの順に証拠となりうるデータを保全するとよいとされている。また並行して、ネットワークレベルの IDS やネットワーク監査ソフトウェアを運用し、組織内のトラフィックデータを収集することが望まれる。なお、データを収集した手順についても併せて記録しておく。

インシデントにおけるの組織外要素の分析

組織を超えたインシデントを取り扱う場合の課題は、組織外の要素の分析である。管理権限を有さない組織外のネットワークやホストにおけるインシデントの正確な状況の把握は困難である。詳細なデータにアクセスできない、あるいは正確な情報に触れることを必ずしも期待できないからである。

しかし、インシデントの立場の一方は自組織であることが通常である。そこで、IDS や Passive OS Fingerprinting[10] といった手法を用いて、通信相手に関するデータを収集し、インシデントに用いられた攻撃手法を推測する際に役立つデータを得ることが重要である。

2.2.4 コーディネーション

インターネットのようなオープンなネットワークを通じて発生するセキュリティインシデントの対応手順においては、組織内あるいは組織間でのコーディネーションが欠かせない。一貫したセキュリティインシデントレスポンスを実施する上では、管理者がインシデント全体を把握したコーディネータの役割を果たすことが求められる。

インシデントコーディネーションにおいては、次のような問題が懸念される:

入手可能性

他組織が保有しているセキュリティインシデントに関係したデータの入手可能性の問題である。他の組織のポリシ上の問題、あるいは技術的な問題などがあるため、自組織に対してインシデントの分析に必要な情報が提供されることは必ずしも期待できない。

信憑性

インシデントコーディネーションにおける情報の信憑性の問題である。相手側から提供された情報が偽情報であったり、身元が詐称される可能性など、他組織から提供されたインシデントに関する情報が、正確な情報であるかを検証することは難しい。

即時性

インシデントレスポンスを迅速に進める上で、即時性のあるコーディネーションが必要である。

セキュリティイベントに対する互換データフォーマットの必要性

セキュリティイベントを複数のシステム間で的確に伝達するためには、データフォーマットを統一する必要がある。

多くの CSIRT ではセキュリティイベントやインシデント情報の報告のため、独自の報告フォーマットを用意している。ただし、管理者による手動の対応を前提としているため、日時や IP アドレスなどほぼ共通の項目を持っているにも関わらず、フォーマットの互換性がないという問題点を抱えている。

セキュリティイベントデータは管理者や CSIRT 間でやりとりする他に、IDS などのシステムとシステムの間、あるいはシステムと管理者間でやりとりされる場合もあるため、互換性のあるデータフォーマットが求められる。

2.2.5 インシデントのフォローアップ

セキュリティインシデントに対する対応を行った後には、インシデントの詳細を調査し、再発を防止するなどのフォローアップが求められる。

セキュリティインシデントのフォローアップ手順においては、原因究明と再発防止という大きな目標が存在する。フォローアップ以前のインシデントレスポンスの各段階においては、被害の拡大防止という面が一般的に第一の目標として重視されたが、フォローアップ手順においては、セキュリティインシデントの全貌を明らかにすることが求められるため、重要である。

セキュリティインシデントの原因究明のためには、全体像を描く必要がある。全体像を描くには、これまでに収集されたセキュリティイベントのデータや、分析段階で収集したディスクイメージなどの証拠となりうるデータ、セキュリティ上の脆弱性のデータを組み立て、前後関係を推測する。

セキュリティインシデントの原因が明らかになった場合には、その原因が脆弱性を抱えたソフトウェアの利用や、ファイアウォールのフィルタの設定ミスといったシステム上の問題に起因するのか、違法性のあるファイル共有がネットワーク上で行われているといった利用ポリシ上の問題に起因するのかという問題の種類によって、適切なフォローアップの手段を講じる。また、原因が明らかにならない場合でも、考え得るリスクから原因を推測し、そのリスクを軽減する策を講じる。

フォローアップ手段としては、組織内外に対する注意喚起、当該システムの構成変更によるインシデントの再発防止、各種ポリシの改訂、JPCERT/CC といった公的 CSIRT、警察への被害届けの提出などが選択肢がある。

2.3 インシデント対応の要件

管理者が適切な対応を講じるためには、以下に挙げる要件を満たす必要がある。

2.3.1 即時性

コンピュータセキュリティインシデントには短時間のうちに被害が拡大する特性がある。特に、ワームなどの自動化された攻撃に伴うセキュリティインシデントは人間の攻撃によるそれと比較して、短い時間に被害が拡大する傾向にある。

1988 年 11 月 2 日に発生したインターネットワーム事件は、インターネット史上最初の大規模セキュリティインシデントとされている。このワームは、当時の低速なネットワーク環境においても、初めて感染が発見されてから数時間のうちに当時のインターネットの広範囲に拡散した [11],[12]。Morris Worm と呼ばれるこのワームは当時コーネル大学の大学院生だった Robert Tappan Morris Jr. 氏によって書かれたもので、sendmail のデバッグオプションや finger デーモンの入力バッファ、rsh および rexec コマンドに存在する脆弱性を通じて感染を広げた。

また、2003 年 1 月 25 日に発生した SQL Slammer ワーム (別称 Sapphire ワーム) に伴うセキュリティインシデントでは、これまでにない速度でワームが拡散した。David Moore 氏らの報告 [13] によれば、8.5 秒毎にワームの拡大範囲は倍増し、10 分で脆弱なホストのうちの約 9 割に感染した。観測データを図 2.2に示す。

Slammer ワームは脆弱性を抱えた Microsoft SQL Server[14] と MSDE 2000[15] を狙い、UDP[16] の 1434 番ポートにパケットを送信する感染活動を行う。短時間で被害の拡大した要因としては、404 バイトという小さなペイロードサイズであったため、短時間で転送が行える点、UDP プロトコルを利用したため、ネゴシエーションが不要であった点がある。

ワーム以外のセキュリティインシデントであっても、その対策に即時性が求められることには変わりがない。意図的に脆弱なシステムを設置し、攻撃者の行動を観察するハニーポットを構築する Honeynet Project を通じて明らかになったケース [17] では、標準インストール状態の RedHat Linux 6.2 をインターネットに接続してから、わずか 15 分のうちにスキャン、プローブを受け、攻撃されるというインシデントが観察されている。

従って、セキュリティインシデントが発生した場合には、その種類を問わず、いかなる場合においても迅速な対応が必要となる。

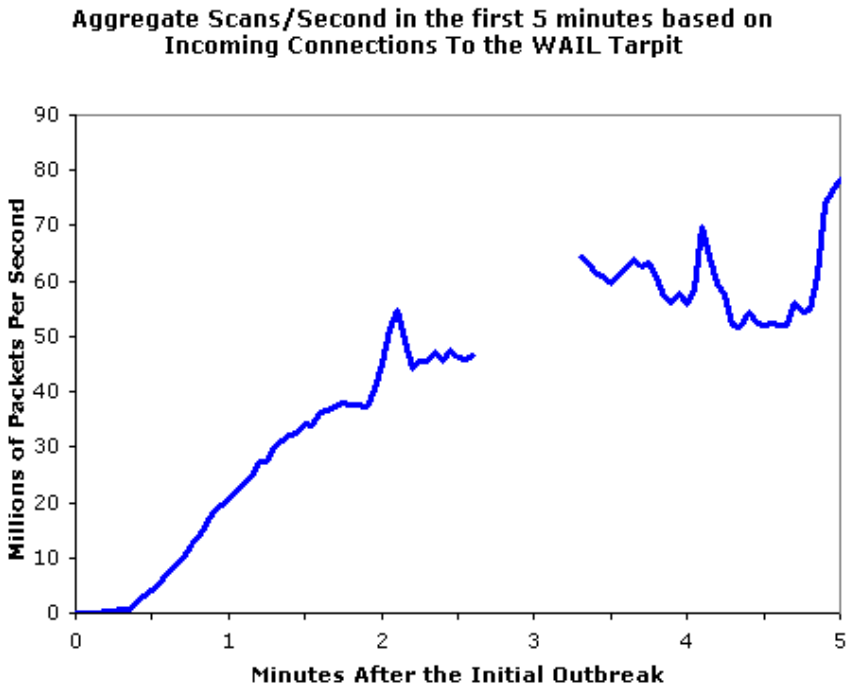


図 2.2: The early progress of Sapphire, as measured at the University of Wisconsin Advanced Internet Lab (WAIL) Tarpit

2.3.2 インシデント拡大の抑止

セキュリティインシデントが発生した後、被害範囲の拡大を防止するためには攻撃者の活動を封じ込める必要がある。第 2.1.3節で述べたように、組織の内外を問わずインシデントに伴う被害が拡大しうるため、その範囲を最小限に留めるような対策が求められる。

一般的に、ネットワークの設計段階において、インシデントの拡大を抑止する設計が行われている。例えば、ファイアウォールを用いたネットワーク構築手法では、外部向けサーバを公開するための DMZ(DeMilitarized Zone, 非武装地帯) と呼ばれる専用のセグメントに設置する。

現在のファイアウォールは、通常セグメントとセグメントの境界に設置する。これはセグメント間の通信を制御するルールを定義することで、セグメントを超えた攻撃の防止を意図している。DMZ セグメントに配置されたサーバから、他のセグメントに対して攻撃が行われたとしても、ファイアウォールを介した通信となる。

しかし、ファイアウォールでは同一セグメント内の通信を制限できない。従って、同一セグメント内で行われる攻撃から防御するためには、ホスト単位での対策が必要である。

2.3.3 ポリシの反映

本論文ではポリシの具体的な表現をルールと呼ぶ。抽象度の高いポリシを、OSI 参照モデル [18] の枠内で表現可能なルールに記述することで、実際のネットワーク環境においてポリシを適用できる。

インターネットにはさまざまな国や地域、組織が接続している。しかし、ネットワークセキュリティインシデントに関連する法規制の内容は、国や地域によって異なっているため、その規制に従ったオペレーションを行うことが法令遵守の観点から重要である。

例えば、アメリカ合衆国においては 18 U.S.C. 2511(2)(c)(monitoring by law enforcement agency) もしくは 2511(2)(d)(provider monitoring)[19] に基づき、ログオンバナーを表示しない限りは通信内容を監視できない。

また、インシデントレスポンスにおいては情報セキュリティポリシや利用許可ポリシ (AUP) の反映も必須事項である。情報セキュリティポリシにおいては、実際に守るべき規定や、それを実際に実施する詳細な手順が定められており、場合によってはインシデントレスポンスに関する条項が含まれる場合もある。

そのため、インシデントレスポンスの際にも、これらの情報セキュリティポリシが遵守されていたかを検討する。また、利用許可ポリシも情報セキュリティポリシと同様、インシデントレスポンスの際に考慮しなければならないポリシである。利用許可ポリシにおいて、組織の内外を問わずポートスキャンや攻撃を禁止することで、インシデントレスポンスの根拠を明文化できる。

さらに、これらのポリシにおいては、組織や部門、役職によって異なるポリシが適用されることもあるため、インシデントに関係するポリシを見極めることが適切なインシデントレスポンスの実施につながる。

2.3.4 証拠の確保

セキュリティインシデントの発生を定義するためには、その根拠が必要である。単なるセキュリティ上の問題の疑いであるセキュリティイベントから、真に脅威であるセキュリティインシデントを特定し、対応を進めていく過程においては証拠の確保が重要である。

第 2.2.1節で述べた通り、セキュリティイベントや外部のサイトからの連絡がインシデントの発見のきっかけとなる。しかし、疑陽性 (false positive) の可能性があるため、これらのデータを無条件で信頼できない。得られたデータが真のセキュリティインシデントであるかについて確認作業が必要である。特に現在の IDS の出力は、誤検知データを多く含んでいるため、全てをインシデントとして扱うことは困難である。

上記のような IDS の過剰反応や管理者の勘違いなどに起因する疑陽性の問題を解決するため、インシデントレスポンスの過程で証拠の確保が求められる。

2.4 インシデント対応の課題

第2.3節で述べたとおり、セキュリティインシデント対応の要件は多岐に渡る。管理者は継続的なインシデント対応体制を維持するために、多大な運用コストを支払っている。

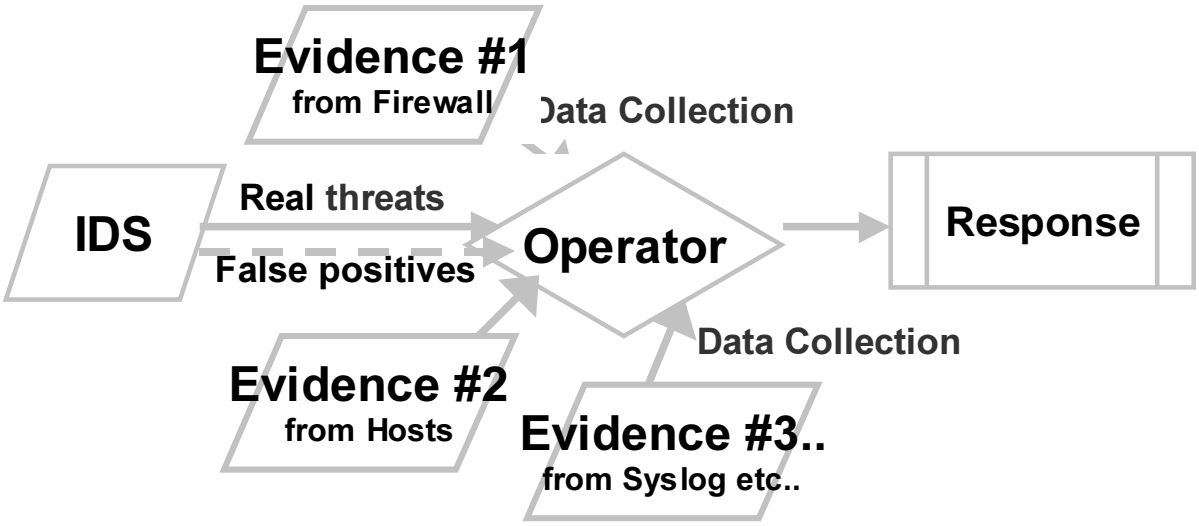


図 2.3: 管理者によるインシデントレスポンスの手順の例

現状のセキュリティインシデント対応の大部分は人間の手に委ねられているため、コンピュータが対応可能な対応作業でさえ、管理者が個々のデータを解釈して対応策を判断し、手動で対応している。図2.4に、管理者のインシデントレスポンスの手順の一例を示す。

そのため、実時間性を重視したインシデント対応体制の構築には、24時間365日にわたる有人対応体制が要求される。また一般に、人間が実際に対応する速度はコンピュータの対応する速度に劣るため、迅速なセキュリティインシデント対応という面で課題が残る。

管理者がセキュリティインシデント対応を手動で行っている背景には、IDSなど既存のセキュリティデバイスの出力の信憑性が低く、個々の出力の信憑性を検証する必要がある点、セキュリティインシデントを裏付ける十分なデータを提供していない点、既存のシステム管理自動化ツールでは、対応要件を満たすことができない点が挙げられる。

本研究のフレームワークでは、定型化可能な運用作業をコンピュータに記述することで、コンピュータが対応可能な問題に関してはコンピュータが対処を行い、コンピュータが対処できない問題に関してのみ管理者が対応できるようにする。

特に、管理者が行ってきたセキュリティインシデントの証拠収集作業を自動化することにより、これまでより短い時間で幅広いデータを得られる。

その結果、従来、管理者が行ってきた作業の一部をコンピュータが実施することにより、セキュリティインシデントの対応コストを軽減するとともに、インシデントに対してより迅速に対応できるようにすることを目標としている。

第3章　インシデント対応における要素技術

3.1　インシデント情報の収集

2.2.1で述べたとおり、インシデント情報を収集するためにはさまざまな方法がある。本項では、さまざまなインシデントの発見を目的とした各種手法について述べる。

3.1.1　IDS

IDS(Intrusion Detection System) は、侵入トラフィックを検知するシステムである。IDS は、その運用形態から次の二種類に分類できる。

NIDS

Network based Intrusion Detection System; ネットワーク上のトラフィックを観測し、トラフィックの中に攻撃と思われるトラフィックを発見した場合に警告する。一般的には、ネットワークセグメントの境界において、トラフィックをタッピングし、通過するトラフィックに対して監視を行う運用形態がある。

HIDS

Host based Intrusion Detection System; 単独のホスト上でプロセスやログを監視し、侵入と考えられる振る舞いが観測された場合に警告する。具体的にはホスト上の全ファイルのハッシュ値のデータベースを作成しておき、ハッシュ値検査によりファイルの改竄を検知する、あるいはメモリ空間を監視し、kernel rootkit を検出する機能等がある。

本研究においては方式を問わず、IDS からセキュリティイベントのデータを収集する。

3.1.2　IDPS

IDPS(Intrusion Detection & Prevention System) は、ルータやブリッジに IDS の機能を組み込んだシステムである。

IDPS は受動的に侵入を検知するだけであるのに対し、IDP はパケットの内容に対して侵入検知を行い、その結果に基づき当該パケットの転送の可否を決定する。結果として IDPS は、ファイアウォール的な動作を行うことになる。

本研究においては、IDPS をセキュリティイベントのデータソースとして活用できる。

ただし、現状の IDPS はパケットに対する検査能力が低いため、パケット転送レートが低くなってしまう。また、パケット転送時におけるネットワーク遅延も拡大してしまう。

3.1.3 NMS

ネットワーク管理システム (NMS) は、ネットワークの稼働性を維持するための監視を行うシステムである。サーバやスイッチ、ルータなどネットワーク上に存在するデバイスの稼働状況を監視し、障害が発生した際には管理者に通知する。また、各ネットワーク機器の設定を変更できる。

セキュリティインシデント対応システムとネットワーク管理システムの間には、

- 影響が複数のシステムに波及する
- 即時性を重視した対応が必要
- 運用コスト軽減に対する要求

という共通点がある。従ってこれらの観点から、ネットワーク管理システムは本研究の参考となるといえる。

3.2 インシデント情報の交換フォーマット

3.2.1 IDMEF

IETF においては、IDWG(Intrusion Detection Exchange Format Working Group) が 1998 年に発足した。このワーキンググループでは、侵入検知とレスポンスシステム、管理システム間で情報共有するためのデータ形式と交換手順を定義することを目標としている。

2003 年 12 月現在、IDWG は次のようなドキュメントを提出している。

- Intrusion Detection Mesage Exchange Requirements [20]
  侵入検知メッセージ交換に関する要件を定義
- Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition [21]
  侵入検知メッセージのデータモデルと XML の DTD を定義
- The Intrusion Detection Exchange Protocol (IDXP) [22]
  侵入検知データを交換するプロトコルを定義する。

- The TUNNEL Profile [23]
  コネクション志向型のトランスポートプロトコル “BEEP[24]” 用トンネルプロファイルを定義

idwg で作成されたドキュメントのうち、TUNNEL Profile はすでに RFC となっている。また、残りの Internet-Draft はすでに IESG に提出されており、RFC 化の最終段階にある。
図 3.1に UML による IDMEF のデータモデルの概略を示す。

3.2.2 IOMEF

IODEF(Incident Data Exchange Format) はセキュリティインシデント情報の交換フォーマットである。IODEF は IETF の INCH(Extended Incident Handling) Working Group で標準化が進められている。
2003 年 12 月現在、IDWG は次のようなドキュメントを提出している。

- Requirements for Format for INcident Report Exchange (FINE) [25]
  インシデント報告の交換に関する要件を定義

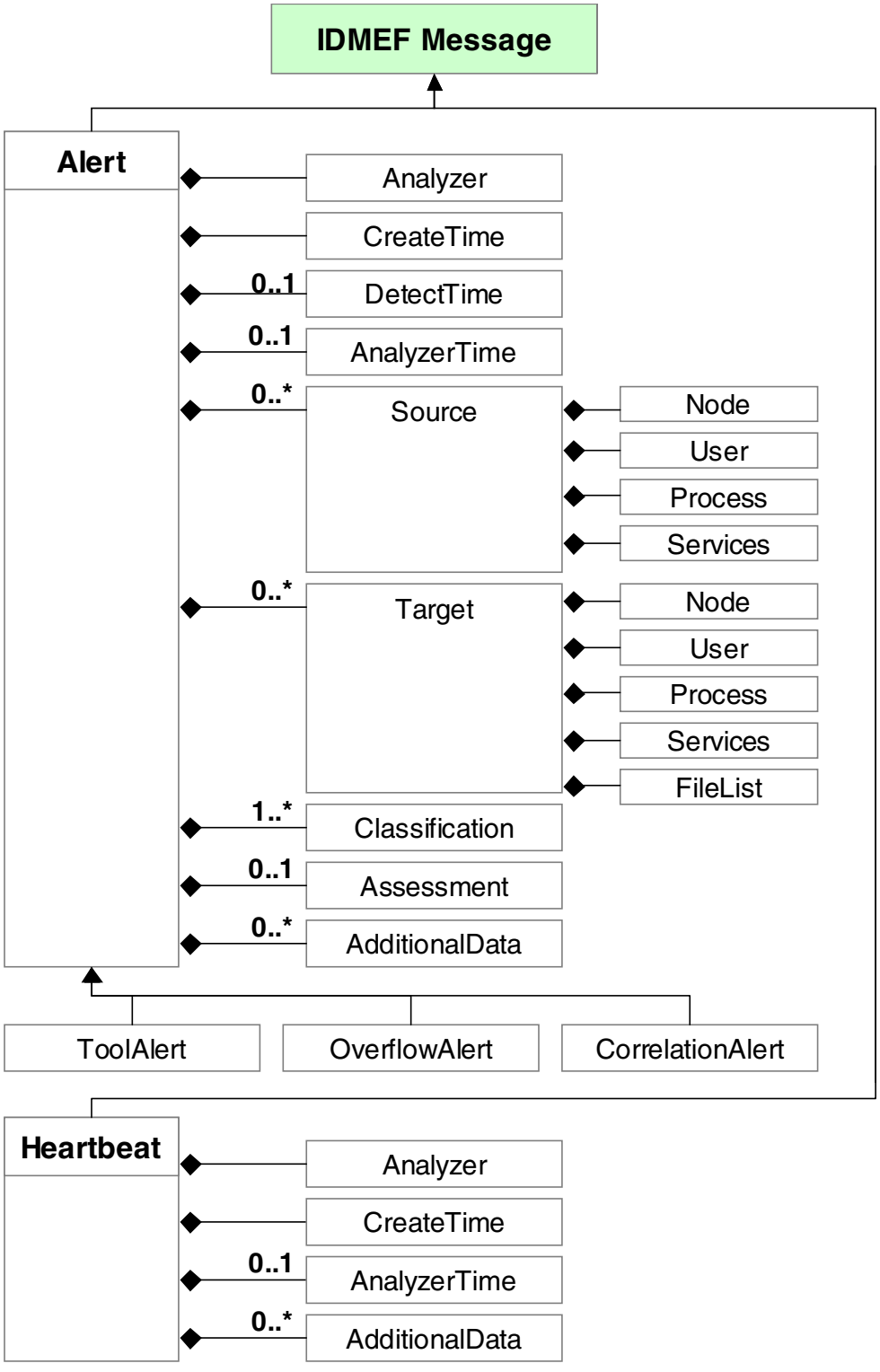
- The Incident Data Exchange Format Data Model and XML Implementation Document Type Definition [26]
  インシデントデータの交換フォーマットに関するデータモデルと XML の DTD を定義

また、実装ガイドラインドキュメントも 2003 年 11 月に Internet-Draft として提出される予定であったが、2003 年 12 月現在、Internet-Draft は提出されていない。
IODEF のデータモデルは、IDMEF のそれに基づいているため、原則的には IDMEF の上位互換となっている。両者の差異は想定環境にあり、IDMEF は単一ドメイン内のセキュリティイベントデータの流通を想定しているのに対し、IODEF はドメイン間のセキュリティインシデント情報の流通を想定している。

本研究のモデルでは、主に単一サイト内におけるセキュリティインシデントレスポンスを扱うため、IODEF を直接は扱わない。しかし、外部サイトからの連絡も重要なセキュリティインシデントのデータソースである以上、IODEF も外部からのデータソースとしての活用できる。

3.2.3 SNML

SNML は Simple Network Martup Language の略で、AirCERT プロジェクトで用いられている XML DTD である。AirCERT プロジェクトでは、オープンソース IDS の



f

図 3.1: セキュリティインシデントとセキュリティイベント

Snort[27] のプラグインの spo_xml、テキストデータ正規化ツールの rex で用いられている。

SNML はパケットのヘッダを表現することに主眼が置かれており、個々のセキュリティイベントの IP ヘッダ、プロトコル、ポート番号、ペイロードなどを表現できる。個々のセキュリティインシデントに関するデータとしては、IDS のアラートとイベントの参照データを表現できる。

SNML は、IDMEF と比較して比較的抽象度が低い DTD であるといえる。

図 3.2に UML による SNML のデータモデルの概略を示す。

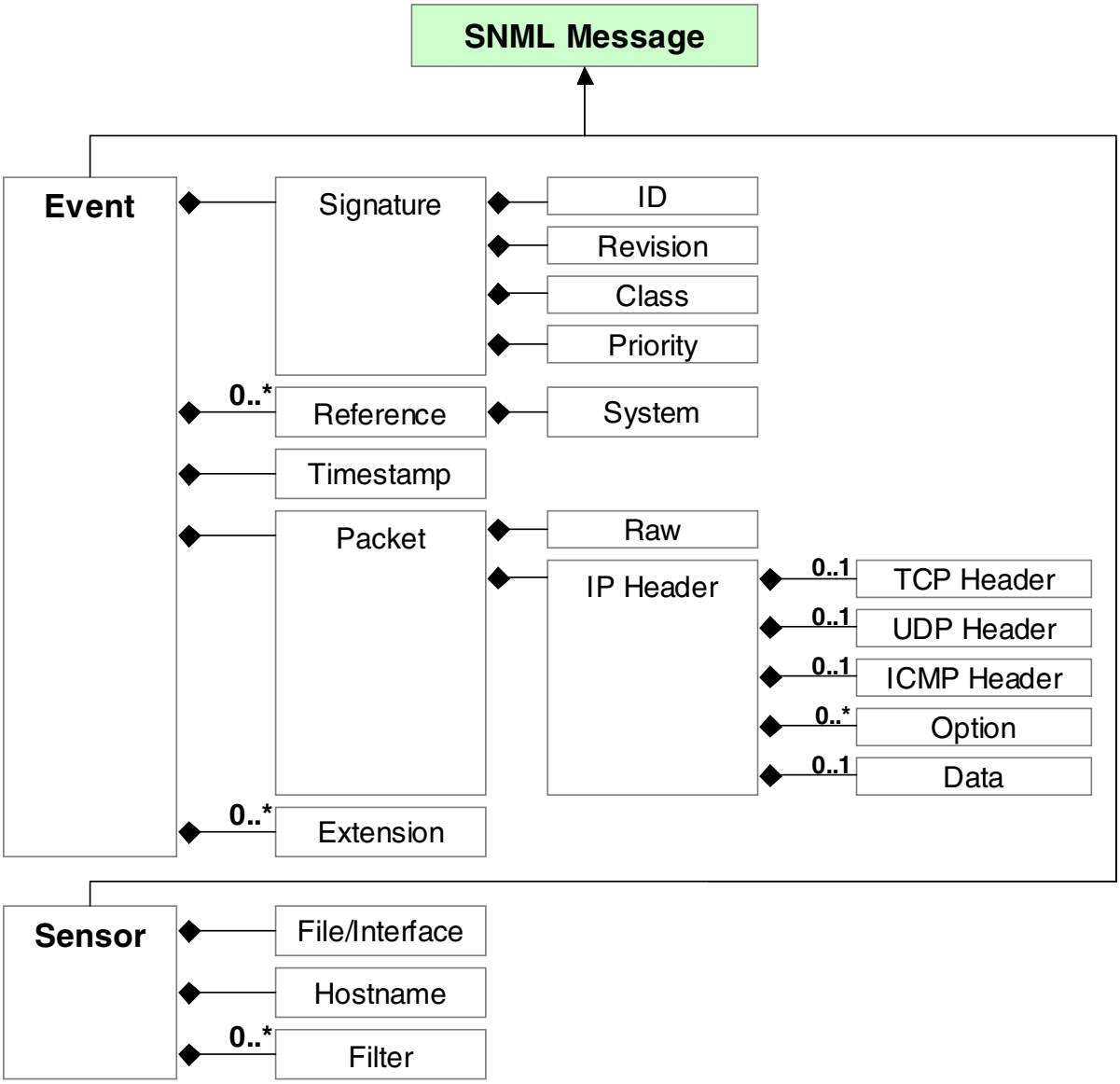


図 3.2: SNML のデータモデル

第4章　関連研究

本章では、本システムが目的とする複数システム間におけるセキュリティイベントデータの流通を実現し、インシデントレスポンスを支援する機能を実現する手法について議論する。

4.1　OPSEC

OPSEC(Open Platform for Security)[28] は、ファイアウォールベンダである Check Point 社が提供する API に基づき、さまざまなセキュリティアプリケーションと同社の製品を連携できるプラットフォームである。

OPSEC は現在、以下の四種類のカテゴリに属するアプリケーションと連携できる。

- セキュリティ強化
- 管理とレポート
- パフォーマンスと可用性
- e-ビジネス・アプリケーション

本研究の関連研究として、セキュリティ強化カテゴリにおいて、侵入検知に対する動的なアクションを実現する SAMP(Suspicious Activity Monitoring Protocol)[29] がある。OPSEC フレームワークにおいて、SAM Client と呼ばれる SAMP に対応した IDS 製品が SAM Server と呼ばれる SAM の管理サーバにアラートを送信し、アラートはファイアウォールのマネージャに転送される。最終的にはファイアウォール マネージャがファイアウォールのルールを変更する。

ただし、SAMP の規格によれば SAM Client がファイアウォールに通知できるデータは IP アドレスとアクセス遮断およびコネクション切断という限定されたアクションしかなく、本研究の目指すインシデントレスポンスの枠組みとしては利用できない。

また、この API はプロプライエタリであるため、OPSEC を利用するためには、Check Point 社と契約する必要がある。

4.2　swatch

swatch[30] は、syslog[31] のログファイルを通じてシステムの監視を行う Perl スクリプトである。

swatch を動作させるには、設定ファイルに監視対象の文字列のパターンと、当該パターンが発見された際のアクションを記述する。swatch は設定ファイルに基づき、syslogd の出力するログファイルを継続的に監視する。

監視対象パターンには Perl の正規表現を利用でき、また複数のパターンも記述できる。アクションには、外部コマンドの実行や、メールの送信、write コマンドの実行などが指定できる。

syslog が持つ複数のデバイスのログを集中管理できるという特徴を活かし、swatch は、早い時期から複数システムのログデータを一括して管理するモデルを提示した。

しかし、syslog は文字列を平文で送受信するプロトコルであり、伝送内容の暗号化、通信先の認証といった今日のネットワークセキュリティイベントを扱う上で必要とされる要求事項を満たしていない。また、syslog の内容は XML のような構造化された文章ではなく、単なる文字列に過ぎないため、設定作業が容易ではない。

swatch の実装上の問題点としては、設定ファイルの内容変更には再起動が必要であることが挙げられる。このため、動的に変化するネットワーク環境の情報を設定ファイルに動的に反映できない。

4.3 cfengine

cfengine[32] はポリシを記述可能な言語に基づく自律的なシステム管理ツールである。

システム管理者が行う作業を高水準言語として記述することにより、cfengine がその操作を代行できる。例えば、ネットワークインタフェースの状態の確認や設定変更、テキストファイルの編集、シンボリックの作成と管理、ファイルの所有者の確認と維持、外部コマンドの実行といった作業を設定ファイルに記述し、正常な状態を定義しておくことで、システムが異常状態になった場合でもシステムをあらかじ定義された状態に “収束” させることができる。

cfengine は外部からのデータ入力を処理するために設計されたものではなく、ホスト内のシステムの振る舞いに基づいて動作するように設計されている。従って、複数のシステムにおいて問題が発生するインシデントレスポンスの枠組みとして直接利用できない。しかし、複数ホストに渡るシステム管理とインシデントレスポンスにおいては、システム間のセキュリティ確保やポリシの記述、作業の自動化といった面で共通点があり、システム管理の抽象化およびスクリプトの記述方法が参考になる。

4.4 Network Flight Recorder

NFR(Network Flight Recorder) [33] は、N-code と呼ばれるスクリプト言語を通じて柔軟なフィルタリングが可能なネットワーク型 IDS である。

N-code は元々UberMUD というマルチユーザダンジョンゲームで利用されていたプログラミング言語から派生したもので、フロー制御や手続き、変数などを備えたプロ

グラミング言語である。

NFR では syslog といったアプリケーション層のデータを解析し、そのデータに基づいてスクリプトを記述できる。N-code による SPAM 送信の検知例を示す。

```
if ( $sender == $known_spammers )
{ ...
if ( $n_recipients > 10000 )
{ ... / possible spam
```

N-code では通常のデータ型に加え、IP アドレスをはじめとするデータ型を備えており、セキュリティイベントを表現するのに適している。また、ip.src や ip.dst のような書式を通じて、パケットのデータに対してアクセスできる。

NFR はネットワーク型 IDS であり、複数システムの連携を想定していない。しかし、セキュリティイベントに対するルールの言語体系を設計する上で、セキュリティイベントの表現方法が参考となる。

4.5 ArcSight

ArcSight[34] は複数のデバイスからログやセキュリティイベントデータを収集し、統合的にリスクを評価するソフトウェアである。

それぞれの相関関係に基づいたルールを記述することで、リスクを評価し、真の脅威となるイベントを絞り込める。

本研究は ArcSight と独立に進めてきたが、さまざまなデバイスからセキュリティイベントを収集するというコンセプトは共通である。

4.6 AirCERT

AirCERT は Automated Incident Reporting の略で、カーネギーメロン大学の Software Engineering Institute で開発された、分散環境においてセキュリティに関するデータを共有するシステム [35] である。このシステムでは IDS などから得られるセキュリティイベントのデータを複数組織間で交換し、人間による評価することを目的としている。

AirCERT フレームワークでは IODEF、IDMEF および SNML といった XML ベースのフォーマットを用いて情報交換する。また、このフレームワークではデータの不要部分の削除、正規化、共有といった手順を自動化し、大規模環境下において複数組織間の連携および協調することが考慮されている。

4.6.1 データ正規化コンポーネント

データ正規化コンポーネントは、様々なデータソースから得られる独自データ形式を AirCERT フレームワークで用いられている形式に標準化するコンポーネントである。この変換過程においては、データの再形式設定や意味的翻訳を伴う。次節以降では、AirCERT をフレームワークについて述べる。

rex

rex は正規表現を通じてテキストファイルの正規化するツールである。各種のログファイルや syslog の出力を AirCERT フレームワークに準拠した XML 形式に正規化する。

tabula

tabula はデータベースのデータの正規化および出力するツールである。このツールはデータベースに対して SQL の SELECT 文を継続的に発行し、結果データセットのいくつかの桁からデータを選択し、AirCERT フレームワークに準拠した XML 形式に出力する。

spo_xml

spo_xml はオープンソースの IDS である Snort のデータ出力プラグインである。このプラグインを用いることで、Snort の侵入検知結果を XML 形式でファイルまたはネットワーク経由で出力できる。ネットワークへの出力には HTTP[36], HTTPS[37],TLS[38],TCP の各プロトコルがサポートされている。

Dredge

Dredge はデータの再転送エンジンであり、HTTP,HTTPS,TLS,TCP といったプロトコルを用いてファイルをサーバに送信する機能を持つ。他のツールによって生成された XML 形式のメッセージを転送することが想定されている。

4.6.2 データ収集コンポーネント

mod_air

mod_air は AirCERT フレームワークにおいて、データ正規化コンポーネントから得られたデータを収集するサーバを構築するためのソフトウェアで、Apache のモジュールとして実装される。mod_air は HTTP の POST メソッドを通じてセキュリティイベントデータを受信し、データベースへ格納する処理する。また、トランスポートにはクライアント認証機能を利用した HTTPS を用いる。

4.6.3 分析コンポーネント

ACID

ACID は PHP[39] と adodb[40] をフロントエンドに用いたセキュリティイベントの分析コンソールであり、データベースに蓄積されたセキュリティイベントのデータに対して問い合わせを行い、管理者によるセキュリティインシデントの分析作業を支援する。

第5章　ネットワーク環境情報

本章では、ネットワーク環境情報という概念について説明する。

本研究では、ネットワークと各ホストに関するデータから把握できる情報をネットワーク環境情報と定義し、IDMEF や SNML で定義されたデータ以外のデータについてもセキュリティイベントの処理の際の判断基準の一部として利用することを提案する。

ネットワーク環境情報は、個々のセキュリティイベントに登場する存在の背景情報である。本章ではネットワーク環境情報を攻撃者側、被害者側の両側面、そしてインシデントで用いられた手法の側面からとらえ、各側面におけるネットワーク環境情報の役割と、その収集方法の一例について述べる。

管理者の暗黙知である情報をネットワーク環境情報と定義し、インシデント対応システム間で流通させることで、コンピュータがインシデントレスポンスを行う際に不足していた情報を補える。

5.1　主要な要素

インシデントに関係したエンティティにおいて、どのようなリスクが存在しているかを把握することで、はじめて的確なインシデントレスポンスが可能となる。

第 3.2.1節で述べたとおり、IDMEF は侵入検知メッセージのデータモデルを定義している。しかし、IDMEF や SNML で定義されたデータだけでは、インシデントレスポンスを行う上では不十分である。

以下に IDMEF の Alert クラスに属する要素を整理し、各データについて述べる。

5.2　時刻

セキュリティイベントに関連した日時を表現する。IDMEF ではメッセージを生成した時間の情報については `CreateTime` クラス、アラートのきっかけとなったイベントを検知した時間については `DetectTime` クラス、アナライザの現在時刻を `AnalyzerTime` クラスで定義している。

5.3 攻撃元

セキュリティインシデントの起点である。IDMEF ではイベントの起点を Source クラスとして定義している。Source クラスのサブクラスには、Node,User,Process,Service がある。

以下に攻撃元におけるネットワーク環境情報のデータと想定利用例を挙げる。

5.3.1 OS データ

OS の種類とバージョンに関するデータである。

たとえば Blaster ワームは Windows 2000 と WindowsXP に感染して、攻撃を行うが、その他の OS には感染しない。したがって攻撃元の OS のデータは、セキュリティインシデントの原因がワームにあるのか、あるいは人の手にあるのかを推測する上で役立つ。

データをリモートから取得するには、OS Fingerpriting ツールを利用できる。OS Fingerpriting は、IP パケットのデフォルト TTL 値や TCP のウィンドウサイズなど、スタックの実装の違いから OS 名を類推する技法である。

5.3.2 IP アドレスデータ

攻撃元 IP アドレスは、インシデントを特徴付ける大きな要素である。

IP アドレスデータを入手できれば、whois を通じて当該 IP アドレスを利用している国名、逆引き DNS サーバ名、ネットワーク管理者の連絡先など、インシデントに対応する際に必要な情報を入手できる。

ただし、IP アドレスは容易に偽装できる。ネットワークの境界ルータにおいて Ingress Filtering[41] を実施していないサイトを経由してパケットが届く場合や、同一セグメント内からの届いたパケットは、IP アドレス自体が偽装されている可能性がある。

5.4 標的

セキュリティインシデントの終点である。IDMEF ではイベントの終点を Target クラスとして定義している。

5.4.1 OS データ

OS データは特定の種類のバージョンの OS や、その OS が用いている TCP/IP スタックが抱える特有の脆弱性が存在した場合の脅威を判断する際に役立つ。

たとえば、Apache HTTP Server はマルチプラットフォーム環境で動作する Web サーバソフトウェアである。しかし、2002 年 8 月に発見された脆弱性 [42] は Apache 2.0 か

ら 2.0.39 の Windows 版のみが影響を受ける。そのため、攻撃を検知した際に、OS データを有していれば、その脅威の大きさを評価できる。

OS のバージョンをローカル環境から調査するには、UNIX 系 OS の場合には `uname` コマンドを実行する。Windows の場合は `winver` コマンドを実行する。

5.4.2 ポート番号データ

ホストでオープンしているポート番号のリストである。通常、ポート番号は稼働しているサービスと関連性がある。しかし、特定のポート番号を Firewall で通過可能にしている設定では、そのポート番号を用いるようにバックドアが仕掛けられた場合に通信を遮断することができない。

データの取得方法はコマンドによるローカル調査およびポートスキャンによるリモート調査があるが、ipchains プログラムやインターネット接続ファイアウォールといったシステム上のパケットフィルタなどの影響を考慮し、実質的に外部からアクセス可能なポートを調査するためにはリモート調査がよい。

UNIX 系 OS および Windows の場合は listen 状態のポートを調査するには `netstat` コマンドを実行する。また、プロセス毎のポート番号の使用状況を確認するには、UNIX 系 OS の場合には `lsof`[43] ツール、Windows XP 以降の `netstat` コマンドの場合に引数 “-o” が利用できる。

5.4.3 ソフトウェアおよびバージョンデータ

サーバソフトウェアのバージョンのデータである。脆弱性を持つバージョンのサーバソフトウェアの利用状況やパッチの適用状況の調査により、攻撃を受けた際の影響を判断できる。

例えば、Windows でのみ稼働する Web サーバである IIS[44] を狙った攻撃として、FrontPage Server Extensions のバッファオーバーラン攻撃 [45] がある。この攻撃はパッチを適用していない IIS のみ影響を受けるため、Web サーバとして Apache を稼働するサイトは影響を受けない。

ローカル環境からインストールされているプログラムのパッチの適用状況を確認するには、個々のプログラムごとに用意されているバージョン表示コマンドを用いる。OS 側でソフトウェアパッケージ管理システムを利用している場合や、パッチ適用状況確認プログラムを用意している場合には、それらを利用できる。RedHat 系 Linux の場合には `rpm` コマンド、Windows の場合には `Microsoft Baseline Security Analyzer` [46] が利用できる。

5.4.4 入出力デバイスデータ

ホストの入出力デバイスのデータである。従来より各種ディスクドライブや USB スロットなど、様々なメディアを通じて情報が盗難されたりバックドアを仕掛けられたりする事件が報告されている。最近ではビデオカメラやマイクロフォンが接続されたホストにおいて、Spyware と呼ばれるソフトウェアによってリモートから利用者の気づかないうちに盗撮・盗聴が行われることがある。従って、これらの入出力デバイスとその役割に関するデータが必要である。

5.4.5 信頼関係データ

ホストやネットワーク間の信頼関係についてのデータである。古典的な例では、.rhost を用いたホスト間の認証関係を悪用し、あるホストを乗っ取られた後にそのホストを信頼している他のホストも次々に乗っ取るというケースがある。また SMTP サーバにおいては、特定のネットワークやドメイン名からはメールを無制限に中継を許可することがあるが、そのネットワーク内に迷惑メールの送信者が存在した場合や、DNS の詐称が行われた場合には、問題が発生することになる。

ホストやネットワーク間の信頼関係を明らかにすることは、セキュリティインシデントの拡大範囲を推測するのに役立つ。

5.4.6 言語データ

ホストに設定されている言語データもネットワーク環境情報を構成するデータである。多くの OS においては、システムを利用する際に用いる言語を選択できるようになっているが、特定の言語設定やローカライズ版のソフトウェアがセキュリティインシデントの引き金となる可能性が想定される。

例えば、2003 年 8 月に流行した Nachi Worm は、ソースコードのコメントから日本語環境を狙って作成された [47] といわれている。実際、Worm が英語版、韓国語版、中国語版 Windows について感染した場合には修正版プログラムを適用するものの、日本語版 Windows においては脆弱性を放置して感染を広げる活動を行っている。

5.5 ノード識別子

ノードを識別する上で関するデータである。例えば以下のような項目がある。

5.5.1 物理層データ

ノードの所在地や接続先機器など、物理層に関するデータである。セキュリティインシデントの対応おいては、物理的セキュリティについて考慮する必要があるため、ホ

ストの位置情報を得ることは有益である。

例えば、外部からの侵入者が LAN 上のファイルサーバに不正にアクセスするために、物理的に侵入してネットワークに接続した場合には、当該ノードの MAC アドレスを接続しているスイッチのポートから、侵入者の位置を突き止めることができる。

所在地に関するデータとしては、有線 LAN であればスイッチの収容ポート番号、無線 LAN であれば収容基地局とチャネルとシグナルのデータなどが想定される。これらのデータは、攻撃者もしくは被害者の物理的な位置の特定に役立つ。

5.5.2 メディアデータ

ノードがネットワーク接続に用いているメディアとリンクのデータである。例としては、IEEE 802.3 シリーズのような有線 LAN なのか、802.11 シリーズのような無線 LAN なのか、あるいは ISDN、ATM、フレームリレー、PSTN なのかといったメディア種別データである。

メディアの種類が明らかになることで、共有メディアか、Point-to-Point といったリンクの形態が判明し、ネットワークに対する盗聴のリスクや成りすましの手法を推測する上で役立つ。

データリンクに Ethernet を用いている場合、ネットワークに接続されたホストを識別する際には MAC アドレスが必要になる。また、IEEE 802.1q や Cisco ISL のような tagged vlan を用いている場合、物理的なネットワークと論理的なネットワークを区別して取り扱う必要がある。

インシデントに関係したリンクの帯域データは、(D)DoS 攻撃のような帯域を埋め尽くすことを狙った攻撃に対してどの程度耐性を有しているのかの参考になる。

上記に挙げたようなレイヤ 2 データを取得することにより、レイヤ 3 以上のデータでは窺い知ることのできない、IP アドレススプーフィングのようなセキュリティイベントが明らかにできる。

5.5.3 IP アドレスデータ

標的の IP アドレスデータは、セキュリティインシデントに関連したノードを把握する際の必須データである。IP アドレスは全ての IP パケットに含まれるデータであるため、容易にデータを入手できる。

なお、DHCP や PPP によるダイヤルアップ接続などにみられる動的 IP アドレス割当機構を導入しているネットワークにおいては、単一の IP アドレスが時間の経過に伴い、異なるノードに対して割り当てられるケースがある。このため、IP アドレスデータだけでは確実にノードを特定できない。IP アドレスからノードを特定するには、IP アドレスデータに加え、時刻や IP アドレス割当システムのログデータが必要となる。

5.5.4 SMB データ

Windows ネットワークにおけるセキュリティインシデントの特定には、SMB に関するデータが必要になる。

例えば NetBIOS ホスト名、ワークグループ名もしくは Windows ドメイン名、ログインユーザ名、共有名が該当する。

5.5.5 アカウント名データ

ネットワーク接続において PPP 接続や VPN 接続、認証 VLAN などの認証を用いた形態で接続している場合、その認証に用いられたアカウント名をインシデントの範囲やホストを特定するデータとして利用できる。例えば、認証 VLAN を用いているキャンパスネットワークにおいてホストがワームに感染した場合、当該ホストの接続に用いられたアカウントのデータを通じて対応策を講じることができる。

5.5.6 ハードウェアデータ

ホストのハードウェアに関するデータである。例えば CPU に Pentium 4 を搭載しているか、あるいは UltraSparc III を搭載しているかといったデータがあれば、x86 系 CPU を狙った Shell Code のリスクを判断する上で役立つ。

5.5.7 利用形態データ

ホストの利用形態に関するデータである。ホストがサーバとして利用されているのか、クライアントとして利用されているのか、あるいは特定ユーザのみが利用するのか、不特定ユーザが利用するのかといったデータが利用できる。

5.5.8 通信履歴データ

当該ホストがいつからそのネットワークに存在するのか、どの程度のトラフィックが定常的に存在するのかといった通信履歴データを蓄積しておくことで、アノーマリ的手法を用いて踏み台や DoS 攻撃等の検知に応用できる。

5.6 攻撃手法

5.6.1 Classification

IDMEF の `Classification` クラスはアラートの分類を定義するクラスである。

Classification クラスには、origin 属性があり、該当する脆弱性情報へのポインタを提示できる。

IDMEF では、origin 属性の内容に脆弱性情報データベースである BUGTRAQ[48] と CVE(Common Vulnerabilities and Exposures)[49]、もしくはベンダ独自 URL を指定できる。

5.6.2 Assessment

Assessment クラスは、イベントの影響度を表現するクラスである。影響度のデータは、インシデントトリアージの際に必須である。

イベントの影響度は、攻撃手法やホスト環境によって異なる。例えばポートスキャンというセキュリティイベントがあったとする。このイベントはホストに対して直接の影響を与えるものではないため、影響度は低いと定義できる。

5.6.3 プロトコル

攻撃に用いられたプロトコルデータである。プロトコルデータを利用することで、通信元のなりすましの可能性や、プロトコル自体のデザインエラーのリスクを判断できる。

たとえば、ICMP や UDP のようなプロトコルでは通信元を容易に詐称できる。しかし、TCP の場合には、3way handshake があるため、なりすましは比較的困難である。

第6章　セキュリティイベント対応システムの設計

本章では、セキュリティイベントおよびネットワーク環境情報に基づいたセキュリティイベント対応システム “Automated Incident Responce and Incident Reporting On Network”(以下、AirIRON) の設計について述べる。

6.1　設計概要

本システムにおいてはIDSのアラートやファイアウォールのログ、各種サーバのログ等をセキュリティイベントのデータソースとして活用する。さらに、ポリシに基づきインシデントレスポンスを支援する出力プラグインを呼び出す。ルールには、セキュリティイベントのデータだけでなく、ネットワーク環境情報を記述可能にする。

図6.1に、本システムにおける各コンポーネントの関係を示す。

6.2　設計要件

本節では、ネットワークセキュリティイベント対応システムの設計要件について述べる。

セキュリティイベントは、第2.1節で述べた通りセキュリティインシデントの構成要素である。本節では、セキュリティイベントとインシデントに共通の要件、およびセキュリティイベント特有の要件を明らかにする。

6.2.1　即時性

即時性はセキュリティイベント処理手順全般に渡って求められる要件である。本システムにおいて即時性を考慮する範囲は、イベントの収集に要する時間、イベント処理に要する時間、イベントに基づく対応を実施する時間、また各コンポーネント間の伝送に要する時間である。

また、本システムはネットワークの中継ノードに特化したシステムではない。従って、個々のパケットの通過の可否の決定は即時性の要件の範囲外とする。

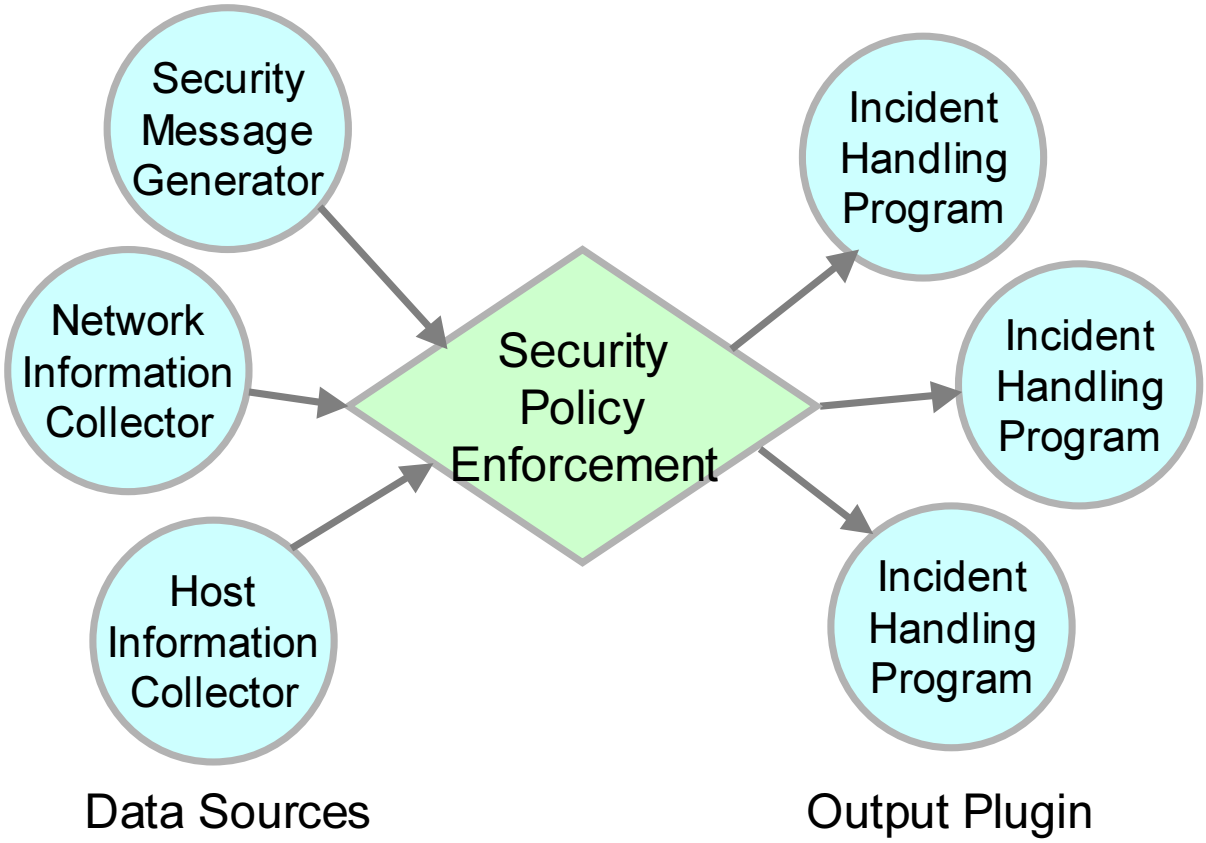


図 6.1: 本研究のモデル

6.2.2 ポリシの反映

セキュリティインシデント対応には、ポリシに基づいてセキュリティイベントを扱うことが求められる。

例えば、一時的な営業損失が発生するとしても、セキュリティホールを修正するというポリシがあるとした場合、ファイアウォールによりセキュリティホールのあるサービスに関連した通信を遮断するルールを記述できる。

また逆に、どんなセキュリティホールが存在しても、営業を継続するというポリシであれば、ファイアウォールにはセキュリティホールの存在するサービスに関連した通信を許可するルールと、IDS による監視のみを行うというルールを記述する。

また、ルールは動的に適用できなければならない。新たな脆弱性情報の公開やネットワーク内部へのノートパソコンの持ち込みなど、ネットワークにおけるリスク要因は流動的である。例えば、特定の IP アドレスが Windows であることを想定したルールを記述している場合には問題が生じる可能性がある。DHCP を導入している環境では、時間の経過とともに、ある IP アドレスが異なるホストに割り当てられるためである。

このように、セキュリティポリシや AUP といったさまざまなポリシを反映するには、内容に応じた対応の実施をルールとして記述できなければならない。また、セキュリ

ティイベントに対するトリアージの対応基準や、ネットワーク環境情報に基づいたセキュリティイベントの扱いを行うためには、ルールファイルを動的に反映できなければならない。

6.2.3 証拠の収集

セキュリティインシデントに関する証拠として、ファイアウォールやサーバのログや、脆弱性監査ツールのデータ、OS 情報などがあげられる。有効な攻撃を含むセキュリティインシデントを判断するために、これらの証拠を収集し、各イベントを統合的に関連づけること必要がある。

セキュリティイベントは、必ずしもセキュリティインシデントに結びつく情報であるとは断定できない。特に、単一のデバイスから得られたセキュリティイベントだけでは、セキュリティインシデントの証拠としては不十分である。例えば、既存の IDS の出力には、実際のセキュリティ上の脅威ではないデータが大量に含まれており、運用上の大きなコストとなっている。

IT 市場調査会社のガートナーは、IDS は投資に見合う効果をもたらさず、2005 年までに市場がなくなるとしたレポート [50] を発表した。このレポートでは、IDS の疑陽性と疑陰性の問題、24 時間 365 日の監視体制による管理者への負荷、過重なインシデントレスポンスの運用負担などが指摘されている。

ガートナーのレポートで指摘された IDS の問題点を解決しつつ、インシデントの裏付けを行うには、管理者の介入なしにインシデントに関わる証拠を収集しなければならない。

6.2.4 規模性

本研究のモデルでは複数のデバイスをデータソースとして活用するため、従来は単一のイベントとして観測されていた事象が、複数の事象として観測されるケースが出てくる。従って、本システムは多数のセキュリティイベントを処理できる規模性を備える必要がある。

また、ネットワークの拡大に伴い、単一のセキュリティデバイスで対応できる範囲の限界が問題となっている。例えば、ネットワークの広帯域化に伴い、単独の NIDS やファイアウォールではネットワークの帯域を全て検査対象とすることが不可能となってきたため、ロードバランサー [51],[52] を導入し、複数の NIDS やファイアウォールにトラフィックを分散して処理させるネットワーク構築手法がある。

この場合には、セキュリティイベントデータが分散するが、複数のデバイスに分散したセキュリティイベントを集約し、統合的に管理することで、ネットワーク全体に対する規模性を確保できる。

6.2.5 多彩なセキュリティデータソースの活用

できるだけ多くのセキュリティイベントを収集するには、多彩なデータソースの活用が必要である。

今日、ネットワークに接続されている様々なデバイスには、程度の差はあれログ取得機能を備えているデバイスが多い。本研究では、ログなどのセキュリティに関連したデータを収集できるデバイスをセキュリティデータソースとして定義し、これらのデバイスからセキュリティイベントを幅広く収集することで、これらのデータがセキュリティインシデントレスポンスに役立つ。

具体的な例としては、ネットワークトラフィックを監視を目的とした IDS、ログ機能を有するファイアウォールや Access Control List(ACL) 機能を実装したルータ、あるいは Web サーバやメールサーバといったログ取得機能を備えたサーバソフトウェアがある。

このように、セキュリティデータソースは必ずしもセキュリティイベントを収集する目的で設置されたデバイスであるかを問わず、幅広いデバイスからログを収集できることが重要である。

6.2.6 複数システムのサポート

前節で述べた多彩なデータソースを活用するためには、複数システムからのセキュリティイベントデータの収集機能が求められる。複数システムからのデータ収集には、多様なデバイスからデータを収集できるという利点に加え、負荷分散を行っているデバイスから収集したデータを一元的に管理できるメリットがある。

例えば、現在のところ 1Gbps の帯域幅を持つネットワークにおいて、ワイヤスピードの処理性能を持つ IDS は少ないため、負荷分散機を用いてトラフィックを複数に分散させ、IDS に処理させることも少なくない。このような環境においては、複数システムから収集したデータの統合的な管理により、監視対象のネットワーク全体の動向が把握しやすくなる利点がある。

6.2.7 システムの認証

複数システムをサポートする際の課題として、セキュリティデータソースから得られたデータが本当に正しいデータソースから得られたデータであるのかという保証、また通信経路上における盗聴の危険に対応することが求められるため、それぞれデータソースに対する認証システムと暗号化システムを提供する必要がある。そのため、本研究では TLS を用い、通信相手の認証と通信内容に対する暗号化機能を提供する。

6.3 データフォーマット

インシデントレスポンスの過程においては、インシデントを発見に必要なデータの入手が重要な鍵となる。そのためには、IDS やファイアウォール、各種サービスのログといったセキュリティデータソースからセキュリティイベントのデータを幅広く広く入手する必要がある。

その際、異なるセキュリティデータソースからセキュリティイベントデータを収集する際には、相互運用性が大きな課題となる。相互運用性を確保するためには、データフォーマットを統一が必要となる。このような課題に対しては、XML[53] を用いることで異なるデータフォーマットに共通の書式を利用できる。

第 4章において述べたように、複数のセキュリティシステム間のデータ交換の必要性は広く認識されつつあり、IETF の Intrusion Detection Exchange Format Working Group においては、XML をベースにした IDMEF 規格の標準化が行われている。

本研究では IETF において提唱されている IDMEF、AirCERT プロジェクトの定義する SNML(Simpled Network Markup Language) を DTD(Document Type Definition) とする XML フォーマットに対応し、これらの DTD に対応したシステムをデータソースとして活用する。

本研究においては、IDS 以外のデバイスをセキュリティデータソースとして活用できるようにするため、IDMEF および SNML のデータモデルから共通の要素を抽出する。これらのデータをセキュリティイベント基本データと定義し、全てのセキュリティデータソースから取得する。具体的なデータを以下に示す。

- システム識別子 (センサもしくはアナライザ)
- タイムゾーンを含む日時
- 送信元 IP アドレスとプロトコル、ポート番号
- 送信先 IP アドレスとプロトコル、ポート番号
- イベントの分類
- イベントの評価
- 追加データ

なお、セキュリティインシデントレスポンスにおいては、これらのデータは必要条件であり、十分条件を意味するわけではない。インシデントレスポンスの際に必要とされるデータについては、第 5節で検討する。

```
<IDMEF-Message version="1.0">
  <Alert ident="abc123456789">
    <Source ident="s01">
      <Node ident="s01-1">
        <Address category="ipv4-addr">
          <address>127.0.0.1</address>
        </Address>
      </Node>
    </Source>
  </Alert>
</IDMEF-Message>
```

図 6.2: IDMEF における IPv4 アドレスの表現 (抜粋)

6.3.1 セキュリティイベントの表現方法

さまざまなデータソースからデータを収集するためには、既存の IDMEF では定義されていないデータの表現方法を検討する必要がある。本システムにおいては、IDMEF との互換性を損なわない形でデータを表現する。

例えば、IDMEF のデータモデルにおいて表現できないデータに関しては、ADDITIONALDATA クラス以下に属性を設けることでデータを表現する。例えば、TCP のシーケンス番号は、$ADDITIONALDATA.TCP.SEQ として表現する。

また上記のデータを表 6.1に示す環境変数名に代入し、セキュリティイベントの各データや、ネットワーク環境情報を活用できる。すなわち、第 6.4.2節で述べるセキュリティイベント処理システムにおいて、これらのデータをスクリプトから扱えるようになる。

本システムで用いる環境変数名の設計では、基本的に IDMEF のデータモデルに沿って変数名を付与する。例えば、攻撃元ノードの IPv4 アドレスは IDEMF において図 6.2のように記述する。

6.3.2 ネットワーク環境情報の表現方法

ネットワーク環境情報はルールの一部として利用される。ネットワーク環境情報はそれ自体がセキュリティに関連したデータを含むわけではないので、セキュリティイベントとして扱うことは適当ではない。しかし、第 6.2.5 節で述べた通り、多彩なデータソースのデータを一元的に記述するためには、セキュリティイベントに含まれるデータと同じように扱う必要がある。

そこで、個々のネットワーク環境情報を環境変数名に代入することで、セキュリティ

```
<rule set var="ENV_OS_203_178_143_1" value="NetBSD_1_8" />
<rule set var="ENV_OS_203_178_143_2" value="Linux_2_4" />
<rule set var="ENV_OS_203_178_143_3" value="Windows_2000_SP2" />
```

図 6.3: ルールにおけるネットワークの定義

```
<rule if expr="$SOURCE_NODE_ADDRESS_IPV4_ADDR = 203.178.143.1" &&
 expr="${ENV_OS}.${SOURCE_NODE_ADDRESS_IPV4_ADDR} = Windows_200" &&
 expr="$CLASSIFICATION_NAME = ATTACK-TO-WINDOWSHOST " />
<rule exec cmd="/usr/local/bin/realalert" />
<rule endif />
```

図 6.4: ネットワーク環境情報を用いたルールの記述例

イベントと同じように扱える。例えば OS Fingerprinting により入手した OS データというネットワーク環境情報を図 6.3のように定義することで、図 6.4に示したように、ルールにおいてネットワーク環境情報を活用可能にする。ネットワーク環境情報の活用方法は、攻撃元と標的で異なる。しかし、そのデータ自体は攻撃元でも標的でも変らないため、ネットワーク環境情報の表現においては、前者と後者を区別せずに扱う。

6.4 ポリシとルール

インシデントレスポンスの方法はセキュリティポリシや AUP、あるいは法規制に大きく依存する事実を第 6.2.2節において指摘した。従って、インシデントレスポンスの一手順であるセキュリティイベントの処理もこれらのポリシに従って行う必要がある。

ポリシの実効性を確保する上では、ネットワークの利用者がポリシに従うようにできるか、またネットワークの運用体制が運用ポリシに従うようにできるかを検討する上では、ルールをどのように記述するかが鍵となる。

6.4.1 ルール

ルールはネットワークの設計や運用における各段階において適用することが肝要である。

例えば、ある組織において組織外とのファイル共有を禁止するというポリシを適用する場合、ファイアウォールにおいてはファイル共有アプリケーションが利用するポートを遮断する、あるいはペイロードに基づきパケットを破棄するルールを記述する。ま

た、IDS においては当該アプリケーション特有の通信に対してルールを書くといった実装ができる。

本研究のフレームワークにおいては、あらかじめ定義されたセキュリティイベント等をトリガーとし、セキュリティインシデント対応を行う出力プラグインを実行するルールセットにより、ポリシを反映できるようにする。

6.4.2 イベント処理システム

本研究ではポリシを記述できるスクリプト言語 “IronScript” を設計した。“IronScript” では、データソースから得られたセキュリティイベントごとに設定されたルールを解釈し、インシデント対応を支援するアプリケーションを呼び出すことで、セキュリティポリシを反映できる。

ルールにおいては、セキュリティイベントおよびネットワーク環境情報を評価対象として記述できるよう、各イベントごとにセキュリティデータソースから得られたデータおよびネットワーク環境情報の一部を環境変数として定義し、スクリプトを実行することで、個々のセキュリティイベントに対して処理できるようにする。

例えば、ワームに感染したというセキュリティイベントを受信した場合に、当該ホストをネットワークから切り離す機能を持った出力プラグインを実行するというルールを記述できる。

6.5 セキュリティイベントの蓄積

セキュリティインシデントレスポンスの対応においては、信憑性と即時性という一見相反する要件を満たさなければならない。信憑性を満たすためには、セキュリティインシデントに関するセキュリティイベントデータが必要であるため、データの収集および解析のために即時性が損なわれる。一方、即時性を満たす際には、十分なデータの収集が困難になりがちである。

そこで、データベースに個々のセキュリティイベントを蓄積する必要がある。フォローアップ過程において、即時性を重視した対応とは別に、データベースに蓄積されたデータに基づいて詳細な解析をできるようにする。

6.6 設計のまとめ

本章においては AirIRON システムの設計を行った。本システムではさまざまなデータソースからセキュリティイベントおよびをネットワーク環境情報を収集し、ルールに基づきいて各イベントを処理する。またインシデント対応のためのプログラムを実行できる。

表 6.1: データソースから代入される環境変数名

データ	環境変数名
IP 関連データ	
システム識別子	$ANALYZER_ANALYZERID
タイムゾーンを含む日時	$CREATETIME
送信元 IP アドレス	$SOURCE_NODE_ADDRESS_IPV4-ADDR
送信先 IP アドレス	$TARGET_NODE_ADDRESS_IPV4-ADDR
上位プロトコル	$TARGET_SERVICE_PROTO
IP バージョン	$ADDITIONALDATA_IP_VER
IP ヘッダ長	$ADDITIONALDATA_IP_HLEN
全データ長	$ADDITIONALDATA_IP_LEN
TTL	$ADDITIONALDATA_IP_TTL
チェックサム	$ADDITIONALDATA_IP_CSUM
IP オプションコード	$ADDITIONALDATA_IP_OPT-CODE
IP オプション長	$ADDITIONALDATA_IP_OPT-LEN
IP オプションデータ	$ADDITIONALDATA_IP_OPT-DATA
フラグメントオフセット	$ADDITIONALDATA_IP_OFFSET
TCP 関連データ	
送信元ポート番号	$SOURCE_SERVICE_PORT
送信先ポート番号	$TARGET_SERVICE_PORT
TCP シーケンス番号	$ADDITIONALDATA_TCP_SEQ
TCP 確認応答番号	$ADDITIONALDATA_TCP_ACK
データオフセット	$ADDITIONALDATA_TCP_OFFSET
予約ビット	$ADDITIONALDATA_TCP_RESERVED
TCP フラグ	$ADDITIONALDATA_TCP_FLAGS
ウィンドウサイズ	$ADDITIONALDATA_TCP_WINDOW
チェックサム	$ADDITIONALDATA_TCP_CSUM
緊急ポインタ	$ADDITIONALDATA_TCP_URP
TCP オプションコード	$ADDITIONALDATA_TCP_OPT-CODE
TCP オプション長	$ADDITIONALDATA_TCP_OPT-LEN
TCP オプションデータ	$ADDITIONALDATA_TCP_OPT-DATA
UDP 関連データ	
送信元ポート番号	$SOURCE_SERVICE_PORT
送信先ポート番号	$TARGET_SERVICE_PORT
データ長	$ADDITIONALDATA_UDP_LEN
チェックサム	$ADDITIONALDATA_UDP_CSUM

データ	環境変数名
ICMP 関連データ	
ICMP タイプ	$ADDITIONALDATA_ICMP_TYPE
ICMP コード	$ADDITIONALDATA_ICMP_CODE
チェックサム	$ADDITIONALDATA_ICMP_CSUM
識別子	$ADDITIONALDATA_ICMP_ID
ICMP シーケンス番号	$ADDITIONALDATA_ICMP_SEQ
イベント関連データ	
イベントの分類	$CLASSIFICATION_NAME
アラートの情報源	$CLASSIFICATION_ORIGIN
イベントの重大性	$ASSESSMENT_IMPACT_SEVERITY
イベントの結果	$ASSESSMENT_IMPACT_COMPLETION
イベントの対象	$ASSESSMENT_IMPACT_IMPACTTYPE
要求される対応策	$ASSESSMENT_ACTION
その他のデータ	
データペイロード	$ADDITIONALDATA_DATA_PAYLOAD
シグネチャID	$ADDITIONALDATA_SRC_SIG_SID
シグネチャリビジョン番号	$ADDITIONALDATA_SRC_SIG_REV
シグネチャクラス名	$ADDITIONALDATA_SRC_SIG_CLASSNAME

第7章　ポリシに基づいたインシデント対応の実現

AirIRON フレームワークにおいては、大きく分けて以下の 3 つの部分に分類できる。

- データソース部
- セキュリティメッセージ処理部
- インシデント対応支援部

本研究においては、セキュリティメッセージ処理およびインシデント対応支援部分の実装を行った。本章ではセキュリティメッセージ処理部の実装について述べる。

7.1　実装環境

本研究では、複数のセキュリティイベントデータを扱うデバイスが存在するネットワークという想定環境において情報交換する際、AirCERT フレームワークが適していると判断し、本フレームワークを用いて実装を行った。

AirIRON のセキュリティメッセージ処理部は、Apache 1.3[54] のモジュールである mod_air を拡張して実装した。表 7.1に実装に用いたソフトウェア環境を示す。

表 7.1: 実装ソフトウェア環境

OS	Debian/GNU Linux testing (Kernel 2.4.23)
ソフトウェア	Apache 1.3.29 mod_ssl 2.8.14 mod_air 0.9.8 libair 0.3.37
RDBMS	PostgreSQL 7.4
プログラミング言語	C 言語
コンパイラ	gcc 3.3

7.2 実装概要

図 7.1に本実装の概要を示す。

セキュリティメッセージ処理部では、セキュリティイベントデータを解析し、ルールにイベントデータの内容をあてはめ、インシデント対応を支援する。

例えば、Apache HTTP Server を通じてセキュリティイベントデータを受信する。そして、受信した SNML や IDMEF といった XML 形式のデータを解析し、XML の各要素ごとの値を環境変数に代入する。またスクリプトを実行し、環境変数の内容に応じてプログラムを実行する。

図 7.1に、セキュリティイベントデータ処理部の実装概要図を示す。

7.2.1 データソース連携部分

セキュリティイベントは、HTTP/1.0 の POST メソッドを用いて Apache HTTP Server に送信させる。この際、拡張子 ".air" を含む URI がリクエストされた場合、mod_air が呼び出される。

Apache HTTP Server のモジュールでは、module 構造体の 8 番目の要素がイベントハンドラとして定義されている。Apache から mod_air に処理が渡された際には、ハンドラとして定義された clt_handlers 関数が呼び出される。

clt_handlers 関数からは、clt_read_content 関数が呼び出され、送信された内容の正当性を検査する。検査に合格したイベントは、clt_event_log 関数に渡され、IDMEF や SNML といったセキュリティイベントのデータフォーマットに従って処理される。

SNML 形式のセキュリティイベントを処理する場合には、まず clt_event_log_snml 関数が実行され、XML のノードからデータの抽出およびデータベースへを蓄積する。また、本拡張においては、第 6.3節で述べた通り、XML のノードから取得したデータを環境変数に設定する。

7.2.2 セキュリティメッセージ処理部

セキュリティメッセージ処理部分では、管理者が設定したルールの記述に従い、外部プラグラムを実行するセキュリティメッセージ処理部を実装した。

管理者はインシデント対応ポリシ記述言語 "IronScript" を用いてセキュリティイベントデータとネットワーク環境情報に対するポリシを記述できる。

なお、`mod_air_iron.c` は Apache 1.3.27 の配布パッケージに含まれる `mod_include.c` [55] に基づき実装を行った。そのため、Apache が実装している SSI[56] のディレクティブのほとんどを設定できる。

ルールファイルの設定は、Apache の設定ファイルの `httpd.conf` の `IronConfigFile` ディレクティブを通じて行う。下記の例に例を示す。

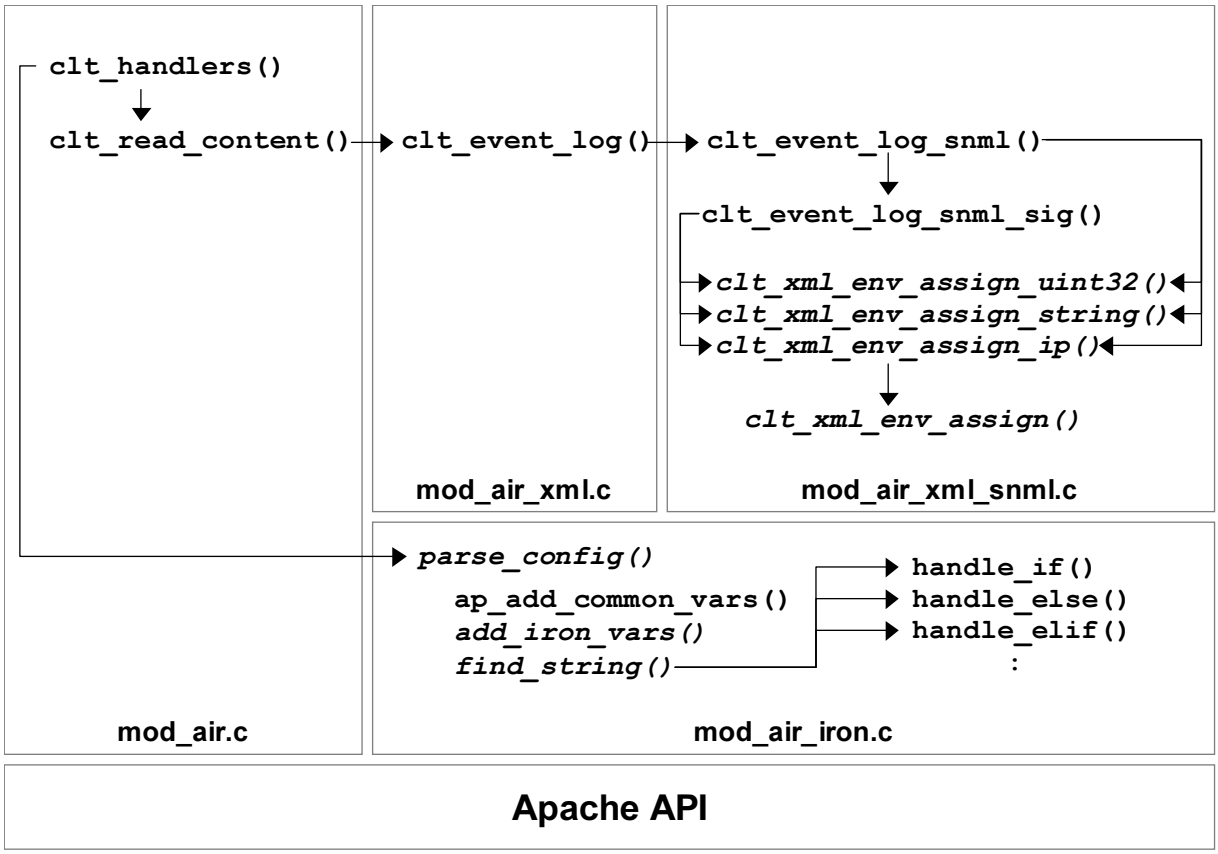


図 7.1: 実装: 環境変数を設定するイベント処理の流れ

```
IronConfigFile /etc/httpd/conf/iron.rules
```

上記の例では、/etc/httpd/conf/iron.rules ファイルをルールファイルとして定義している。

iron_parse 関数は、本システムにおいてルールファイルを解釈する関数である。

制御構文

ルールファイルにおいて利用可能な制御構文は以下の通りである。ルールの制御構文には、7.2.2節で述べた環境情報の値を利用できる。

以下の例に例を示す:

表 7.2: 制御構文

<rule if expr=”条件式” / >
<rule elif expr=”条件式” / >
<rule else / >
<rule endif / >

```
<?xml version="1.0" encoding="utf-8"?>
<config>
<rule if expr="$Alert.Classification.name = MS03-039_Unpatched" />
<rule exec cmd="/usr/local/bin/interfacedown $Alert.Source.Node.Addr
ess.mac" />
<rule else />
do nothing
<rule endif />
</config>
```

この例では、変数$Alert.Classification.name に MS03-039_Unpatched という値が含まれていた場合、$Alert.Source.mac の値を引数として/usr/local/bin/interfacedown プログラムを実行する。

関係演算子

ルールファイルにおいて、下記の文字列を関係演算子として利用できる。

表 7.3: 利用可能な関係演算子

関係演算子	意味
==	等しい
!=	等しくない
>	より小さい
>=	より小さいか等しい
<	より大きい
<=	大きいか等しい

関係演算子には正規表現を利用できる。正規表現を用いる場合には、/正規表現の文

字列/ のようにスラッシュで区切った範囲内で、マッチ条件を指定する。以下に例を示す。

```
<?xml version="1.0" encoding="utf-8"?>
<config>
<rule if expr="($Alert.Source.ipv4-addr = /\^203.178.143/) &&
($Alert.Classification.name = Blaster)" />
Blaster has been detected from RGNET!
<rule exec cmd="/usr/local/bin/change_acl $Alert.Source.ipv4-addr" />
<rule endif />
</config>
```

上記の例では、ソースIP アドレスを意味する$Alert.Source.ipv4-addr の値が 203.178.143 で始まるセキュリティイベントで、なおかつIDS のアラートの内容である$Alert.Classification.name の値が Blaster であった場合に、$Alert.Source.ipv4-addr の値を引数として /usr/local/bin/change_acl コマンドを実行する。

論理演算子

ルールファイルにおいて、論理演算子として利用できる文字列は下記の通りである。

表 7.4: 利用可能な論理演算子

論理演算子	意味
condition1 && condition2	論理積
condition1 || condition2	論理和
(condition1)	肯定
!condition1	否定

7.3 出力プラグインの実現

本研究ではインシデント対応を実際に行うプラグインとして、下記のモジュールを実装した。

7.3.1 出力プラグインに必要なデータ

出力プラグインの実現にあたっては、単にコマンドを実行できるだけでなく、インシデント対応に必要な情報をプラグインに対して提供可能でなければならない。

例えば、10.0.2.3 という IP アドレスを持ったホストが侵入を受け、DDoS 攻撃に荷担していることが判明したとする。出力プラグインがセキュリティインシデントへ対応するには、第 5.5節で述べたデータを有していなければならない。

本実装においては、セキュリティイベントデータから入手したデータを出力プラグインに引数を渡すことで、出力プラグインが引数からセキュリティインシデントへの対応や管理者への通知に必要なデータを得られる。

7.3.2 802.11 通信切断モジュール

802.11 通信切断モジュール “void11” は、IEEE 802.11b および IEEE 802.11g に準拠した無線 LAN ネットワークにおいて、特定の MAC アドレスを有するステーションに対して disassociation パケットを送信し、無線 LAN から切断するモジュールである。

```
void11 {mac_addr|ipaddr}
```

ワームに感染したホストが無線 LAN ネットワークに接続し、感染が拡大しているような状況において本プログラムを用いることで、当該ホストを無線 LAN から切り離すことができる。

引数には MAC アドレス、もしくは IP アドレスが指定できる。IP アドレスが指定された場合には、本プログラムが ARP テーブルを参照し、当該 IP アドレスに該当する MAC アドレスに対して disassociation パケットを送信する。

7.3.3 DHCP サーバ連携モジュール

DHCP サーバ連携モジュール “hoihoicfg” は、特定の MAC アドレスに対し、通常とは異なる IP アドレスおよびデフォルトゲートウェイ等を割り当てるよう DHCP サーバの設定を変更するモジュールである。

```
hoihoicfg {mac_addr|ipaddr}
```

このモジュールを通じて DHCP サーバの設定を変更し、指定された MAC アドレスを持つホストを “囲い込む” ことができる。そのため、通常のクライアントにはインター

ネットアクセスを許可しつつ、“囲い込まれた” クライアントに関してはインターネットとの通信を制限できる。

引数にはMACアドレス、もしくはIPアドレスが指定できる。IPアドレスが指定された場合には、本プログラムがARPテーブルを参照し、当該IPアドレスに該当するMACアドレスに対して設定を変更する。

7.3.4 ファイアウォール設定変更モジュール

ファイアウォール設定変更モジュール “fwconfig” は、パケットフィルタリングプログラムであるipchainsに対して、指定された条件に該当するパケットを破棄する設定を追加するプログラムである。以下に本プログラムの引数を示す。

```
fwconfig proto src_ipaddr src_port dest_ipaddr dest_port
```

各引数には、以下の環境変数を指定することが適切である。

表 7.5: データソースから代入される環境変数

引数	環境変数
proto	$Alert.Target.Service.proto
src_ipaddr	$Alert.Source.Node.Address.ipv4-addr
src_port	$Alert.Source.Service.port
dest_ipaddr	$Alert.Target.Node.Address.ipv4-addr
dest_port	$Alert.Target.Service.port

このモジュールはIDMEFのActionクラスのcategory属性において、block-installedが指定された場合に特に有用である。

7.3.5 メール通知モジュール

メール通知モジュール “alertmail” は、指定されたRFC2822形式のメールアドレスにメールを送信するプログラムである。以下に本プログラムの引数を示す。

```
alertmail mail_addr body
```

管理者にメールを送信する場合には、引数に$Env.Admin.Address.e-mail環境変数を用いることを想定している。

このモジュールはIDMEFのActionクラスのcategory属性において、notification-sentが指定された場合に特に有用である。

7.3.6 SMB Alert通知モジュール

SMB Alert通知モジュール “smbalert” は、指定されたIPアドレスに対してSMBアラートを通知するモジュールである。Messenger Serviceの稼働するWindowsワークステーションにSMBアラートが送信された場合、指定されたメッセージを表示する。

```
smbalert ip_addr message
```

このモジュールはIDMEFのActionクラスのcategory属性において、notification-sentが指定された場合に特に有用である。

第8章　本システムの評価

本章では、本システムの定性的評価および定量的評価について述べる。

8.1　定性的評価

本システムの設計の正当性を検証するために、設計概要で挙げた各項目について AirIRON の動作確認を行った。

8.1.1　ポリシの反映

セキュリティインシデント対応において、ポリシの反映できるようにするためにルールを記述可能にした。また、ルールを動的に更新可能にすることにより、常に流動的なネットワーク環境にも追従できた。

第 7.2.2節で述べた通り、本システムはルールファイルに基いてセキュリティイベントを処理できる。ルールファイルには制御構文を用いることが可能であり、ポリシに基づいたルールを記述できる。

本実装では、各セキュリティイベントを処理するごとに、ルールファイルを読み込み、ルールを解釈してするため、適用するルールを動的に更新可能である。

8.1.2　証拠の収集

設計したシステムは多彩なデータソースからセキュリティイベントデータの収集を行うことにより、セキュリティインシデント対応に十分な証拠の収集できる。

本システムは、dredge を通じて XML 形式のセキュリティイベントデータを収集できる。syslog 形式のデータや各種サービスのログなどに関しては、rex を通じて XML に変換することで、本システムで利用可能になる。

また、ネットワーク環境情報については、第 6.3.2節で述べた通り、ルールファイルでデータを定義する。データの収集には、各プログラムの出力結果を、ルールファイルの書式に変換するスクリプトを用いた。

第 6.2.3節で述べた通り、セキュリティイベントデータにはノイズ的なデータが多数含まれている。特に IDS に関しては誤検出が多く、そのデータからセキュリティインシデント対応に十分な証拠を得られないため、他のデバイスからセキュリティイベントデータを収集することで、インシデント対応に十分な証拠を確保できる。

8.1.3 規模性

本研究のモデルにおいては、複数のデバイスをデータソースとして活用するため、セキュリティイベント処理部に複数のシステムに対応できる規模性が必要になる。

本システムの実装は、Apache モジュールとして実装した。またトランスポートに HTTPS を用いたため、本システムのスケーラビリティは Apache の SSL 処理性能に依存する部分が大きい。ApacheCon 2000 にて発表された Apache の SSL 処理性能に関するデータ [57] によれば、Athlon 600MHz の Linux を搭載したマシンでは、毎秒 100 回の RSA 署名が実行できるため、十分な性能を確保しているといえる。

表 8.1: 要求事項

要求事項	要求事項解決の有無
ポリシの反映	○
証拠の収集	○
規模性	○

以上の結果から、本システムがインシデントに対応できる。設計したシステムがセキュリティイベント処理に要求事項であるポリシの反映、証拠の収集、規模性を満たすことを示した。

従って、本システムは、表 8.1に示す要求事項をすべて満たすといえる。なお、設計要件で述べた各項目のうち、即時性については次節にて定量的評価を行う。

8.2 定量的評価

定量的評価として、本システムの処理時間の計測を行った。図 8.2に、本システム全体の処理時間を示す。今回の試験では、このうち t_3 の値について計測を行った。

8.2.1 実験環境

図 8.2.1に、実験環境の概要を示す。

表 8.2に評価に用いたハードウェア環境を、表 8.3に評価に用いたソフトウェア環境を示す。

また、Apache およびモジュールの主要な設定パラメータを表 8.3, 表 8.4, 表 8.5に示す。表に記述のないパラメータについては Apache 1.3.29 の Debian パッケージのデフォルト設定に準ずる。

Apache は、シングルスレッドで動作させるために “-X” 引数を用いて起動した。データ転送プロトコルには、TLSv1 を用い、暗号には DHE-RSA-AES256-SHA (256 ビット) を用いた。

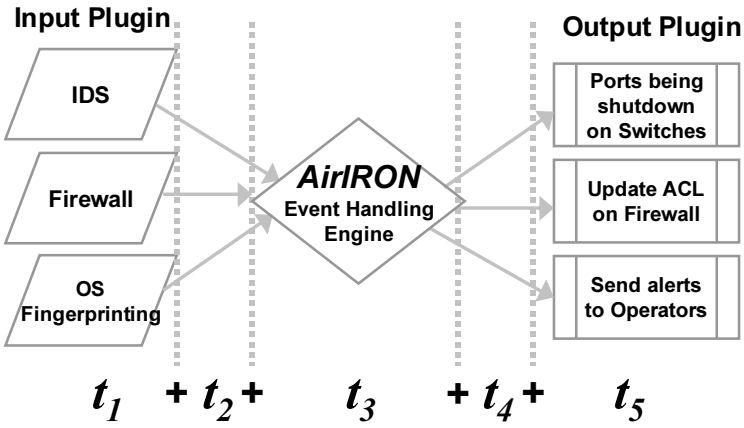


図 8.1: 本システムの処理時間

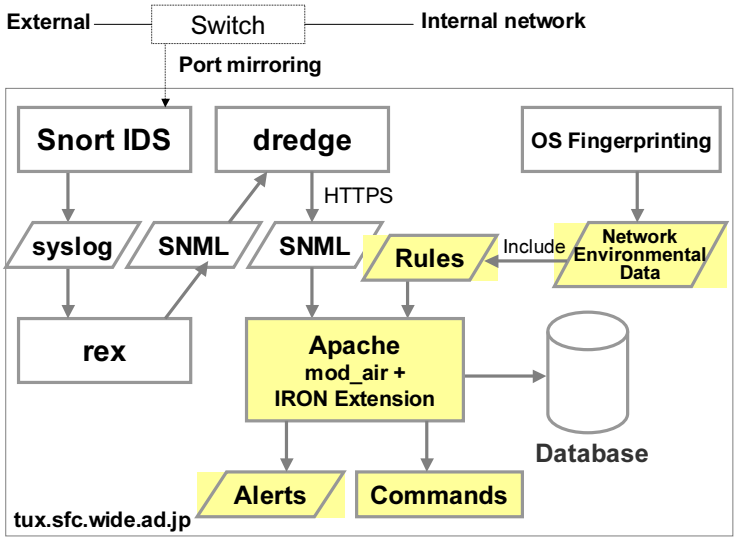


図 8.2: 実験環境の概要

表 8.2: ハードウェア環境

CPU	Intel Pentium III 800MHz
メモリ	512MB
ハードディスク	Ultra160 SCSI 18GB

ネットワークによる性能劣化の可能性を排除するため、クライアントとサーバを同一ホストで稼働させた状態で評価を行った。

また予備実験の結果、データベースにエントリ数が増大した際の応答時間の低下が

表 8.3: ソフトウェア環境

OS	Debian/GNU Linux testing (Kernel 2.4.23)
クライアントソフトウェア	dredge 0.2.5
サーバソフトウェア	Apache 1.3.29 mod_ssl 2.8.14 mod_air 0.9.8 + Iron Extension 20031217 libair 0.3.37
RDBMS	PostgreSQL 7.4

```
Timeout 300
HostnameLookups Off
MaxRequestsPerChild 10000
```

図 8.3: Apache 関連設定

大きいことがわかった。評価におけるデータベースの性能の影響を最小限に留めるため、実験ごとにデータを書き戻した。これにより各実験開始時のデータベース全体の内容が同一になるようにした。

評価対象

定量評価は、本システムがセキュリティイベントを処理するのに要した時間を計測した。測定は、以下の条件の元で行った。

- 処理内容を含まないルール
- 条件分岐を 500 回繰り返すルール
- 条件分岐を 1000 回繰り返すルール
- 条件分岐を 1500 回繰り返すルール
- 条件分岐を 2000 回繰り返すルール

また、比較対象として、mod_air 0.9.8 の性能計測を行った。mod_air はセキュリティイベントをデータベースに保存する機能を有するが、ルールに基づくセキュリティイベント処理機能を有しないため、仮説として処理内容を含まないルールと同等の性能を有するはずである。

以下にルールファイルの例を示す。

```
SSLEngine on
SSLPassPhraseDialog builtin
SSLCipherSuite ALL:!ADH:!EXP56:RC4+RSA:+HIGH:+MEDIUM:+LOW:
               +SSLv2:+EXP:+eNULL
SSLSessionCache         dbm:/var/log/apache/ssl_scache
SSLSessionCacheTimeout  300
SSLMutex file:/var/log/apache/ssl_mutex
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
SSLCertificateFile /usr/local/aircert/collector.crt
SSLCertificateKeyFile /usr/local/aircert/collector.key
SSLCACertificateFile /usr/local/aircert/ca.crt
SSLVerifyClient require
SSLVerifyDepth 1
```

図 8.4: mod_ssl 関連設定

```
<?xml version="1.0" encoding="utf-8"?>
<config>
<rule if expr="$SOURCE_NODE_ADDRESS_IPV4_ADDR = 1" />
<rule set var="SEQ" value="1" />
<rule elif expr="$SOURCE_NODE_ADDRESS_IPV4_ADDR = 2" />
<rule set var="SEQ" value="2" />
中略
<rule elif expr="$SOURCE_NODE_ADDRESS_IPV4_ADDR = 2000" />
<rule set var="SEQ" value="1000" />
<rule endif />
</config>
```

なお、$SOURCE_NODE_ADDRESS_IPV4_ADDR の値は 1 から 2000 のいずれにもマッチしないため、ルールファイルの終端に達した時点で処理が終了する。

8.2.2 評価手順

セキュリティイベントを記述した SNML 形式のファイルを 10000 件用意し、単一のコネクションで各ファイルをサーバに送信し、イベントの処理に要する時間を計測した。

以下に用いたセキュリティイベントファイルの内容を示す:

```
# Sets file extension on which to invoke mod_air
AddHandler air .air

# Should mod_air process events: (on | off)
AirEngine on

# Maximum size (in bytes) for each event
AirMaxEventSize 100000

# Shared-State file
AirSharedStateFile /tmp/collector/mod_air.state

# Logging Configuration
#AirTraceFile : full path to the trace log file
AirBufferLogging off
AirTraceFile /tmp/collector/mod_air_log
AirErrorFile /tmp/collector/mod_air_error
AirLogLevel 7

# Client Throttling Configuration
#AirThrottle: should client throttling be enabled (on | off)
#AirThrottleInterval : throttle interval over which to track clients
#(in seconds)
#AirThrottleThreshold: maximum allowable connections per Interval
AirThrottle off
AirThrottleInterval 1
AirThrottleThreshold 150

# Database configuration
#AirEventDBUri : URI of the event database
#AirCADBUri : URI of the CA database
AirEventDBUri postgresql://localhost/air
AirCADBUri   postgresql://localhost/air_ca

# Rulefile configuration
# IronConfigFile : Path to rulefile
IronConfigFile /usr/local/aircert/iron.conf
```

図 8.5: mod_air 関連設定

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SNML-Message>

<SNML-Message version="0.3">
  <sensor encoding="hex" detail="full">
    <file>rex</file>
    <hostname>localhost</hostname>
  </sensor>
  <event>
    <signature id="2003" revision="2">
      MS-SQL Worm propagation attempt
    </signature>
    <reference system="bugtraq">5311</reference>
    <timestamp>2003-12-08 19:38:48+0900</timestamp>
    <packet>
      <iphdr saddr="10.97.53.24" daddr="203.178.143.183"
       ver="4" hlen="20" tos="0" len="404" id="33166" flags="0"
       ttl="109" proto="17">
        <udphdr sport="3034" dport="1434" len="376">
        </udphdr>
      </iphdr>
    </packet>
  </event>
</SNML-Message>
```

上記の内容の試験を、ルールファイルを変更して 5 回ずつ行った。

8.2.3 評価結果

性能劣化

本実装と、mod_air の性能を比較し、この実装のモデルの正当性を評価した。

処理内容を含まないルールを設定した本実装と mod_air にそれぞれ 1 万件のデータを処理させ、その処理時間の違いを測定した。図 8.6に測定結果を示す。

本実装と mod_air の全体値に対して平均値の差の検定を行った。t 検定の結果を表 8.4に示す。

従って、本実装において有意な性能低下が発生しているとはいえない。

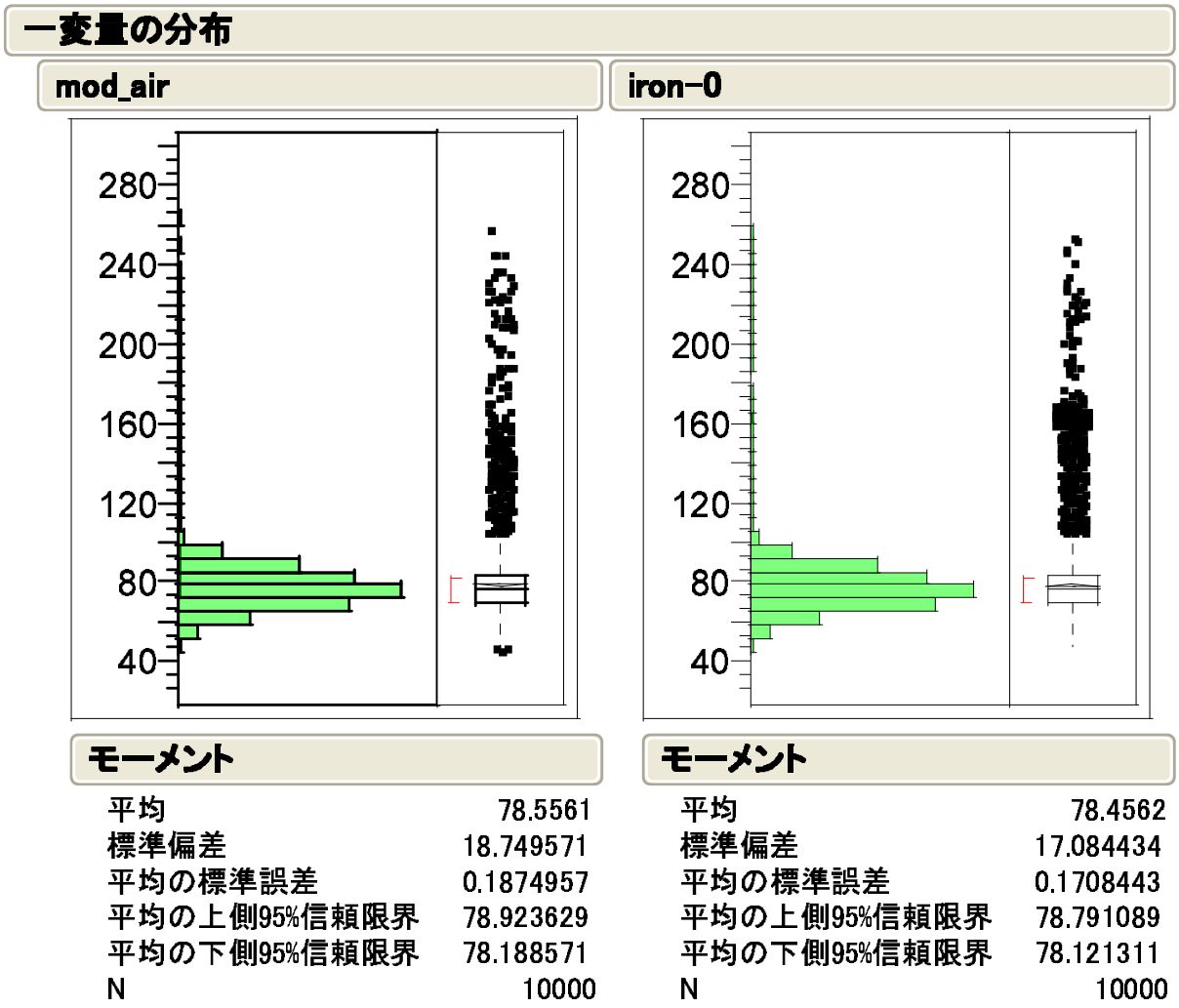


図 8.6: ルール数に伴う処理時間の変化

表 8.4: iron_parse 関数の処理時間 (N=10000, 単位:ms)

検定統計量	-0.585
p 値 (Prob¿—t—)	0.559
p 値 (Prob¿t)	0.721
p 値 (Prob¡t)	0.279

iron_parse の処理速度

および表 8.5および図 8.7に各ルールファイルの条件数を変化させた際の iron_parse 関数の処理時間のデータを示す。

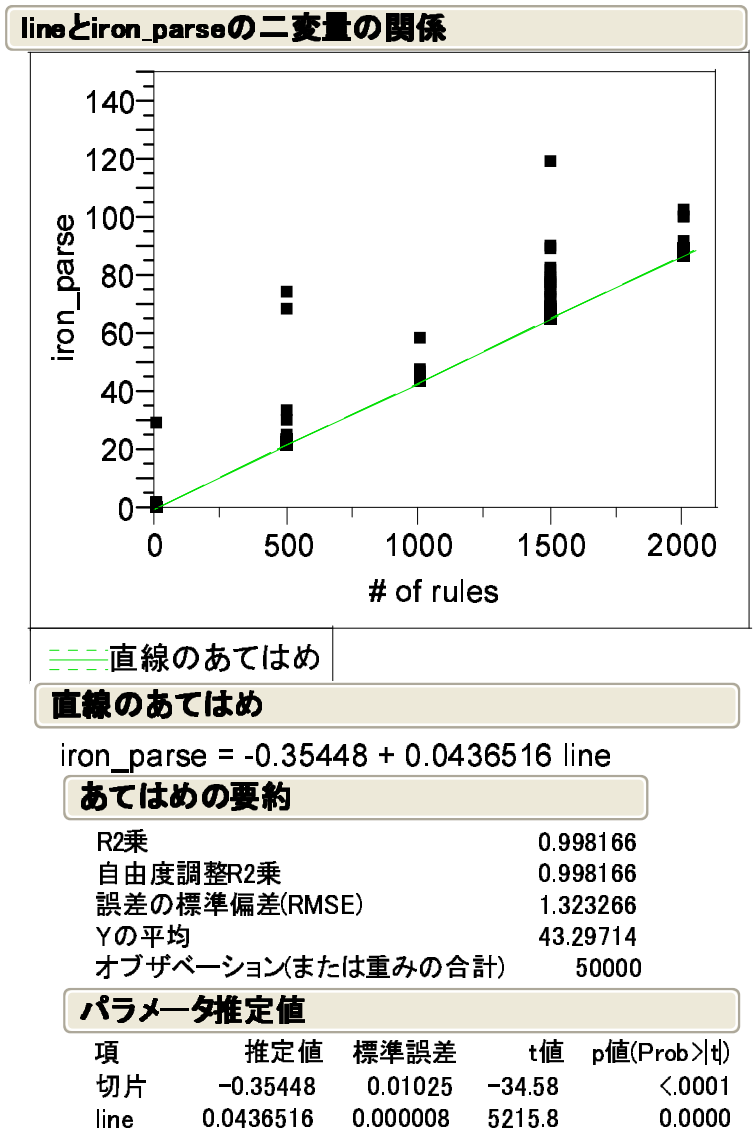


図 8.7: ルール数に伴う処理時間の変化

この結果をルール一件あたりの処理時間を x、iron_parse 関数全体の処理時間を y として直線にあてはめると、

$y = -0.35448 + 0.0436516x$

となる。このあてはめの R^2 乗は 0.998166 となり、当てはまっている。従ってルール 100 件あたり 4.4ms の処理時間を要することがわかった。

ルール数	平均	標準偏差	標準誤差	上側 95%信頼限界	下側 95%信頼限界
0	0.003	0.29	0.003	0.009	-0.002
500	21.069	0.761	0.008	21.084	21.054
1000	43.150	2.428	0.024	43.198	43.103
1500	65.193	1.293	0.013	65.218	65.167
2000	87.071	0.443	0.004	87.079	87.062

表 8.5: iron_parse 関数の処理時間 (N=10000, 単位:ms)

8.2.4 まとめ

本システムにおいては、処理内容を含まないルールファイルを設定した場合において、処理時間全体にデータベース関連の処理が占める割合は 49.52%であった。

また、ルールファイルの処理に要する時間は 100 件あたり 4.4ms であった。2000 件のルールを設定した場合、処理時間全体に占める割合は 52.09%であった。

従って、本システムの処理時間の半分はデータベースに依存しているといえる。そのため、処理時間を短縮するには、データベースの応答性能の向上が、今後の重要な課題である。

第9章　結論

9.1　まとめ

本研究により、さまざまなデバイスからセキュリティイベントデータとネットワーク環境情報を収集し、ルールに基づいたセキュリティイベントデータを処理できた。ルールに基づいたセキュリティイベントデータの処理は、Apache のモジュールの mod_air の機能を拡張することにより実現した。

本研究により、セキュリティイベントデータおよびネットワーク環境情報を統合的に扱うことが可能になった。また、ルールに基づき、セキュリティイベントデータを処理し、自動的なインシデントの対応や管理者への通知が可能になった。

これにより、管理者の負担軽減と、即時性のあるインシデント対応が実現した。

9.2　今後の課題

本システムの今後の課題としては、データベースの応答性能の向上がある。今回の評価では、データベースに格納されたエントリ数の増加に伴い、応答性能の低下が観測された。この応答性能の低下現象は PostgreSQL 特有の現象であるか、また他のデータベースを用いた場合にはどの程度改善されるのかについての調査する必要である。

また、本システムはルールに基づいて動作するため、ルールに定義されたセキュリティイベントのみの対応となる。従って、未知のセキュリティイベントに対して自動的に対処することはできない。今後は、データベースに蓄積されたセキュリティイベントを対象にデータマイニングを行い、未知のイベントに対するルールを自動的に生成できるようする必要がある。

本システムのルールにおいては、一般的な制御構文などを利用できる。しかし、ルールはセキュリティイベントごとにルールを適用しているため、複数のセキュリティイベント間の相関関係を記述できないといった制約が存在するため、複数のセキュリティイベントの解釈に適したルールの表現方法について検討を加える必要がある。

また、本システムは単一サイト内で運用されることを前提としているため、セキュリティイベント自体が偽造されることは想定していないが、他サイトからデータを収集する運用を考慮すると、データの偽造に対抗する方法を検討する必要がある。

本システムは各イベントに基づき動作を行うため、現在の設計では複数のセキュリティイベントの集約を行うことは難しい。そのため、複数のセキュリティイベントを考慮した処理方法を検討する必要がある。

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、同学部専任講師南政樹氏に感謝致します。

絶えず御指導と御助言を頂きました慶應義塾大学院政策・メディア研究科講師土本康生氏、同大学環境情報学部専任講師重近範行博士、同大学院政策・メディア研究科博士課程小原泰弘氏に深く感謝致します。

本論文の作成にあたり，御協力して下さった慶應義塾大学環境情報学部水谷正慶氏、金井瑛氏、並びに慶應義塾大学徳田・村井・楠本・中村南合同研究室の皆様、特に SING 研究グループの皆様に感謝の念を表します。

参考文献

[1] Code-red: a case study on the spread and victims of an internet worm. In *Proceedings of the Internet Measurement Workshop (IMW)*, 2002.

[2] Impress INTERNET Watch. ワーム「Code Red」被害拡大〜東京めたりっく・インターリンクなど被害続出〜. August 2001.

[3] R. McCool. The Common Gateway Interface. *http://hoohoo.ncsa.uiuc.edu/cgi/*, 1994.

[4] Hoglund, Greg. rootkit.com. *http://www.rootkit.com/*.

[5] CERT Coordination Center. CERT Advisory CA-1999-02 Trojan Horses. *http://www.cert.org/advisories/CA-1999-02.html*.

[6] Oystein Homelien. Smurf Amplifier Registry (SAR). *http://www.powertech.no/smurf/*.

[7] N. Brownlee, E. Guttman. RFC2350 Expectations for Computer Security Incident Response. *http://www.ietf.org/rfc/rfc2350.txt*, pages 1–38, June 1998.

[8] Moira J. West-Brown, Don Stikvoort, Klaus-Peter Kossakowski, Georgia Killcrece, Robin Ruefle, Mark Zajicek. Handbook for Computer Security Incident Response Teams (CSIRTs). *http://www.sei.cmu.edu/publications/documents/03.reports/03hb002.html*, pages 1–193, April 2003.

[9] Brezinski, D. and T, Killalea. Guidelines for Evidence Collection and Archiving. *http://www.ietf.org/rfc/rfc3227.txt*, February 2002.

[10] The Honeynet Project. Know Your Enemy: Passive Fingerprinting. *http://www.honeynet.org/papers/finger/*, March 2002.

[11] Eugene H. Spafford. The internet worm program: An analysis. Technical Report Purdue Technical Report CSD-TR-823, West Lafayette, IN 47907-2004, 1988.

[12] Donn Seeley. A Tour of the Worm. *Proceedings of the Winter 1989 USENIX Conference*, January 1989.

[13] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. The Spread of the Sapphire/Slammer Worm. Technical report, January 2003.

[14] Microsoft Corporation. Microsoft SQL Server: Home. *http://www.microsoft.com/sql/*, 2000.

[15] Microsoft Corporation. Microsoft SQL Server: MSDE 2000 Home. *http://www.microsoft.com/sql/msde/default.asp*, 2000.

[16] J. Postel. RFC768 User Datagram Protocol. *http://www.ietf.org/rfc/rfc0768.txt*, pages 1–3, August 1980.

[17] Honeynet Project. Know Your Enemy: Statistics Analyzing the past ... predicting the future. *http://project.honeynet.org/papers/stats/*, July 2001.

[18] International Standard Organization. Information Processing. OSI Reference Model - Part3: Naming and Addressing. *International Standard 7498-3, ISO. (6).*

[19] US Congress. 18 U.S.C. 2511. Interception and Disclosure of Wire, Oral, or Electronic Communications Prohibited. Technical report, October 1986.

[20] M. Wood and M. Erlinger. Intrusion Detection Message Exchange Requirements draft-ietf-idwg-requirements-10. pages 1–25, October 2002.

[21] D. Curry and H. Debar. Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition. pages 1–119, January 2003.

[22] B. Feinstein and G. Matthews and J. White. The Intrusion Detection Exchange Protocol (IDXP) draft-ietf-idwg-beep-idxp-07. pages 1–40, October 2002.

[23] D. New. RFC3620 The TUNNEL Profile. pages 1–18, October 2003.

[24] Rose, M. RFC3080 The Blocks Extensible Exchange Protocol Core. pages 1–57, March 2001.

[25] Yuri Demchenko and Hiroyuki Ohno and Glenn M Keeni. Requirements for Format for INcident information Exchange (FINE) draft-ietf-inch-requirements-02.txt. pages 1–12, October 2003.

[26] J. Meijer and R. Danyliw and Y. Demchenko. The Incident Data Exchange Format Data Model and XML Implementation draft-ietf-inch-iodef-02.txt. pages 1–85, September 2003.

[27] Martin Roesch. Snort: Lightweight intrusion detection for networks. In *13th Systems Administration Conference (LISA '99)*, November 1999.

[28] Check Point Software Technologies Ltd. Build Your Security Infrastructure With Best-of-Breed Products From OPSEC. 2003.

[29] Check Point Soft-
ware Technologies Ltd. OPSEC SDK Support: FireWall-1 SAM API Specification. *http://www.checkpoint.com/support/technical/sdksupport/docs/apis/specs/samptoc.html*, 1997.

[30] S. E. Hansen and E. T. Atkins. Automated system monitoring and notification with swatch. *Proceedings of the Seventh Systems Administration Conference (LISA VII) (USENIX Association: Berkeley, CA)*, page 145, 1993.

[31] C. Lonvick. The BSD Syslog Protocol. *http://www.ietf.org/rfc/rfc3164.txt*, August 2001.

[32] Mark Burgess. Cfengine: a site configuration engine. *USENIX Computing Systems, Vol 8, No. 3 1995.*, 1995.

[33] Marcus J. Ranum, Kent Landfield, Mike Stolarchuk, Mark Sienkiewicz, Andrew Lambeth, and Eric Wall. Implementing A generalized tool for network monitoring. In *Proceedings of the Eleventh Systems Administration Conference (LISA '97)*, San Diego, CA, 1997.

[34] ArcSight Inc. ArcSight Inc. - Enterprise Security Management Software. *http://www.arcsight.com/*.

[35] Pickel, Jed and Danyliw, Roman. Enabling Automated Detection of Security Events that Affect Multiple Administrative Domains. *MS Thesis, Information Networking Institute, Carnegie Mellon University*, December 2000.

[36] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee. RFC2616 Hypertext Transfer Protocol – HTTP/1.1. *http://www.ietf.org/rfc/rfc2616.txt*, June 1999.

[37] E. Rescorla. HTTP Over TLS. *http://www.ietf.org/rfc/rfc2818.txt*, May 2000.

[38] Dierks, T. and C. Allen. The TLS Protocol. *http://www.ietf.org/rfc/rfc2246.txt*, January 1999.

[39] The PHP Group. PHP: Hypertext Preprocessor. *http://www.php.net/*, 2001.

[40] John Lim. ADOdb Database Library for PHP: Create Portable DB Apps. *http://php.weblogs.com/ADODB*, 2000.

[41] P. Ferguson and D. Senie. RFC2827 Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. *http://www.ietf.org/rfc/rfc2827.txt*, May 2000.

[42] Apache Software Foundation. Apache 2.0 vulnerability affects non-Unix platforms. *http://httpd.apache.org/info/security_bulletin_20020809a.txt*, August 2002.

[43] Victor A. Abell. Lsof. *ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/*, Dec 1994.

[44] Microsoft Corporation. Internet Information Services. *http://www.microsoft.com/iis/*.

[45] Microsoft Corporation. Buffer Overrun in Microsoft FrontPage Server Extensions Could Allow Code Execution (813360). *http://www.microsoft.com/technet/security/bulletin/MS03-051.asp*, November 2003.

[46] Microsoft Corporation. Microsoft Baseline Security Analyzer. *http://www.microsoft.com/japan/technet/security/tools/tools/mbsahome.asp*, June 2003.

[47] インターネット セキュリティ システムズ株式会社. 注意：Nachi に関する追加情報. *http://www.isskk.co.jp/support/techinfo/general/Nachi.html*, September 2003.

[48] SecurityFocus, Inc. SecurityFocus HOME Vulns Archive. *http://www.securityfocus.com/bid*, 1999.

[49] MITRE Corporation. Common Vulnerabilities and Exposures. *http://www.cve.mitre.org/*.

[50] Gartner, Inc. Gartner Information Security Hype Cycle Declares Intrusion Detection Systems a Market Failure. *http://www3.gartner.com/5_about/press_releases/pr11june2003c.jsp*, June 2003.

[51] Top Layer Networks. IDS Balancer.

[52] F5 Networks, Inc. BIG-IP and Firewall Load Balancing. *http://www.f5.com/solutions/applications/Firewalls/fwlbflash.html*.

[53] W3C. Extensible Markup Language(XML). *http://www.w3.org/XML/*.

[54] The Apache Software Foundation. The Apache HTTP Server Project. *http://httpd.apache.org/*.

[55] The Apache Software Foundation. Apache module mod_include. *http://httpd.apache.org/docs/mod/mod_include.html*.

[56] NCSA HTTPd Development Team. NCSA HTTPd Tutorial: Server Side Includes (SSI). *http://hoohoo.ncsa.uiuc.edu/docs/tutorials/includes.html*, 1995.

[57] Mark Cox and Geoff Thorpe. High Scalability for SSL and Apache. *ApacheCon 2000, London*, September 2000.