

博士論文

標的型攻撃における  
感染経路検知方式の研究

Study on Infection Route Detection  
Method against Targeted Attack

2019年3月

佐藤信

Study on Infection Route Detection  
Method against Targeted Attack

DISSERTATION

Makoto Sato

Tokyo Denki University

March, 2019

# 目次

目次.....	3
第1章 緒言.....	1
第2章 研究を取り巻く状況.....	6
2.1 サイバー攻撃.....	6
2.2 関連技術.....	11
2.2.1 揮発性情報取得ツール：Onmitsu.....	11
2.2.2 サイバー攻撃観測記述形式：CybOX.....	14
2.2.3 Resource Description Network.....	17
2.3 関連研究.....	19
2.3.1 サイバー攻撃推定手法.....	19
2.3.2 サイバー攻撃における情報共有手法.....	19
2.3.3 ネットワーク経路探索手法.....	22
第3章 マルウェアにより不正挙動をとるPCの検知方式の提案と評価...	25
3.1 ネットワークの表示方法の検討.....	25
3.1.1 センサネットワークと利用情報の適切な選択方法の検討.....	25
3.1.2 プラットフォームの開発.....	30
3.1.3 シミュレーション実験.....	38
3.2 検知方式の検討.....	45

3.3	プロセスログの CybOX 変換手法の検討.....	47
3.4	プロセスログを用いたマルウェア動的解析 .....	54
3.5	プロセスログを用いたマルウェア攻撃検知手法 .....	58
3.5.1	提案手法 .....	58
3.5.2	検証実験 .....	59
3.5.3	考察 .....	60
3.6	プロセスパターンを用いたマルウェア検知手法 .....	61
3.6.1	マルウェアプロセスパターンの作成.....	62
3.6.2	提案手法 .....	63
3.7	実験.....	64
3.7.1	実験環境 .....	64
3.7.2	事前準備：比較用マルウェアプロセスパターンの作成 .....	65
3.7.3	実験手順 .....	66
3.7.4	実験結果 .....	68
3.7.5	考察 .....	70
第 4 章	侵入検知ツールの開発.....	71
4.1	感染経路検知ツールの開発 .....	72
4.1.1	標的型攻撃における内部侵入・調査段階の挙動.....	72
4.1.2	内部通信時の挙動解析 .....	72

4.1.3	感染経路特定手法 .....	76
4.2	実験 .....	77
4.2.1	実験環境 .....	77
4.2.2	実験手順 .....	79
4.2.3	実験結果 .....	80
4.3	考察 .....	81
4.3.1	感染経路検知手法の検証 .....	81
4.3.2	開発プログラムの検証 .....	83
4.4	開発プログラムの課題 .....	84
第5章	今後の課題 .....	86
第6章	結論 .....	89
	参考文献 .....	91
	謝辞 .....	97

## 第1章 緒言

近年、サイバー攻撃が多様化している。その中でも、標的型攻撃が大きな脅威となっている。標的型攻撃とは、組織がもつ知的財産や個人情報等に関わる機密性の高い情報の取得を目的としたサイバー攻撃の一種である[1]。標的型攻撃はその目的から、一度でも攻撃が成功すると組織に甚大な被害を及ぼす。近年の大きな被害事例として日本航空、日本年金機構、JTB が挙げられる[2]。さらに今後クラウドサービスや IoT が全世界的に広がることが予想されサービスを受ける利用者の個人情報や、サービスを提供するための企業の知的財産の価値は飛躍的に上昇しており、その保護はさらに重要性を増している。そのため、標的型攻撃への対策は今後さらに重要になると考えられる。標的型攻撃の一般的な手法は、偽装メールと不正プログラム（マルウェア）を組み合わせて行う方法である。攻撃者は偽装メールを送り、それを見た特定のユーザに対して特定の行為を誘導することで不正な情報の取得を行う。さらに、標的型攻撃の攻撃対象は組織内の機密情報を取得するため偽装メールを受けた端末のみにとどまらずネットワーク全体の端末に及ぶ。

現在、標的型攻撃を受けた際の調査は下記のフェーズに分けられている[3]。

1. 攻撃かどうか調査（メール等が標的型攻撃のものかどうか）
2. 感染端末の調査（端末が感染しているかどうか）
3. 被害範囲の把握(通信先, 侵入元, 被害範囲, 情報漏えいの把握)
4. 対策の有効性の把握（端末に施した対策が有効かどうか）

しかし、標的型攻撃が発見されてから対策の実施に至るまで多大な時間を要しており、対策の実施や外部への公表等の対応が遅れている。その原因として下記の点が挙げられる。

調査フェーズ 2 :

- ① 攻撃者による痕跡の削除（攻撃を発見するためのログ不足）
- ② 端末によるログ記録方法, 粒度の不統一（各ログの時間と粒度へのギャップ）

調査フェーズ 3 :

- ③ 総合的なログ解析ができる専門家の不足（複数ログ上の痕跡発見方法の不足）

したがって、標的型攻撃に適切に対処するためには、端末内で起きた事象の解析だけでなく、これらの情報を組み合わせて解析し総合的に判断する必要がある。また、事象情報を組み合わせて解析する際には、事象情報の表現形式が統一されていることが望ましい。さらに、利用者による解釈を容易にするために解析結果はネットワーク全体の事象として表現する必要がある。つまり、上記の問題を解決するためには下記の対応が必要となる。

- ① 痕跡の削除が困難なログの記録
- ② 事象情報の表現形式の統一
- ③ 複数の事象を組み合わせた自動的な標的型攻撃の検知手法

そこで、これらの要求を満たすために本研究では標的型攻撃のための標準を利用した感染経路検知方式を検討する(図 1.1)。

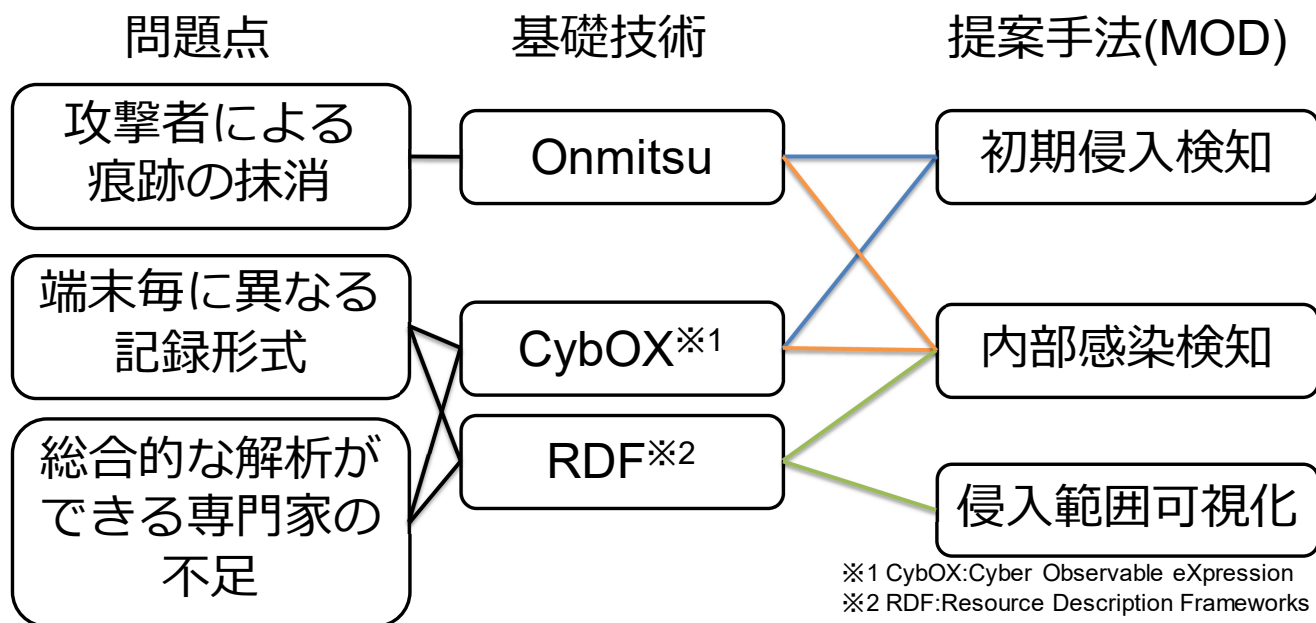


図 1.1 問題点と提案手法の関係

本研究室ではこれまで、標的型攻撃の原因を特定するための研究としてプロセス情報とそのプロセスが発した通信に関するログを記録する手法を検討しており、現在、「Onmitsu」という常駐型のプログラムが開発されている。これにより、時間と共に取得が困難となるが攻撃事象の事実関係の把握や法的証拠として有用である揮発性情報の取得が容易にできる。

また、情報を記述して解析するための統一した表現として、本研究では「CybOX」を利用する。CybOX (Cyber Observable eXpression)とは、サイバー攻撃の観測事象を記述するための標準の1つである。このCybOXを使うことで事象情報は統一された表現形式で記述でき情報交換が容易になる。CybOXは現在Microsoftが開発しているセキュリティや脅威情報のナレッジを交換するためのプラットフォームであるMicrosoft Interflowでの活用を検討している。このように、サイバー攻撃の振る舞いがCybOXで記述されることが今後一般的になると予想される。しかし、その表現可用性の評価はほとんど検討されていないため、その評価も同時にする必要はある。



さらに、本研究では標準をベースとしたプロセスパターンを用いたマルウェア検知手法を検討した。なお、プロセスパターンの作成には前述した「Onmitsu」を使用する。本手法では、標準を利用しているため情報共有が容易であり、また共有される多大な情報を容易に使えるためマルウェアの個別検知精度が向上することが期待される。さらに、解析結果をオントロジーで表現する。オントロジーを使うことで提案手法は攻撃行動を予測できるようになり、その結果これまでのように各端末内の警告だけではなく、ネットワーク全体での感染経路を検知することができる。これにより、利用者は時々刻々と変わる事象に合わせた総合的な対策ができる。上記の通り、本研究で提案するシステムを図 1.2 に示す。詳細は 2 章以降で説明するが、提案システムは主に下記の機能を持つ。

- **Onmitsu to CybOX** : Onmitsu で取得した揮発性情報を CybOX 形式に変換(3 章)
- **Onmitsu to RDF** : Onmitsu で取得した揮発性情報を RDF 形式に変換(3 章)
- **端末挙動検知** : CybOX 形式のログから端末内に生じた攻撃活動を検知(3.5 節)
- **ネットワーク検知** : 端末間の活動から、攻撃感染経路を検知(4 章)
- **ネットワーク状態可視化** : 検知した攻撃活動を可視化(4 章)

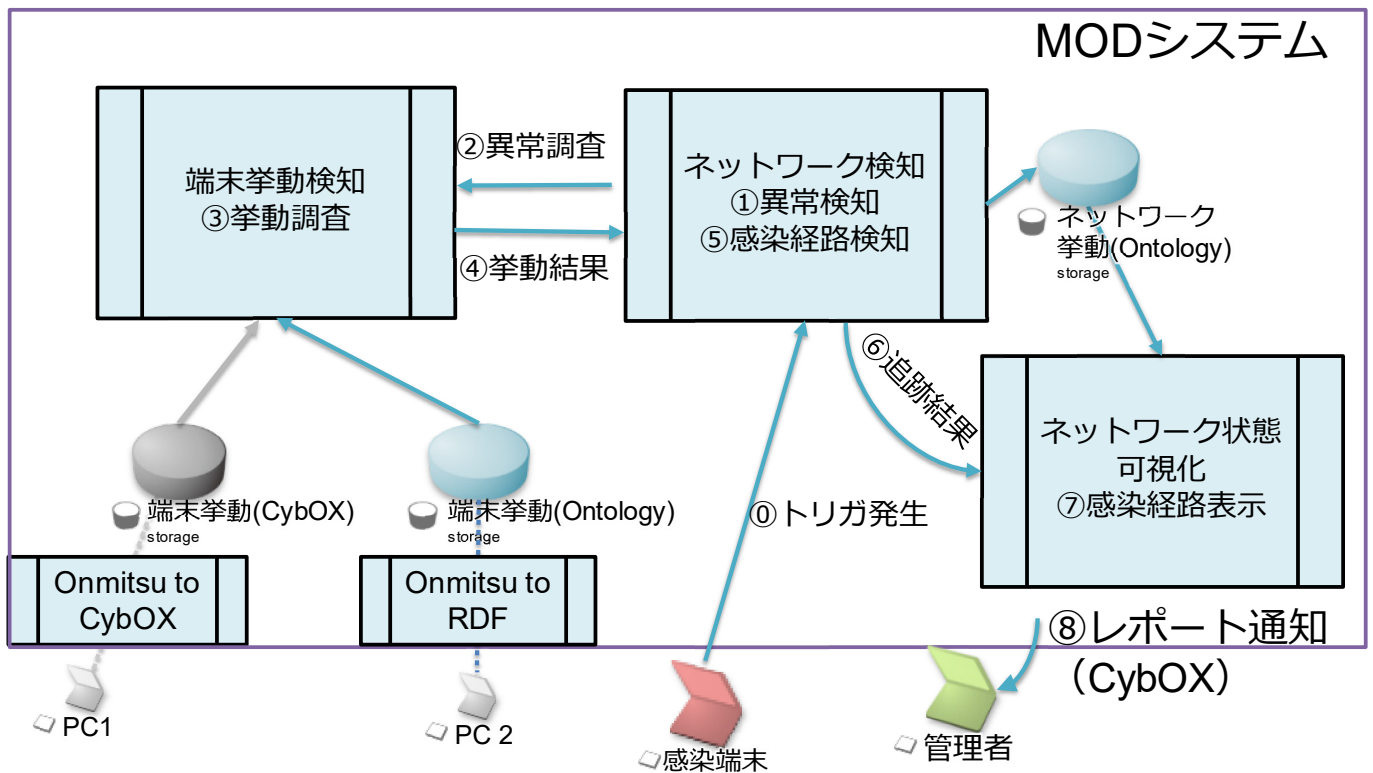


図 1.2 提案システムの概要

本論文の構成は以下の通りである。

第 2 章でサイバー攻撃の概要と本論文の基礎となる技術の説明，そして関連研究について述べる．第 3 章でネットワーク上の端末のログの **CybOX** によるログ記録形式の統一方法と，感染端末の検知方法（図 1.2 の①～④）について述べる．第 4 章でネットワーク上の感染源検知方法（図 1.2 の⑤～⑦）について述べる．第 5 章で今後の課題と展望について述べ，第 6 章で本研究の結果についてまとめる．

## 第2章 研究を取り巻く状況

### 2.1 サイバー攻撃

サイバー攻撃が大きな問題になっている。その中でも、標的型攻撃が大きな脅威となっている。IPA のよる報告（表 2.1）では、昨年と同様標的型攻撃による情報流出が 1 位となり、脅威の注目度は依然高いままである[4]。

さらに、JNSA による、インシデントの情報を集計した結果は表 2.2 の通り。なお、各項目の値については、収集した情報を元に、漏えいした組織の業種、漏えい人数、漏えい原因、漏えい経路などの分類・評価し、独自の算定式（JO モデル）を用いて、想定損害賠償額を算出している[5]。

表 2.1 情報セキュリティ 10 大脅威 2018

「個人」向け脅威	順位	「組織」向け脅威
インターネットバンキングやクレジットカード情報等の不正利用	1	標的型攻撃による被害
ランサムウェアによる被害	2	ランサムウェアによる被害
ネット上の誹謗・中傷	3	ビジネスメール詐欺による被害
スマートフォンやスマートフォンアプリを狙った攻撃	4	脆弱性対策情報の公開に伴う悪用増加
ウェブサービスへの不正ログイン	5	脅威に対応するためのセキュリティ人材の不足
ウェブサービスからの個人情報の窃取	6	ウェブサービスからの個人情報の窃取
情報モラル欠如に伴う犯罪の低年齢化	7	IoT 機器の脆弱性の顕在化
ワンクリック請求等の不当請求	8	内部不正による情報漏えい
IoT 機器の不適切な管理	9	サービス妨害攻撃によるサービスの停止
偽警告によるインターネット詐欺	10	犯罪のビジネス化 (アンダーグラウンドサービス)

表 2.2 2017 年 個人情報漏えいインシデント 概要データ

漏えい人数	519万8,142人
インシデント件数	386件
想定損害賠償総額	1,914億2,742万円
一件あたりの漏えい人数	1万4,894人
一件あたり平均想定損害賠償額	5億4,850万円
一人あたり平均想定損害賠償額	2万3,601円

標的型攻撃の事例として、2016年には株式会社 i.JTB（JTB グループ）や国立大学法人富山大学，研究推進機構水素同位体科学研究センター，および一般財団法人日本経済団体連合会が標的型攻撃の被害を受けている。このように、2015年6月の日本年金機構での個人情報の流出事件以降も、標的型攻撃による被害が継続している。これらの攻撃事例の感染原因は、複数回に分けて添付ファイル付きのメールや外部 URL が記されたメールが送付され、これらのメールを受信者が開封、実行したことによる感染が原因と考えられる。標的型攻撃はますます巧妙化しており、業務によってはメールの開封を回避することが困難であるため、不審なメールを開封してしまったことに気づいたらすぐに報告すること、不審な外部との通信を監視することなど、被害を最小化する組織的な取り組みが求められている。そのためには、組織外との連絡窓口、組織内の適切な連絡体制の整備等、セキュリティインシデントを組織として受け止める体制を構築しておくことが重要である。

組織を標的としたこのような新たなサイバー攻撃への対策については、攻撃手

法の解析が困難であることや攻撃を受けた後の対応が確立されていないこと，組織内のネットワーク管理者の対応能力が不足していることが指摘されている等，十分とは言えない状況である．このような状況をふまえ，総務省では平成 25 年度より，官公庁・重要インフラ事業者等のネットワーク管理者のサイバー攻撃への対応能力の向上を目的として，職員が数千人規模の組織内ネットワークを模擬した大規模環境を用いた実践的なサイバー防御演習(CYDER: CYber Defense Exercise with Recurrence)を実施している(図 2.1)．

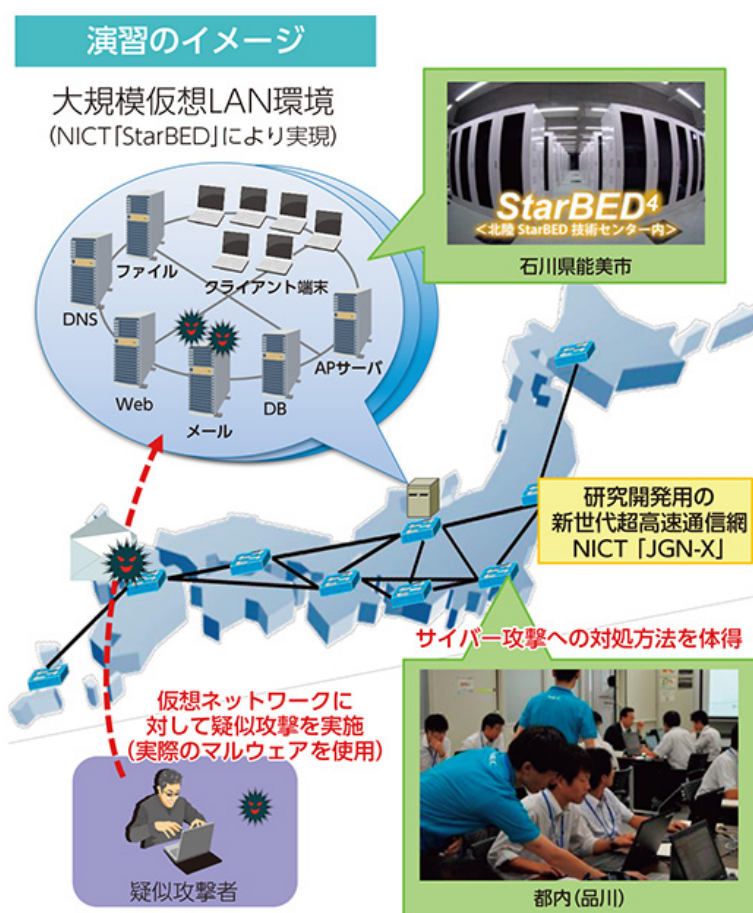


図 2.1 実践的サイバー防御演習 (CYDER : CYber Defense Exercise with Recurrence)

IPA の報告書によると，標的型メール攻撃の典型的な手法は，偽装メールとマル

ウェアを組み合わせる方法である。この攻撃の実施は、図 2.2 のように 6 段階で定義されている [2]。

- (1) 計画立案段階：攻撃対象を決定し情報を収集
- (2) 攻撃準備段階：偽装メールや C&C サーバを用意
- (3) 初期侵入段階：偽装メールによるマルウェア感染
- (4) 基盤構築段階：感染端末の情報を窃取し環境を調査
- (5) 内部侵入・調査段階：端末間での侵害を拡大
- (6) 目的遂行段階：窃取した情報を外部へ送信



図 2.2 標的型攻撃のシナリオと対策

攻撃者は段階(3) で特定のユーザに対して特定の行為をとるよう誘導すること

で侵入し、有用な情報を窃取するために段階(6)を目指す。

以上のように、標的型攻撃に対処するためには、端末内で起きた事象の解析だけでなく、各端末の事象情報を組み合わせて解析し判断する必要がある。これを実現するためには、次の3項目が必要となる。

(1) 標的型攻撃の原因を特定するため、各情報処理機器においてプロセス情報や通信情報などの事象をログとして記録する手段

(2) 解析システムへ送信・解析するために、それらの情報処理機器のログを統一の形式で記述する手段

(3) 集約されたログと情報処理機器間の関係から攻撃の挙動を推定する手段

本研究ではこのように全体的かつ動的に攻撃が診断できる方式を検討する。これまで、専門家が必要な情報を集め解析することで攻撃の振る舞いを診断していたが、これらを実現することで個々の情報処理機器内での検知や、IDSでのパケット監視による不正アクセス検知などの従来の対策にあわせて、それらの関係性も含めたログをすべて用いることで全体的また動的に攻撃の挙動を診断できると期待される。本研究ではまず標的型攻撃の初期フェーズである侵入行動に焦点をあてることとした。

この3項目のうち(1)については本研究室で開発したプロセスログ記録ツール「Onmitsu」を用いることとした[6]。このツールはそれぞれが揮発性情報の一種であるプロセス挙動と、そのプロセスと関係する通信試行を記録する常駐型のプログラムである。カーネルドライバという形で導入しているため、マルウェアのプロセス隠蔽処理など、より多くの情報が記録できると期待される。このログを解析することでマルウェアに起因する通信とマルウェアのプロセスを結びつけることができる。

(2)についてはログを統一的形式で表現するためにCybOX™を用いることとした。その理由は、送信されるログの形式は様々であり、また、複数のログを総合的に解析する際には、表現形式が統一されていることが望ましいからである。CybOXを用いることで機器間の情報交換やログの総合的な解析が容易になる。しかし、収集したログを動的にCybOXの形式へと変換できるかという問題がある。

(3)については、集約したログを総合的に解析する必要がある。あわせて、検知するための診断基準も検討する必要がある。そこで、本研究ではマルウェアのプロセスパターンも CybOX で表現することで、集約したログとマッチングさせ攻撃の挙動を検知する手法を検討する。さらに、この手法をもとに標的型攻撃におけるプロセスレベルでの感染経路追跡手法を検討した。

本手法を実現するための1要素として、攻撃挙動をプロセスレベルで把握するために Onmitsu を利用する。ただし、Onmitsu は導入した端末内のプロセスとそのプロセスが発した通信のみを記録するプログラムであるため、各端末が属するネットワークとプロセスログを関連付ける機能はない。そこで、端末間のネットワーク構造を RDF で表記し、Onmitsu で記録されたプロセスログとネットワーク構造を組み合わせることで、ネットワーク全体で分析することによりプロセスレベルで感染経路を追跡する手法を検討する。

## 2.2 関連技術

### 2.2.1 揮発性情報取得ツール：Onmitsu

2.1 節で述べたように、標的型攻撃の原因を類推するための有用な情報源が必要となる。近年、その情報源として揮発性情報に着目しその取得方法が検討されている。揮発性情報とは、コンピュータのメインメモリ上のデータ等、電源が OFF になると保持されないものを指す。メモリ上に保存されている情報として、プロセス ID やプロセスが実施する通信試行などが挙げられる。

現在、原因を類推するために SIEM などの製品が使われている。しかし、これらの方法では揮発性情報であるプロセス情報やそれによる通信試行に関する情報は十分には記録されていない。また、揮発性情報を取得するためのツールを利用し、解析するにはネットワーク管理者が高度な知識を持つ必要がある。そのため、セキュリティ従事者は、断片的な情報を解析することでその原因を探る必要があった。

これまで、本研究室では標的型攻撃の原因を類推するための有用な情報源として揮発性情報に着目しその取得方法を検討してきた。攻撃の原因を検知する上で、プロセスの挙動と通信状況を記録することは重要である。これまで、それぞれのログをとるツールは存在するが、攻撃に使用されるマルウェアがプロセスを隠蔽する処理を行う可能性が高く、不正通信を行ったプロセスが判明してもそのプロセスの起動における経緯が判明せずマルウェアの特定が困難となる可能性があった。そこで、プロセス情報とそのプロセス情報と関連付けたパケットを記録する手法を検討し、Onmitsu というツールを開発した。Onmitsu は Windows の標準 API を利用しカーネルドライバという形で記録している。さらに、常駐して記録し続けるためマルウェアによるプロセスの隠蔽処理も回避できる可能性が高い。これまで、検証実験により記録したログからマルウェアのプロセ



スとマルウェアに起因した通信とが結びつけられることが確認されている。

次にログの内容について説明する。Onmitsu で記録する対象はプロセスにおける起動・終了・モジュール読み込み・通信試行の4つの挙動(ログタイプ)である。ツールはこれらの挙動とプロセスの情報を関連付けて記録する。記録形式は CSV であり、17 項目に分かれている。その内訳は下記の通りである。

年, 月, 日, 時, 分, 秒, ミリ秒, ログタイプ, PID, ParentPID, ファイルパス, コマンドライン, 接続元 IP アドレス, 接続元ポート番号, 接続先 IP アドレス, 接続先ポート番号, プロトコル番号

ログタイプと記録内容との対応関係を表 2.3 に示す。本研究ではこのツールで取得できるプロセスログを使って攻撃の挙動を把握する。

表 2.3 ログに記録される情報一覧

挙動	記録内容
プロセス起動 (PROCESS_LAUNCH)	起動時刻
	プロセスID
	要求を行った親プロセスID
	実行イメージファイルパス
	コマンドライン
プロセス終了 (PROCESS_QUIT)	終了時刻
	プロセスID
モジュール読み込み (PROCESS_MODLOAD)	読み込んだ時刻
	プロセスID
	モジュールイメージパス
通信試行 (NETWORKV4, NETWORKV6)	通信確立時刻
	プロセスID
	接続元IPアドレス
	接続元ポート番号
	接続先IPアドレス
	接続先ポート番号
トランスポート層プロトコルID	

## 2.2.2 サイバー攻撃観測記述形式：CybOX

CybOX(Cyber Observable eXpression)とは、サイバー攻撃で生じる「標的型攻撃メール受信」「不正通信」等の事象を記録し交換するために、米国政府の支援を受けた非営利団体である MITRE が仕様策定を進めたサイバー攻撃観測記述形式である[7]。サイバー攻撃の観測事象を表現するための仕様で、XML 文書で記述できる。共通攻撃パターン一覧である CAPEC の延長で検討が開始され、2012年11月に CybOX v1.0 がリリースされ、現在は v2.1 がリリースされている。現在もなお検討が続いている。

CybOX では観測事象 (Observable) が XML の根に相当する。観測事象はイベント(Event)とオブジェクト(Object)から構成されている。オブジェクトとはサイバー攻撃で観測された事象の主体である。例えば、ファイルやレジストリキー、プロセスなどである。イベントとはサイバー攻撃の観測事象中に発生した動作である。イベントでは個別の動作をアクションとして記述する。例えば、「メールを受信した」や、「ファイルを削除した」、「レジストリキーを作成した」などがアクションに該当する。このイベントの記述によって観測事象の意味が付加される。CybOX では CybOX Schema を作成しており、前述したオブジェクト名やアクションの形態などを示す語彙が豊富にある。CybOX での記述例を図 2.3 に示す。この図は IPA で作成されたものであり、メールに添付された zip ファイルに関する観測事象について記述している。具体的にはファイル名や拡張子、ファイルサイズ、MD5 ハッシュ、SHA1 ハッシュ、関連しているオブジェクトへのリファレンスが記述されている。このとき、Observable の区切り方によって観測事象の意味づけを変えられる。例えば、「zip ファイル」、「メール受信」を異なる Observable で示せば単なる事象の表現である。しかし、これらを組み合わせてアクションとして「メール受信」を、関連オブジェクトとして「zip ファイル」を記述すれば「添付ファイル付メールを受信した」という攻撃事象として表現することもできる。た

```

- <cybox:Observable id="observable_ipa_000009392_02">
  <cybox:Description>標的型攻撃メールの添付ファイル名は、「事務系連絡網.zip」という、メール本文に関係深い日本語が使われていました。なお、このZIPファイルはパスワード
  無しで解凍可能でした。</cybox:Description>
- <cybox:Object id="object_ipa_email_attachment_zip_000009392">
  - <cybox:Properties xsi:type="FileObj:FileObjectType">
    <FileObj:File_Name>事務系連絡網.zip</FileObj:File_Name>
    <FileObj:File_Extension>.zip</FileObj:File_Extension>
    <FileObj:Size_In_Bytes>326941</FileObj:Size_In_Bytes>
  - <FileObj:Hashes>
    - <cyboxCommon:Hash>
      <cyboxCommon:Type>MD5</cyboxCommon:Type>
      <cyboxCommon:Simple_Hash_Value condition="Equals">40d56443238046a0fbc7f8b30ee04f42</cyboxCommon:Simple_Hash_Value>
    </cyboxCommon:Hash>
    - <cyboxCommon:Hash>
      <cyboxCommon:Type>SHA1</cyboxCommon:Type>
      <cyboxCommon:Simple_Hash_Value
        condition="Equals">3e094f93b8075f1310ed3360d486897516b032ff</cyboxCommon:Simple_Hash_Value>
      </cyboxCommon:Hash>
    </FileObj:Hashes>
  </cybox:Properties>
- <cybox:Related_Objects>
  - <cybox:Related_Object idref="object_ipa_email_000009392">
    <cybox:Relationship xsi:type="cyboxVocabs:ObjectRelationshipVocab-1.0">Contained_Within</cybox:Relationship>
  </cybox:Related_Object>
  - <cybox:Related_Object idref="object_ipa_email_attachment_zip_exe_000009392">
    <cybox:Relationship xsi:type="cyboxVocabs:ObjectRelationshipVocab-1.0">Compressed</cybox:Relationship>
  </cybox:Related_Object>
</cybox:Related_Objects>
</cybox:Object>
</cybox:Observable>

```

図 2.3 CybOX の記述例

だし、複雑な意味づけは **Descriptions** 要素に記入しなければ明確に伝えることはできない。このように、**CybOX** はサイバー攻撃を統一的に表現可能であり、対応範囲は広く有用であると考えられる。

サイバー攻撃観測記述形式として **CybOX** は提案されており、**CybOX** を用いた観測事象の記述も前述したように進んでいるが、その表現可用性について検討している資料が少ない。ここで、表現可用性とはサイバー攻撃時の観測事象を適切に表現できるかの評価である。この適切さとは、文書を分析してセキュリティ専門家の知識を獲得できることを示す。前述したように、**IPA** はこれまでのサイバー攻撃における数々の事象を **CybOX** で記述し公開している。記述対象はこれまでに起きたサイバー攻撃の事案であり、専門家が分析した結果を **CybOX** で記述したものである。このことから、専門家が分析した観測事象は適切に表現可能だといえる。しかし、機械的に取得した観測事象については検討されていない。また、後述する **GRR** プロジェクトでは取得したログを **Protocol Buffers** という **Google** が開発し

たスキーマ言語で表現するために **CybOX** スキーマを用いている。しかし、表現可用性について言及した資料は公開されていない。したがって、収集したログは収集した意図と同じ意味をもつように **CybOX** で記述可能か検討する必要がある。あわせて、表記上の課題があれば **CybOX** を拡張することも検討する必要がある。

そこで、本研究では新たな揮発性情報の取得ツールでありプロセスログを **CybOX** の記述対象とすることでこの課題を検討していく。

標的型攻撃ではネットワーク内の様々な場所で活動する。そのため、攻撃を検知するためには各端末を監視し、ネットワーク内に存在する情報の相互関係を把握したほうがよい。そして、相互関係を適切に把握するためには統一した形式で各端末の情報を表現する必要がある。さらに、解析結果はネットワーク全体の事象として表現する必要がある。そこで、本研究ではネットワーク内のマルウェアの挙動を総合的に把握する手法を検討する。具体的には、プロセスログを用いて各端末内の挙動を取得して、それらを **CybOX** で記述する。その後、**CybOX** を利用してマルウェアのプロセスパターンを作成し、これを用いた検知システムを提案する。なお、本研究ではネットワーク内の様々な端末に影響を与えるマルウェアとして **RAT(Remote Access Trojan)**の挙動を分析対象とした。

### 2.2.3 Resource Description Network

RDF とはウェブ上にある任意の事物（リソース）を記述するために統一された枠組みであり，W3C によって標準化がなされている．特にリソース同士の関係について機械が処理できるように論理的な記述ができることを目的している[8]．

RDF のメタデータのモデルでは，記述対象のリソースである主語（**subject**），リソースの特徴や主語と目的語との関係を示す述語（**predicate**），主語との関係のある物や述語の値である目的語（**Object**）の 3 つの要素でリソースに関する関係情報を表現する．この（主語，述語，目的語）の三つ組を RDF トリプルまたはトリプルと呼ぶ．RDF トリプルは有向グラフでも表現することができる．この場合，主語と目的語を楕円ノードで，述語をラベルつき矢印で表現する．例えば，「PC1 の IP アドレスは 192.168.1.20 である」という情報は（PC1，IP アドレス，192.168.1.20）と表現できる．これを有向グラフで表すと図 2.4 のようになる．

RDF は元々，前述のようにリソースのメタデータを付加することで，コンピュータが効率よく情報を収集し解釈できるようにするセマンティック・ウェブを意識して作られているので，特定の環境を前提とせずに処理できる形で情報を表現できる．そのため，種々の端末やサービスで構築されているネットワーク空間のように膨大で多様な属性情報を表現するのに適している．

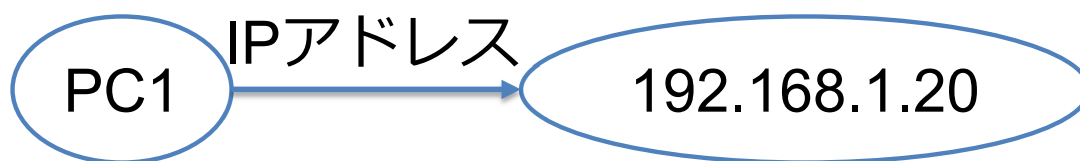


図 2.4 RDF トリプルの有効グラフ表現例

あるトリプルの目的語をリソースとすれば、それを主語として新たなトリプルを作成できる。さらに、別々に作成されたトリプルでも共通するノードを介して連結できるので、テーブルやツリーの形に限定されずに、大きく複雑なデータも表現することができる。また、これらのトリプルはノードが同じであることが識別できれば、異なる空間で記述されていても併合することができ、クラウド型の情報発信を可能にする。さらに、どんな複雑なグラフも単純なトリプルに分解することができ、シンプルな処理の組み合わせで大きなデータも扱うことが可能である。これらのことから再帰的に属性情報を詳細に定義することが可能であり、情報の追加・削除が容易かつ局所的、断片的な情報を用いた柔軟な検索も可能となる。

RDF には **SPAQL** という記述情報のパースや任意の情報の検索、構造の可視化のためのツールが提供されている。このため、アプリケーション開発者の手助けとなる。

さらに **OWL** によってオントロジー言語が提供されているので、あらゆる情報を一意的にモデリングすることができる。これにより、各ネットワーク間での情報連携が可能になることが期待される。

## 2.3 関連研究

### 2.3.1 サイバー攻撃推定手法

Ahmad らはネットワーク攻撃やセキュリティ管理に重要な役割を果たしている攻撃予測をモデル化するために効果的なアプローチを提案している[9]. 具体的にはオントロジーを使ったネットワーク攻撃の予測を目的とし、SDNのアラートを収集し、オントロジー用の推論ルール(SWRL: Semantic Web Rule Language)で予測を行う。この研究ではオントロジーは独自に作成して対象ネットワークをモデル化している。また、予測するための推論ルールには CAPEC の情報をもとに作成している。実験の結果、提案手法は誤警報を低減し、侵入検知の有効性を向上させたことがわかった。

この研究で収集する情報は IDS が主体であり、侵入検知後の各端末への影響などは検討されていない。しかし、標的型攻撃は IDS のみでは検知が困難である。つまり、その他の情報と連携してネットワーク全体の挙動を把握することが重要となる。本研究でも対象ネットワークのモデル化にオントロジーを検討している。しかし、収集する情報はネットワーク上の SDN アラートではなく端末のプロセスログであり、マルウェアのプロセスパターンを作成して攻撃を検知する手法をとる。そのため、より広範囲な端末に対して攻撃の検知が可能となる。

### 2.3.2 サイバー攻撃における情報共有手法

サイバー攻撃を検知する上で、その情報を即座に統一的に共有することは重要である。攻撃情報の表現するためには既存のネットワーク端末の情報の統一化は同様に重要である。またその際、利用者にわかりやすく提供することも同様に求められている。

藤巻は、トポロジが複雑化してきているホームネットワークに焦点をあてている[10]. ホームネットワークでは、例えば、端末は有線接続されていても、ホームルータとの間に無線や電力線通信区間が入る構成が起こりうる。そのため、ネット



ワークに関する知識の少ない一般ユーザにとって、複雑なネットワークから自身で障害発生箇所を切り分け、不具合を解決することは極めて困難である。したがって、ホームネットワークの管理し円滑に運用するため、トポロジ情報を自動的に検出し、ユーザや管理者に提供することが必要とされている。ホームネットワークに接続されたエンド端末やネットワーク機器から、トポロジを検出するための断片的な情報を収集する仕組みが実装されている。その仕組みの一つに、**HTIP(Home network Topology Identifying Protocol)**がある。この **HTIP** で収集してきたトポロジ情報の扱い方について学術的な議論はほとんどされていない。他にも、**SNMP** や **LLTD** など管理ツールを利用して、ネットワークのトポロジを検出する方法もあるが、いずれの方法も複雑なトポロジを形成し、多種多様なデバイスが接続されるホームネットワークの現状に即していない。そこで、提案手法であるトポロジ情報の記述は、多岐にわたる端末間、あるいはネットワーク技術間のトポロジをレイヤ横断的に記述することができ、ホームネットワークのトポロジ全体を把握することができる。また、ネットワーク記述モデルを利用することにより、トポロジ情報をコンピュータで読み込み可能な形式となる。これにより、ホームネットワークの障害原因の特定を容易化・自動化することが可能になる等、トポロジ情報の活用により、ホームネットワークの可用性の向上に貢献できるとしている。既存のネットワーク記述モデルのうち、トポロジ情報の記述に特化した関連技術として、**Physical Topology MIB**, **NDLs**, **YANG data model for Network Topology**, **NetJSON** が存在するが、この研究では **NetJSON** を拡張する形でモデル化を行った。さらに、この情報をもとにホームネットワークに接続された各機器より収集された接続構成情報からトポロジを自動的に検出し、ネットワーク記述モデルによってトポロジ情報を記述する手法を提案した。これにより、多岐にわたる端末間、あるいはネットワーク技術間のトポロジをレイヤ横断的に記述することができ、ホームネットワークのトポロジ全体を把握することを可能とする。

Cohen らはエンタープライズ向けの分散型フォレンジック・インシデントレスポンスを研究している[11]。このソフトウェアは、GRR プロジェクトとしてオープンソースでの開発が進められている[12]。現在、各ホストのシステム情報の表現には CybOX schema を採用している。GRR では DNS クラアント構成などのシステム情報は CybOX スキーマで記述している。このことから、静的情報に対する語彙は十分であると考えられる。

満永らは動的なネットワーク構成変更を可能にする Software Defined Networking (SDN) と CybOX を拡張したサイバー攻撃活動を記述するための技術仕様である Structured Threat Information eXpression (STIX) を組み合わせて、情報共有やインシデント対応を自動化することにより、セキュリティ被害の早期発見や未然防止を図る手法を研究している[13]。

上記のように、CybOX 以外にもネットワーク上の各端末の挙動を統一的に表現する手法は数多く提案されているが、それらはホームネットワークやデータセンター等、対象のネットワークを限定して作成されており、他の種類のネットワークへの拡張については今後の課題となっている。さらに、標的型攻撃などのサイバー攻撃自体の記述については検討されていない。それに対して、CybOX は上記で挙げた研究のように各種の端末の挙動を含め、さらにサイバー攻撃活動の記述も可能である。

しかし、近年メモリ情報などの揮発性情報が取得できるようになっており、動的情報の記述も今後増えていくと予想される。しかし、これまでの研究では静的な情報を記述した CybOX の利用にとどまっており動的情報の記述とその活用については検討されていない。本研究では新たに取得可能となった動的情報としてプロセスログを用いて CybOX の表現可用性を検証する。そのため、本研究では静的・動的両者の情報を総合的に利用できるためより幅広い検知が可能となる。

### 2.3.3 ネットワーク経路探索手法

本研究では感染経路を特定する手法を研究するため、検知手法にはグラフ化されたネットワークについて、感染源となる端末と感染された端末の探索を即座に実施する必要がある。そこで、本研究で活用するグラフ構造の探索に関する研究について調査した。

高橋らは、交通流シミュレーションにおける効率的な経路探索手法を研究している[14]。交通流シミュレーションでは、道路における渋滞の軽減など都市機能の円滑な運行のために道路という大規模ネットワークでの動的な経路探索を高速に行う必要がある。そこで、**GPGPU** を用いて処理時間の高速化を図った。アルゴリズムはラベル修正法を利用、地点をノード、経路をアークで表しネットワークモデルを作成した。この研究で検証したネットワークの規模はノード数 **20580**、アーク数は **34,210** であり、その検証結果は **CPU** で **27250** 秒、**GPU** で **220** 秒と大幅な短縮が可能となった。

また、**SDN** では **OpenFlow** を用いて行われているがその探索の効率性も問題となっている。小野の研究では医療情報ネットワークを対象とし、ネットワーク制御に **OpenFlow** を用いた[15]。これにより、単一のリンクに複数のフローが発生して輻輳が発生しても約 **7** 秒後に各フローに対して個別の経路を割り当ててスループットが向上できることが確認できた。また、フローが利用している経路中で障害が発生した場合には、短時間で経路が再設定されることを確認した。さらに、スケーラビリティを検証するため、計算機シミュレーションを用いて評価を行った。複数台の **OpenFlow** スイッチを用いたコアネットワークと現在の医療情報ネットワークで利用されているコアスイッチとの性能比較を行った。その結果、コアネットワークを構成する **OpenFlow** スイッチをスケールアウトすることによって、現状のコアスイッチに漸近する性能を発揮できることが示された。

これらの結果から、攻撃検知からの同様にネットワーク探索にかかる時間は今

後の研究により探索の効率化が図られている。そのため、本研究では感染源の特定にかかる時間の効率化ではなく、まず検知から感染源の特定に焦点をあてた。

これまで、プロセスログやネットワークトラフィックなどのログ情報から攻撃を検知する手法が数多く研究されてきた[15][16][17][18][19]。しかし、これらの研究では感染端末自体の特定は可能だが、プロセスレベルで感染経路を追跡することは検討されていない。したがって、感染端末の特定後に感染経路を追跡するには、各ログ情報を逐次突き合わせていく必要があり時間を要する。そこで、ある端末でマルウェアを特定した後に自動的に他の端末への感染経路まで調査するような、標的型メール攻撃の攻撃段階もふまえた診断手法が必要となる。

標的型攻撃での内部侵入・調査段階における検知方式として、川口らは不審活動の端末間伝搬に着目し、拡散活動を検出する手法を提案している[20]。この手法では不審性が高い端末が連鎖的に現れる現象を、被攻撃端末をノードとするグラフ構造として抽出する。そのグラフが基準を満たすとき、標的型攻撃の拡散活動が発生していると判断してアラートをあげる。この研究では、アラートされる不審端末が初期感染端末か否かを診断するような追加調査を実施する手法は検討されていない。一方、本研究では不審な端末が検知された後の追加調査としてプロセスレベルの感染経路検知を行うことで標的型攻撃を診断する研究である。したがって、本研究は川口らの手法を組み合わせる手法の研究として位置づけられる。

またこれまで、標的型メール攻撃対策として開発された製品の感染経路調査に関連した特徴を列挙し本研究との差異を述べる。IBMはネットワーク・セキュリティの脅威を検出して防御する先進的ソフトウェアとして **IBM Security QRadar Incident Forensics** がある。本研究との差異として、この製品は検知に特化しているが、本論文は上記のように異常検知後の正確な経路追跡を目的としている。したがって、川口らの手法との差異と同様に、結果を組み合わせる手法の研究として位置づけられる。また、Ciscoはファイルラジェクトリ機能というネットワ

ーク全体でファイルの送信を追跡することが可能な **Cisco AMP for Network** がある。これにより、例えば不審なファイルの送信履歴から攻撃者の感染拡大活動が追跡できる。著者らの研究ではさらにプロセスの振る舞いを記録しているため、ファイルの移動ではなくファイルを移動させたプログラムの振る舞いがある。そのため、不審ファイルの移動だけでなく、それを実行したマルウェアも含めた追跡ができると考えられる。

以上の内容をまとめると、本論文の新規性は次の 3 点を組み合わせることにより、プロセスレベルで標的型攻撃の感染経路を追跡できるようにしたことにある。

(1) 内部侵入・調査段階で頻繁に用いられる **PsExec** などのツールの挙動を解析することで内部感染の特徴を抽出

(2) **Onmitsu** により感染源となるプロセスも含めた感染経路の追跡が可能

(3) オントロジーにより検知処理が高速かつ感染経路の視覚的な把握が容易

## 第3章 マルウェアにより不正挙動をとるPCの検知方式の提案と評価

本章について、3.1節は拙著「Proposal for Knowledge Model using RDF-based Service Control for Balancing Security and Privacy in Ubiquitous Sensor Networks[21]」によるものであり、3.2節～3.6節は拙著「Proposal of a Method for Identifying the Infection Route for Targeted Attacks Based on Malware Behavior in a Network」[22]によるものである。

### 3.1 ネットワークの表示方法の検討

#### 3.1.1 センサネットワークと利用情報の適切な選択方法の検討

近年、IoTに注目があつまり、多種多様な機器がネットワークでつながってきている。これに関連して、以前はユビキタスセンサネットワークという、様々なセンサーやタグリーダーが空間内の情報を自動的に収集し、関連情報を取得するネットワークが検討されてきていた。これらのネットワークの進歩により、センサネットワーク空間における情報量がさらに増加している。さらに、ユーザに関する個人情報、様々な粒度で取得することが期待される。例えば、GPSは大まかな位置情報を取得することができ、カメラは詳細な位置情報を取得することができる。取得された情報の効率的な利用は、ユーザのプライバシー要件を満たす高品質なサービスを提供するために重要である。

しかし、センサーから得られる情報とそれ自身では些細なように思われるユーザ情報とを組み合わせることによって、ユーザの個人情報を識別することも可能である。このような間接的なプライバシー侵害の危険性があるため、ユーザのプライバシー情報を保護するためには得られた情報を制限する必要がある。その結果、質の高いサービスを提供することが困難となる。したがって、プライバシー情報とセキュリティのバランスと考慮する必要がある。プライバシー情報は第三者からの保護を必要とし、ユーザが欲するプライバシー要件を満たす必要がある。そしてセキュリティは、ネットワークが情報を外部に漏らさないことが求められる。さらにプライバシー情報を使用する端末は、暗号化や匿名性の保護などの手段によっ

て機密性を保持しなければならず、プライバシー情報漏えいや改ざんを防止するためのセキュリティ対策を課さなければならない。

そこで本研究では、リソース記述フレームワーク (RDF) を使用して、空間内のすべての情報を統合するプラットフォームを開発した[21]。RDF は、RDF トリプルの形でリソース (サブジェクト, 述語, オブジェクト) に関する情報を表す。RDF は、述語を通じてオブジェクトに関連付けられたリソースのサブジェクトを表す。推論規則と語彙セットとを組み合わせることによって、異なるタイプのデータを接続し、部分和に対する集計を行うことが可能である。RDF トリプルは、空間情報を任意の粒度で表現できる。したがって、センサネットワークにおけるサービス制御情報またはプライバシー情報が柔軟に表現できる。このため、情報を効率的に利用して柔軟なサービスを提供するためには、各サービスが必要とする空間の制御情報とサービス状態情報の RDF トリプルを構成する必要がある。

一方、個人情報保護するには、この情報を制限および適切な制御で収集する必要がある。本研究ではプライバシー保護を次のように定義する。「サービスは、ユーザの意図した情報のみを使用することができる。センサーは、ユーザに対して意図しない情報を収集することはできない。」この 2 点の条件が満たせればプライバシー保護ができるとした。

本研究ではこれまで、RDF ベースの実用的なサービスを使用してプラットフォームに適用できる知識モデルを提案した。RDF によって提供されるサービスを表現し、その時に得られた RDF を分析するのに必要な語彙セットである知識モデルを作成した。この知識モデルは、ユーザ情報を階層的に表現することでプライバシーを考慮しているため、ユーザの要求を反映した機能を追加することでユーザ情報を制御することができた。また、シミュレータの開発により、知識モデルの実現可能性を検証した。

作成した知識モデルは一般的かつ簡単な論理で表現されている。そのため、サービス提供者はサービス開発時にユーザの個人情報適切に使用できているかを検証できるという利点がある。そして、ユーザは個人情報がユーザの意思にしたがって制限されるという利点がある。

しかし、知識モデルの有用性である、様々な空間に様々なサービスを同様に適用できる点が検討・評価できていない。そのため、特定の空間に対してサービスを提供するという現在の段階では、他のサービス管理システムでも同様のことが可能と思われる。そこで、知識モデルの拡張を行うことでより様々な場所に適用できるように検討を行う。そして、シミュレータを用いて知識モデルの柔軟性を評価する。

センサネットワークのための様々な統合管理方法が提案されている[23]。なかでも、RDFを使用したセンサネットワーク情報を表す手法が存在する。藤波らはRDFを用いた位置モデルと物体モデルによる物理環境モデルを示している[24]。位置モデルは、部屋や建物などの単位空間と、入り口や台所などの単位領域との関係で表される。オブジェクトモデルは、仕様情報や動作条件などのオブジェクト情報で表される。これらのモデルを使用することで、開発者は様々なアプリケーションに必要な情報を直接処理できる。HeldらはRDFを使用して、ユーザの好みなどのユーザ固有の情報を記述した[25]。RDFでユーザプロフィールを管理し、コンテキスト情報を評価することにより、パーソナライズされたコンテキスト認識型のサービスコンテンツが可能となる。野口らは、家庭でインテリジェントな支援システムを実現するためにRDFを使用してセンサー情報を管理した[26]。部屋のセンサー構成などの情報を自動的に理解する仕組みとしてセンサー情報を包括的に描写するためのRDFセンサー記述を提案した。提案手法ではセンサーの特性を記述できるだけでなく、新しい情報を含む他の知識情報との協調による拡張も容易に実現できる。これらの機能により、センサデータの統一された処理が可能となった。適用



される RDF 記述の一例として、ミドルウェアにおけるコンポーネント発見のようなアプリケーションの実装を行った。

このように、RDF による記述は様々提案がされてきている。本研究では、サービス実行ルールとユーザ要求を、センサー情報と同じ RDF

で記述することで一元的に管理する。これにより、本研究の知識モデルは、高品質なサービスとプライバシー保護の両方を提供することができる。

上記では RDF による現実世界の情報をモデル化する方法を議論している。合わせて、RDF を使ったモデル化手法として Web 上のアクセス制御が研究されている。Sacco らは、細粒度のアクセス制御を可能にするプライバシー優先オントロジーを提案した[27]。このオントロジーには、RDF データの細かいプライバシー設定を定義するための語彙が用意されている。さらにこのオントロジーでは、リソース、特定のトリプル、およびトリプルのグループを制限するための語彙があり、プライバシー情報へのアクセス制御は、リクエスタが満たす必要があるプロパティによって制限される。Carminati らは、SWRL (Semantic Web Rule Language) を用いてプライバシールールを指定することにより、ソーシャルネットワークのためのアクセス制御フレームワークを提案した[28]。さらに、OWL (Web Ontology Language) を使用してユーザとリソースの関係をモデル化した。Web には多くのプライバシー情報が含まれているため、このようなアクセス制御はプライバシー保護の方法として有効となる。同様に、センサネットワークにおける RDF を用いたプライバシー保護も研究されてきた。Jagtap らは、RDF を使用してプライバシー保護を調査した[29]。この研究ではユーザの位置や環境、従事する活動を含むコンテキストのより包括的で高レベルな「気づき」についての表現と推論のためのモデルを提案している。この研究の重要な要素は、コンテキストに関する知識を統合し共有する機器を共有して協調的な情報を使用することである。その際に、プライバシーとセキュリティメカニズムの必要性が生じる。そこで、ユーザ情報の共有設定を記述する高レ

ベルで宣言型のポリシーを指定する際に、RDF を用いている。この研究では、モバイル機器に、情報から引き出すことができる推論を含め、収集している個人情報を保護するためにプライバシーの適切なレベルをユーザに提供するための枠組みを提示する。

本研究では、モバイルデバイスに適切なレベルのプライバシーを提供し、収集された個人情報（他の情報から推測できる個人情報を含む）を保護するためのフレームワークを提案する。これにより、様々なサービスを提供しながらプライバシーを保護することが可能となる。

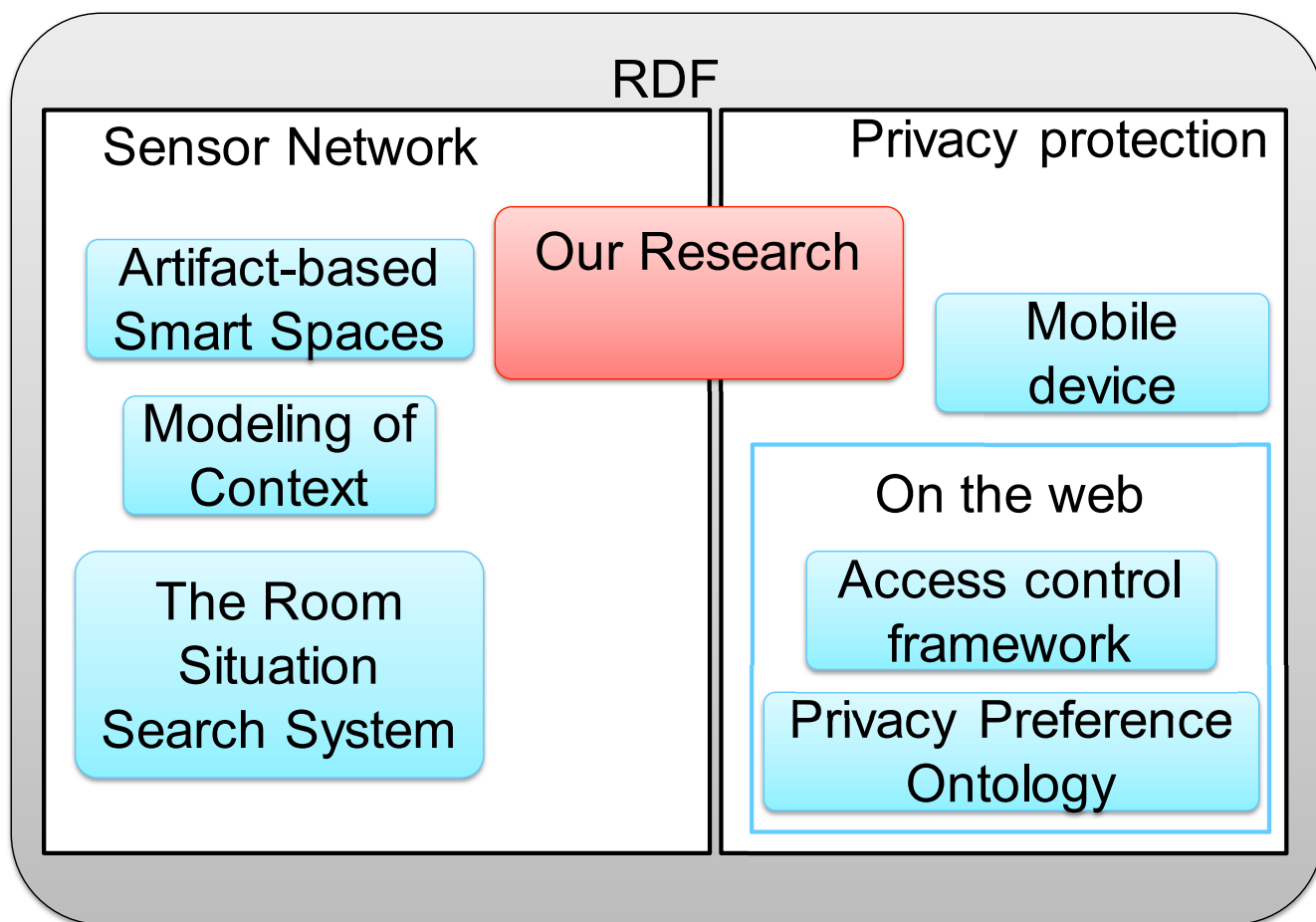


図 3.1 関連研究と本研究の領域

### 3.1.2 プラットフォームの開発

センサー情報，ユーザ情報，RDF を使用するためのサービス状態を記述することによって，空間内のすべての情報を統合する方法を研究してきた．この研究では，センサネットワーク空間内のサービスを，そのサービス制御に関する RDF トリプルで制御することにより，ユーザの要求に応じた情報提供やサービス提供を目指している．また，この研究では現在ユーザのプライバシー情報の保護と高品質なサービスの提供を両立させるために，ユーザの要求を RDF トリプルの使用権限へと反映させることにより，ユーザの要求を満たす適切な情報利用が可能なプラットフォームの開発を進めている．このプラットフォームの基本的なシステムの機能概要を図 3.2 に示す．

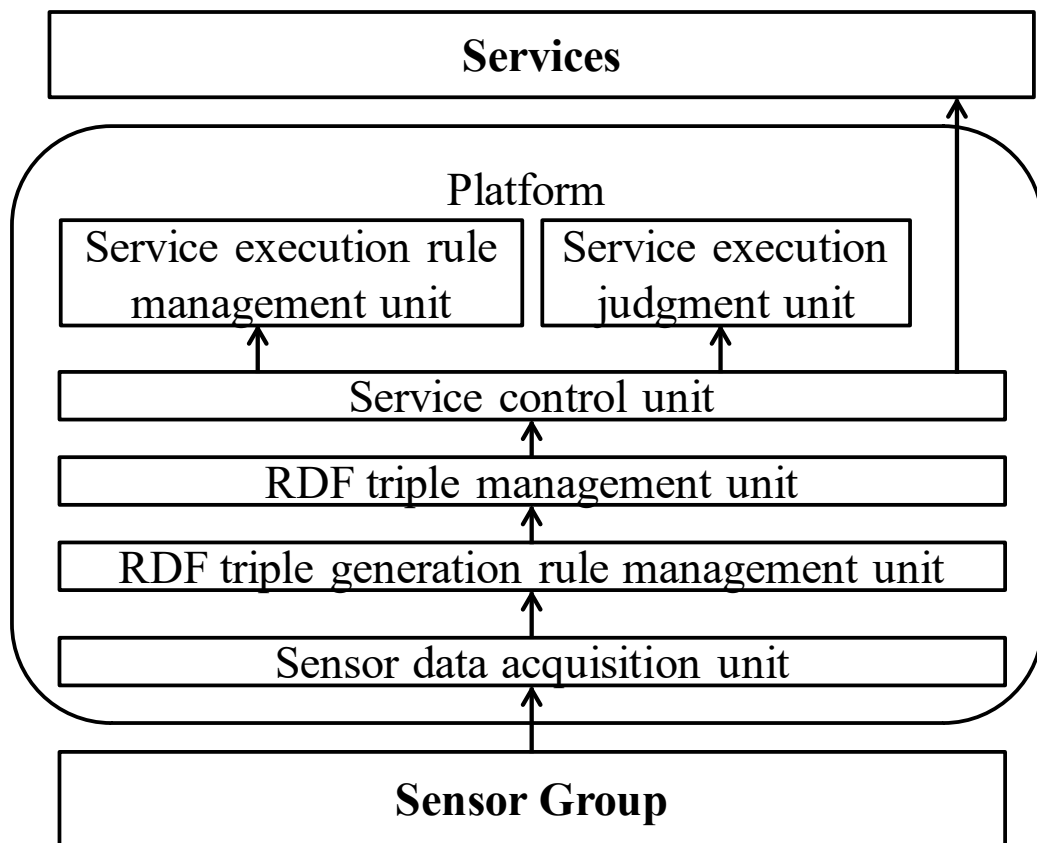


図 3.2 システムの機能概要

このプラットフォームの特徴は、センサーが取得した空間情報から RDF トリプルを生成することであり、RDF トリプルをもとに提供するサービスを選択することである。これらの処理は、それぞれ「RDF トリプル生成ルール管理部」と「サービス実行ルール管理部」で行われている。これらの機能についてより詳細に述べる。RDF トリプル生成ルール管理部では、RDF トリプル生成ルールに基づき、取得したセンサー情報などから RDF トリプルを生成する。また、ここでは RDF トリプル生成ルールを RDF トリプルが生成されるトリガーとなるルールと、生成される RDF トリプルを 1 組として管理している。RDF トリプル生成ルールの例を表 3.1 に示す。サービス実行ルール管理部では、サービス制御部から渡された RDF トリプルとサービス実行ルールを照合することで、提供可能なサービスを選定する。また、ここではサービス実行ルールを提供するサービスと提供するトリガーとなる RDF トリプル、サービスへの実行命令を 1 組として管理している。サービス実行ルールの例を表 3.2 に示す。

表 3.1 RDF トリプル生成ルール

Rule	Generated RDF triple
userA is located at (x, y)	(userA, locate, (x, y))

表 3.2 サービス実行ルール

Service	RDF triple			Excutive instruction
	Subject	Predicate	Object	
Lighting control	User	locate	Room	Light on

前項で述べたように空間情報は RDF で表現される。このとき、空間情報のうち、空間情報の語彙とその関係を RDF で表現したものを知識モデルと定義する。サー

ビス提供可能な知識モデルを作成するためには、想定されるサービスの要求要件が明らかにされた上で作成することが望ましい。しかし、このプラットフォーム上で動作するサービスは存在しない。そこで、はじめに基礎的な知識モデルを作り、それを徐々に拡張していくアプローチが有効であると考えられる。つまり、基礎的な知識モデルはプロトタイプとなるサービスを検討、評価しつつ、現実的な利用の観点から技術的な課題を明らかにした上で作成する。実際の流れとしては、想定するサービスに必要なリソースを洗い出す。次に、リソースの状態遷移を RDF グラフ (RDF トリプルの集合) で表現する。次に、その RDF グラフ内のリソースを同種の概念を持った集合ごとに分類し、それらの分類した集合同士の関係性を表現する知識モデルを作成する。そして、作成した知識モデルを同種のサービスに適用し、足りない概念が存在したら新たに導入する。このように知識モデルを拡張してより汎用的な知識モデルを作っていく。

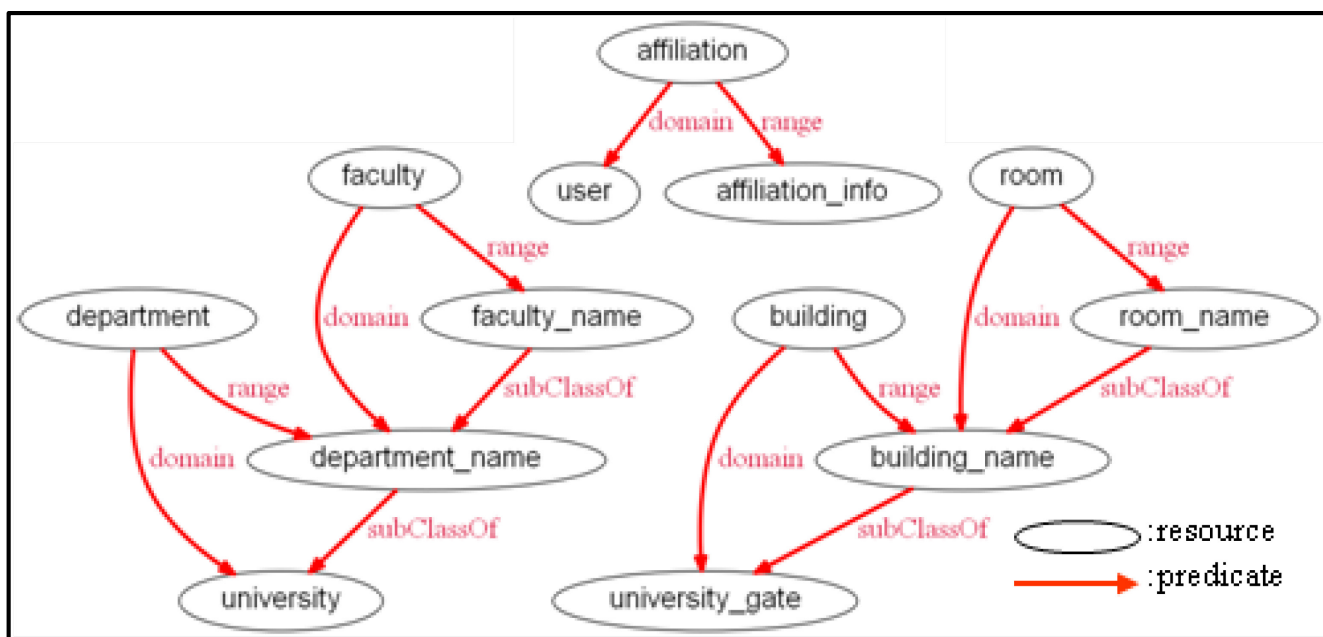


図 3.3 作成した知識モデルの例

このような流れで大学の組織を対象とした、図 3.3 のような知識モデルを作成してきた。この図では楕円がリソースを、矢印が述語を表現している。この知識モデ

ルの特徴は、(述語名, 主語, 主語名), (述語名, 目的語, 目的語名)というようにリソースの関係性を主体として、その関係の主語となるリソース, 目的語となるリソースというふうに表現する点である。これは、RDF トリプルを RDF グラフに追加する場合、述語に着目してリソースがどの分類に属するかという推論が可能であるためである。また、ユーザの所属情報を階層的に表現することで、ユーザのプライバシー情報の利用制限が可能だとわかった。これまで、この知識モデルを用いた机上実験をすることで、サービス実行ルールの柔軟性が増したことを確認している。

作成した知識モデルを用いてサービスを提供するシステムが存在しない。そのため、サービス実行ルールの格納手法など不明確な部分が存在する。そのため、知識モデルの実現可能性を明確に示すことができない。そこで、知識モデルを適用可能なシミュレータの開発を行った。

具体的には、RDF トリプルの入力や推論による RDF トリプルの導出、ユーザ要求反映、実行可能なサービスの選定や実行といった機能である。シミュレータの開発には Jena[30]を用いた。Jena は RDF を扱うフレームワークと推論エンジンを持っている。また、RDF グラフの可視化には Graphviz[31]を用いた。これまで、これらの機能を持つシミュレータを開発して各機能の動作検証を行った。また、知識モデルを用いたサービス制御の実現可能性についての検証も行った。

このシミュレータの特徴であるユーザ要求反映機能について述べる。この機能はセンサーから取得したユーザの所属情報をユーザの要求にしたがって利用の制限を行う。このとき、この機能はセンサネットワーク空間で利用する情報を制限する処理であるため、サービス利用前に利用制限に関するユーザ要求を入力する必要がある。また、ユーザ要求は推論ルールの形式で管理している。具体的な処理は、推論ルールを用いることで (user information, permit, no) という利用制限を表す

RDF トリプルを追加し、追加された RDF トリプルをもとに使用可能情報のみを出力する。

センサネットワーク空間における主体は、空間・ユーザ・サービスである。この内、空間は家、大学、駅などのサービスを提供するための「環境」と空間情報を取得する「センサー」とに分けられる。これまで、RDF を用いたサービス制御に焦点をあてたため、空間情報は「環境」のみと限定していた。そのため、サービス制御情報はサービスが直接必要とする情報として検討していた。このように、情報が得られる「センサー」についての検討が進んでいなかった。そこで、より汎用的な知識モデルを作成するための次の段階としてセンサーの概念を組み込む必要がある。また、このセンサーの概念に加えて、先行研究で検討された情報の収集制限に関する概念も加える必要がある。取得制限は先行研究において次のように検討した。より多くのユーザの要求を満たすため、すべてのユーザが使用を望まないセンサーのみを使用せず、残りのセンサーをすべて使用することとする。その際、ユーザの望まないセンサーから作られた RDF トリプルは使用しない。また、その RDF トリプルをそのユーザに対するサービス以外に適用しないというものである。

本研究で想定するサービスはユーザの所属情報を元に行う大学内の入室管理である。例えば、ユーザが大学に在籍していれば門のカギを開けるようなサービスである。この時、ユーザの所属情報はサービス管理者が適宜変更できるものとし、ユーザはサービスが利用できる所属情報を指定できるものとする。今回は大学構内には正門と、3つの棟、5つの部屋、そして食堂があると想定する。ユーザの所属と入室許可の関係性を図 3.4 に示す。

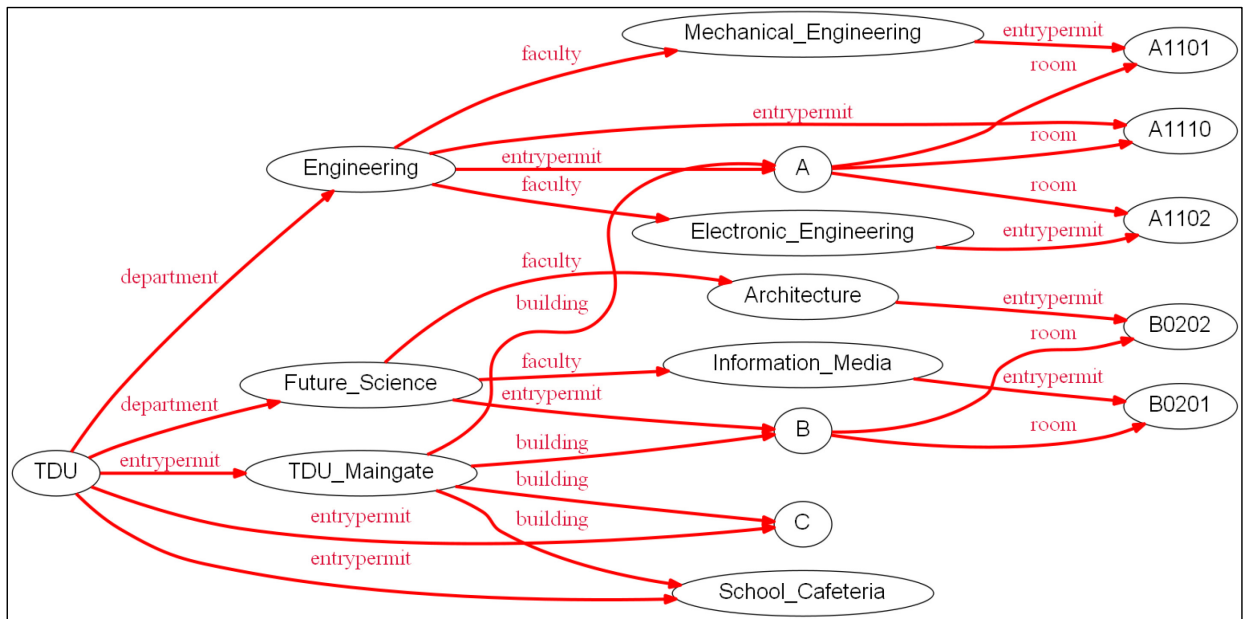


図 3.4 入室許可情報を示す RDF グラフ

この図は、大学敷地内に入るときにはユーザが大学に在籍していること、棟に入るには対応した学部にも所属していること、教室で専門科目を受けるなら対応した学科にも所属していることが必要なことを示している。このとき、A1110は学部情報だけで、食堂は大学に属していれば入室可能としている。この環境において、それぞれの入り口にサービスから出力される入室許可情報と連動する鍵が1つあるものとし、センサーによってユーザの所属情報を取得できるものとする。なお、所属情報は事前にユーザに割り振られているものとする。また、所属と各場所の入室可能性を示す RDF トリプルはサービス管理者によって入力されているものとする。

次に、想定サービスからサービス提供に必要な情報を抽出する。まず、サービス内容からサービス実行のトリガーとなる情報を分析し、サービス実行ルールを作成する。例えば、教室に入室する際はユーザによる所属情報の入力があり、入室を許可された場合に提供される。したがって、このトリガーとなる情報をそれぞれ RDF トリプルで (ユーザ, 所属, X 学科), (ユーザ, 許可, 教室 A10) と表現することができる。その結果取得できたサービス実行ルールを表 3 に示す。



Table1: Service execution rule for room entry

Service	RDF triple			Executive instruction
	Subject	Predicate	Object	
Entry Management	User	permit	Room	Open
	User	affiliation	Affiliation information	

サービスの「環境」である大学のモデルとユーザの所属情報のモデルをもとに知識モデルの拡張を図る。次に、空間情報を取得するセンサーについて検討する。センサーについては、すでに W3C で提唱されている Semantic Sensor Network Ontology (SSN)を用いる。これは、センサーとその振る舞いを記述するものである[32]。そのため、センサーに関する基礎的なモデルとしては十分であると考えられる。

このモデルを本研究で利用するためには収集制限を組み込む必要がある。収集制限は、ユーザが意図しない情報を収集することをセンサーに許可しないことで実現できる。例えば、カメラセンサは位置情報を取得することができるが、記録したくないユーザは、このカメラセンサを停止するよう要求する。センサーがユーザの要求を単純に反映することが許可されている場合、すべてのセンサーが停止する。しかし、センサーがすべて停止してしまうとサービスが提供できなくなる可能性が高い。そのため、効率性の観点から、空間内のすべてのユーザが情報を収集したくない場合にのみセンサーを停止することとした。この要求を満たすため、センサーの利用可能性を記述するために、新たな述語「hasAvailability」を追加した。この述語は、ユーザとセンサーとの間の関係である。さらに、接頭辞「ssn」を持つ単語は SSN の語彙であることを示す。図 3.5 は、この情報に基づく拡張された知識モデルを示す。

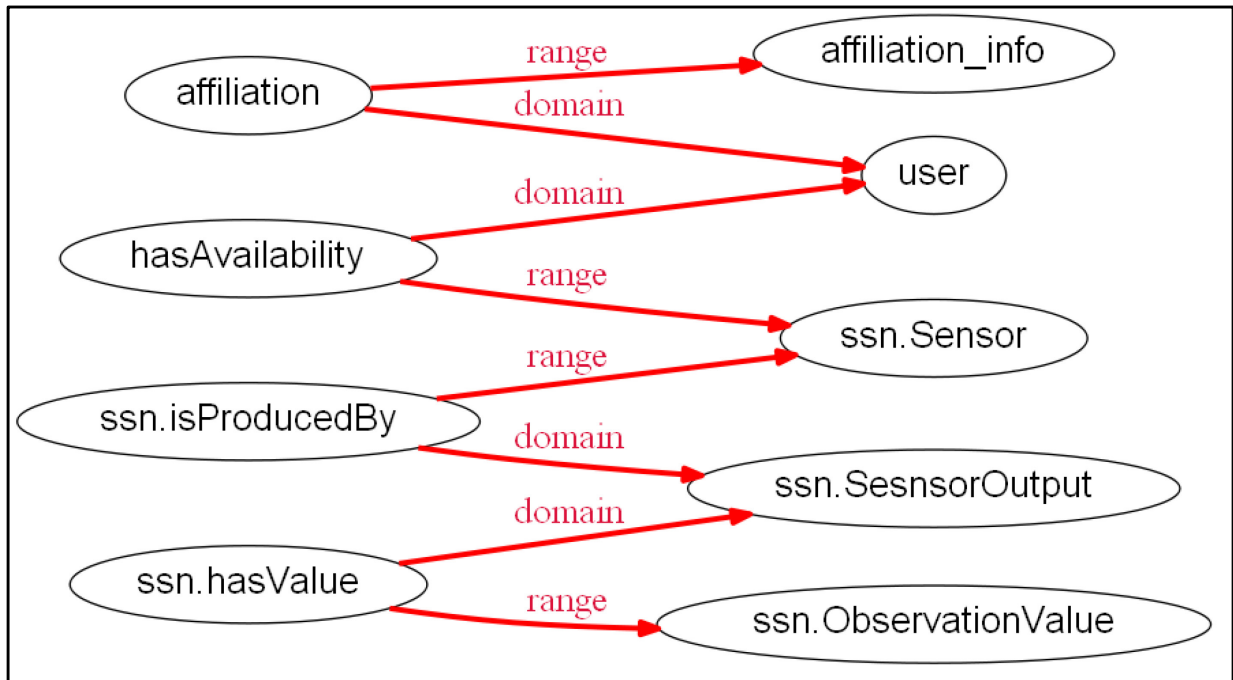


図 3.5 拡張した知識モデルの一部

さらに、収集制限についての新しい推論ルールを導入する必要がある。収集制限は、SWRL を使用して次の式で実現した。ユーザが「hasAvailability」述部を持たない場合、シミュレータはユーザの所属情報を使用しない（式 1）。RDF トリプル（ユーザ, hasAvailability, センサー）が追加されていない場合、センサーは機能しない（数式 2）。

$$\begin{aligned}
 & \text{Affiliation}(?user, ?affiliationof) \wedge \\
 & \text{noValue}(?user, \text{ssh.hasAvailability}) \\
 & \rightarrow \text{Permit}(?affiliationof, \text{no})
 \end{aligned}$$

式 1 収集制限を表現した推論規則(1)

# noValueof(ssh.hasAvailabilityBy, ?Sensor) → Permit(?Sensor, down)

## 式 2 収集制限を表現した推論規則(2)

### 3.1.3 シミュレーション実験

この項では、シミュレーション実験について述べる。この実験では、拡張した知識モデルがユーザの所属情報を使用してプライバシー情報を保護できることを検証する。

実験には以下の仮定を置いた。各ユーザは IC カードを所持している。大学における物理的な場所を想定する。物理的な場所は、緯度と経度の地理座標によって一意に識別することができるものとする。また、大学のキャンパス上のすべての場所は、公的機関によって割り当てられた識別可能な文字列の名前を持っているものとする。そして、センサーは、各場所のドアまたはゲートの近くに設置されているものとする。この空間で使用されるセンサーは、カメラセンサと IC カードリーダーとする。カメラセンサは、空間に存在するユーザの名前情報を取得する。IC カードリーダーは、ユーザの所属情報を取得する。個人の認証は、IC カード内の情報とカメラセンサによって取得された情報とを比較することによって行う。なお、取得されたセンサー情報は自動的に RDF トリプルに変換されることとする。

表 3.3 入力される RDF トリプルの一部

Subject	Predicate	Object
University	department	Engineering
University	department	Future Science
Future Science	faculty	Information Media
A	room	A1101
University	entrypermit	Maingate
Engineering	entrypermit	A
Information Media	entrypermit	A1110

本実験では、以下の2つのシナリオを想定する。2つのシナリオの違いは、センサー情報入力が入力されるシナリオ A でのみ追加されることである。そして、サービス実行ルールおよび知識モデルは、データベースに予め格納されているものとする。そしてユーザは、大学の工学部の学生であると想定する。

シナリオ A :

- 1) 想定空間の初期状態を入力する。具体的には、サービス管理者は、建築許可証およびスペース情報を示す RDF トリプルをシミュレータに入力する(表 3.3)。
- 2) ユーザ要求を入力する。ユーザが使用可能なユーザ情報と利用可能なセンサーを選択し、それらをシミュレータに入力する。シナリオ A では、すべてのセンサーとユーザ情報が利用可能とする。
- 3) 取得した所属情報等のセンサー情報をシミュレータに入力する。
- 4) 入力情報を用いて推論処理を行い新たな RDF トリプルを生成する。
- 5) RDF グラフから利用して入室管理サービスが実行できるかどうかを判断する。
- 6) サービスが入室可能な場所を提示する。

シナリオ B :

シナリオ A と同様の手順が実行されるが，上記のステップ 2 でセンサーを使用せず収集制限をかける．

シナリオがシミュレータによって実行された結果を述べる．図 3.6 は，ステップ 1 の結果の一部を示している．スマートカードおよびカメラセンサが SSN に関連付けられていることがわかる．

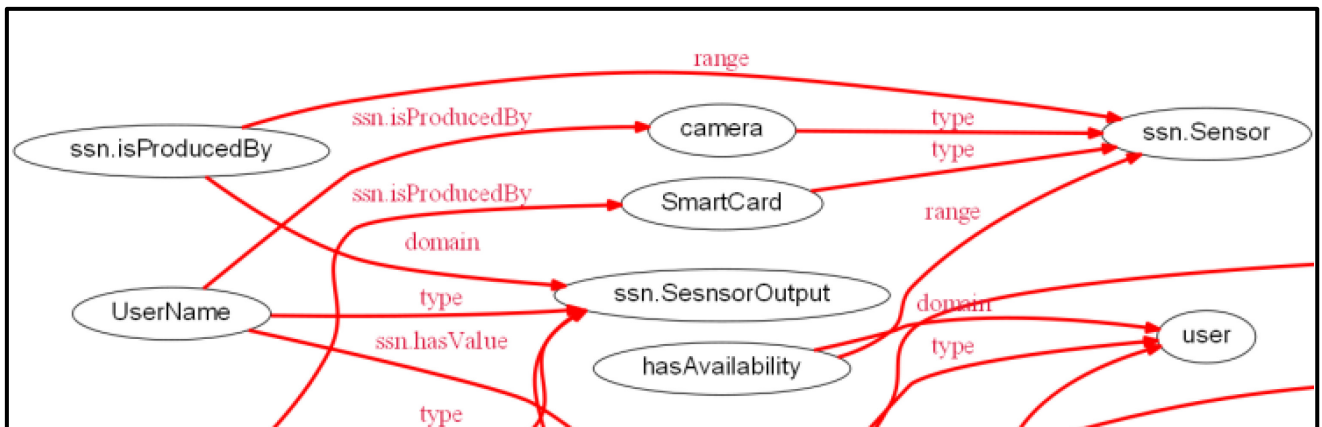


図 3.6 知識モデルの一部

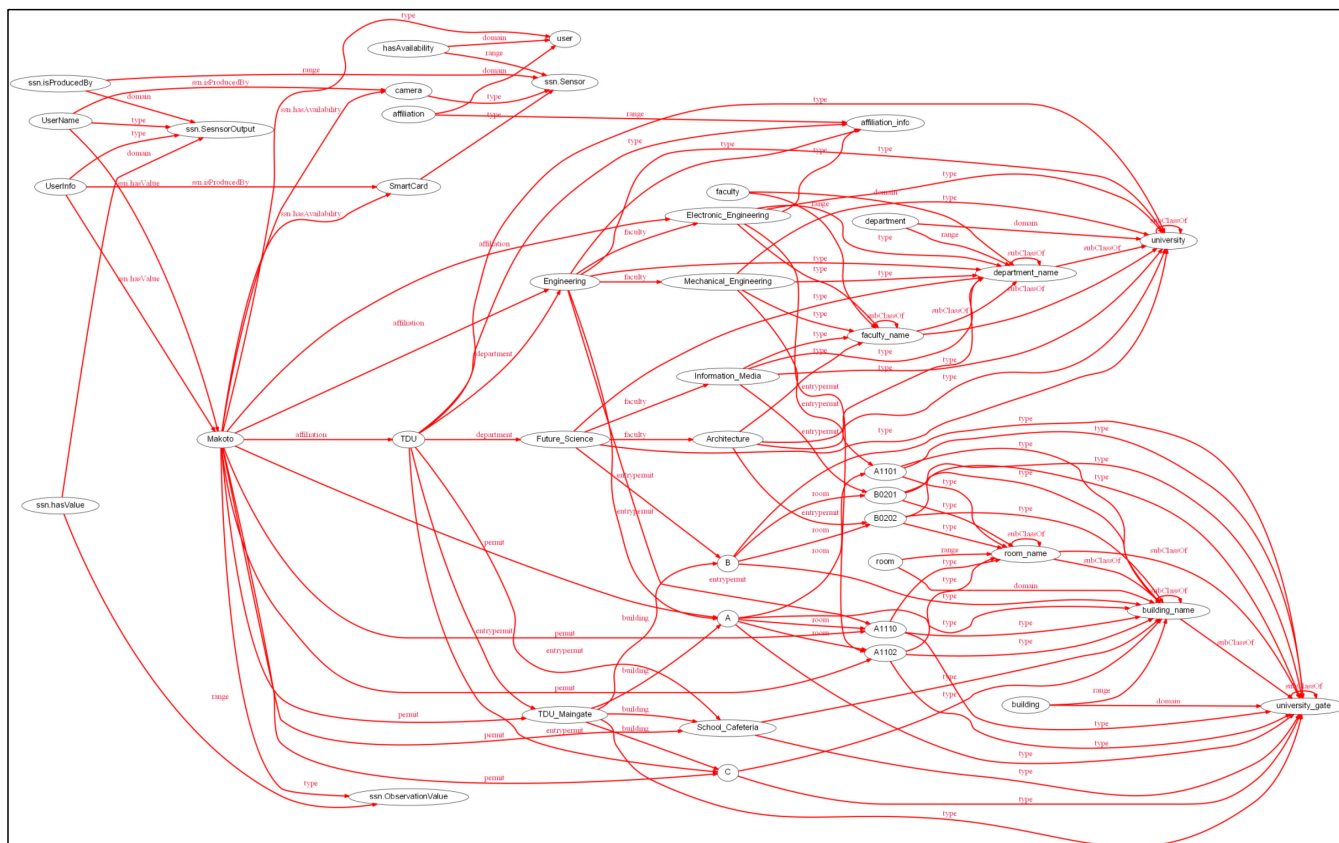


図 3.7 推論処理後の RDF グラフ (Scenario A)

図 3.7 は、シナリオ A のステップ 4 の結果を示す。この図から、推論処理によって、多くの RDF トリプルが自動的に生成されることがわかる。図 3.8 は、ユーザが入力を許可されている部屋のリストを示す。

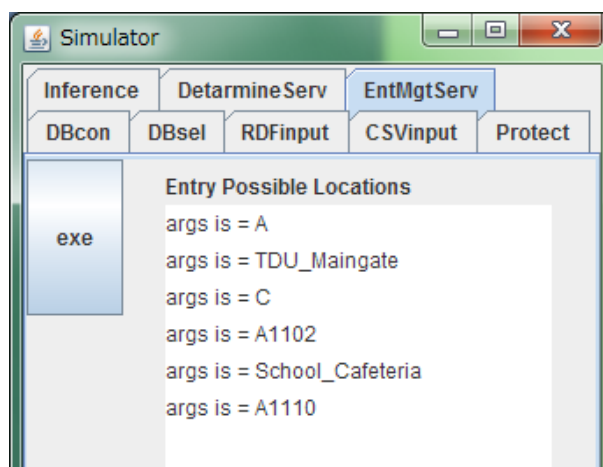


図 3.8 ユーザが入室可能な部屋のリスト (Scenario A)

図 3.9 は、シナリオ B のステップ 2 で入力されたユーザ要求に対する推論規則収集制限を適用した結果を示す。ユーザが「hasAvailability」述語を持たないため、利用可能な情報が削除された。具体的には図 3.9 の上部にある RDF グラフは、ユーザの要件を示す。図 3.9 の中央の RDF グラフは、式 1 の推論ルールと図 9 の上部の RDF グラフから導き出すことができる。一例は(Makoto, affiliation, Electronic Engineering)である。"Makoto"サブジェクトには "hasAvailability"述語がない。したがって、(Electronic Engineering, use\_permit, no)が生成される。図 3.9 の下の RDF グラフは、ユーザの所属情報が削除されたことを示している。このため、拡張知識モデルを用いてユーザに応じた収集制限ができ。さらに、導かれた RDF グラフは、ユーザ所属情報が使用できないことを示している。したがって、利用者の要求を満足するプライバシー情報の制限が満たされていると考えられた。しかし、図 3.9 の下の RDF グラフにはユーザ名を示すリソースが残っていたため、このプライバシー情報を削除する。

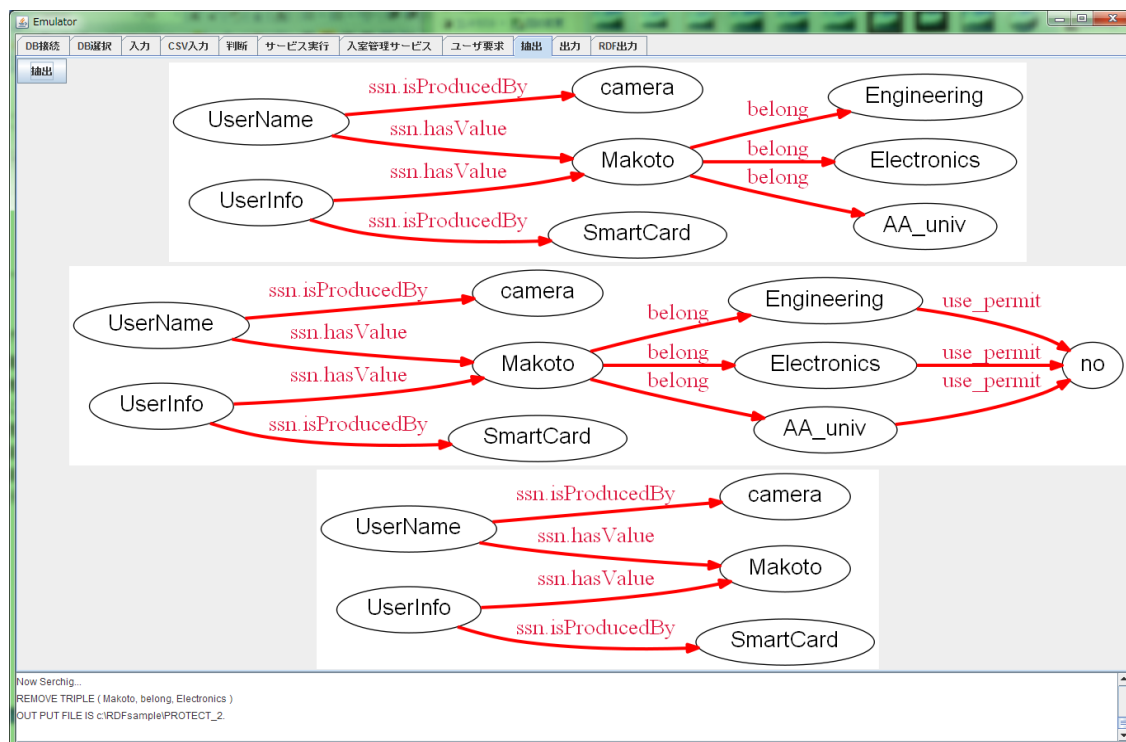


図 3.9 収集制限の推論規則を適用した RDF グラフ

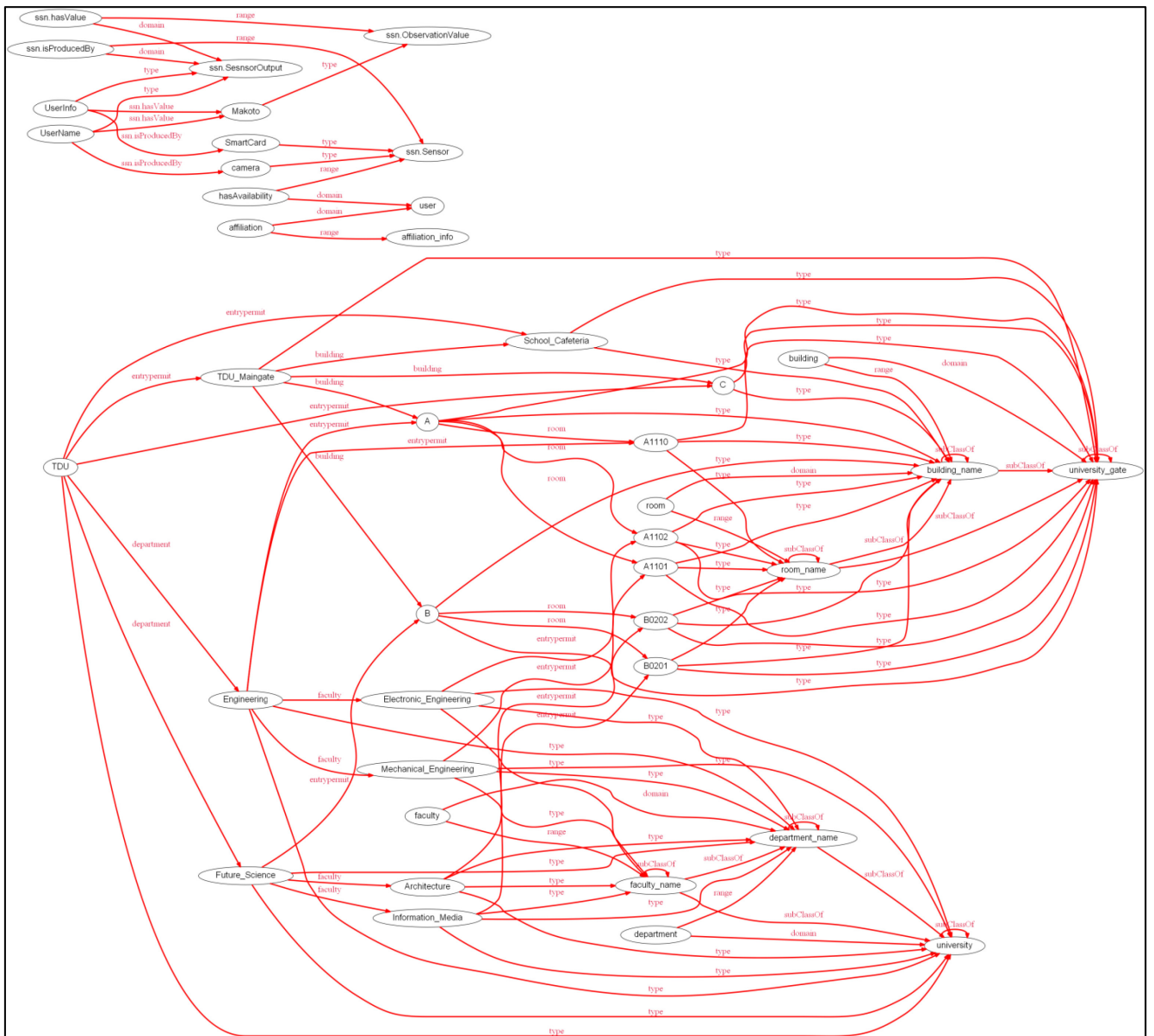


図 3.10 推論処理後の RDF グラフ (Scenario B)



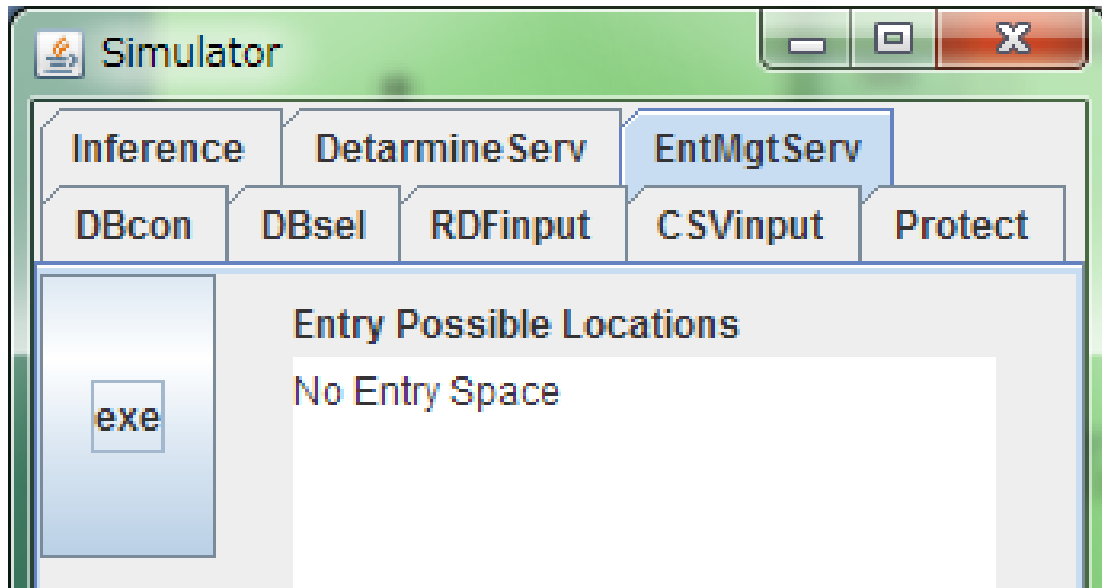


図 3.11 ユーザが入室可能な部屋のリスト (Scenario B)

図 3.10 は、ステップ 4 の結果を示す。図 3.7 と比較すると、図 3.10 は、RDF グラフが 2 つのグループに分断していることがわかる。これは、利用者の所属情報が入力許可情報に拘束されていないためである。図 3.11 は、ユーザが入力を許可されている部屋のリストを示す。

シナリオ A では、すべてのユーザ情報が使用可能であると想定されている。図 3.4 とユーザの所属情報(University: TDU, department: Engineering, faculty: Electronic Engineering)と比較すると、許可されるべきすべての部屋がわかる。その結果、図 3.8 に示すように、入室管理サービスはすべての許可された部屋を提示している。シナリオ B では、すべてのセンサーが利用可能ではないと考えられる。したがって、許可された部屋は存在しない。図 3.11 は、入室管理サービスが入室可能な部屋がないことを示している。このため、許可された部屋への入室は、新しく定義された空間に適切に記述された。そして、2 つのシナリオでは、サービスの提供を自動的に実行できることを確認した。この結果により、

様々な空間における知識モデルを用いてセンサネットワーク空間の実現可能性を示すことができた。

### 3.2 検知方式の検討

近年、サイバー攻撃が多様化している。その中でも、標的型攻撃が大きな脅威になっている。標的型攻撃とは、組織がもつ知的財産に関わる機密性の高い情報の取得を目的としたサイバー攻撃の一種である。標的型攻撃はその目的から、一度でも攻撃が成功すると組織に甚大な被害を及ぼす。そのため、標的型攻撃への対策は今後さらに重要になると考えられる。標的型攻撃の一般的な手法は、偽装メールと不正プログラム（マルウェア）を組み合わせる方法である。これにより、特定のユーザに対して、特定の行為に誘導することで不正な情報の取得を行う。さらに、標的型攻撃の攻撃対象は偽装メールを受けた端末のみにとどまらずネットワーク全体の端末に及ぶ。したがって、標的型攻撃に適切に対処するためには、端末内で起きた事象の解析だけでなく、これらの情報を組み合わせて解析して判断する必要がある。また、事象情報を組み合わせて解析する際には、事象情報の表現形式が統一されていることが望ましい。さらに、解析結果はネットワーク全体の事象として表現する必要がある。

そこで、これらの要求を満たすために本研究では標的型攻撃診断のための標準を利用したマルウェア検知方式を検討する。

本研究室ではこれまで、標的型攻撃の原因を特定するための研究としてプロセス情報とそのプロセスが発した通信に関するログを記録する手法を検討しており、現在、常駐型のプログラムが開発されている。これにより、時間と共に取得が困難となるが攻撃事象の事実関係の把握や法的証拠として有用である揮発性情報の取得が容易にできる。取得したログは端末間で連携させることでネッ

トワーク内での相互連携を図る必要がある。そこで、情報を記述して解析するための統一した表現として、本研究では **CybOX** を利用する。**CybOX (Cyber Observable eXpression)**とは、サイバー攻撃の観測事象を記述するための仕様である。この標準を使うことで事象情報は統一された表現形式で記述でき情報交換が容易になる。**CybOX** は現在様々な場所で利用が検討されている。例えば、**IPA** は **IPA** が保有する観測事象を **CybOX** 言語で記述することを検討している。また、**Microsoft** が開発しているセキュリティや脅威情報のナレッジを交換するためのプラットフォームである **Microsoft Interflow** でも **CybOX** への対応を検討している。このように、サイバー攻撃の振る舞いが **CybOX** で記述されることが今後一般的になると予想される。しかし、その表現可用性の評価はほとんど検討されていないため、その評価も同時にする必要がある。

そこで、本研究では標準をベースとしたプロセスパターンを用いたマルウェア検知手法を提案する。なお、プロセスパターンの作成には前述したプロセスログを使用する。本手法では、標準を利用しているため情報共有が容易であり、また共有される多大な情報を容易に使えるためマルウェアの個別検知精度が向上することが期待される。さらに、解析結果をオントロジーで表現することを検討している。オントロジーを使うことでシステムは攻撃行動を予測できるようになり、その結果これまでのように各端末内の警告だけではなく、ネットワーク全体で汚染された範囲を視覚化するなど、時々刻々と変わる事象に合わせた総合的な対策ができると考えている。今回は **CybOX** の記述可用性について検証し、提案手法を用いて情報の相互連携が適切になされているかを確認した。

### 3.3 プロセスログの CybOX 変換手法の検討

プロセスログを **CybOX** で表現可能か検討した。まず大まかな構成として、プロセスの動作を記録していることをから、アクションを記述する **Event** をベースにした構成を考えた。次に、プロセスログの必須項目であるログ取得時間について検討した。ログ取得時間はアクションが起こった時間であると考えたので **Action timestamp** 内に記述した。次に、プロセスログで記録された各項目が **CybOX** のどの項目に対応するのか検討した。その結果をプロセスタイプに分けて説明する。なお、プロセスタイプはアクションを分類していると考えて、**CybOX Schema** で定義されている **Action Type, Action Name** から適当な語彙を選択した。

プロセスの起動(**PROCESS\_LAUNCH**)の記述内容を図 3.12 に示す。**Action Type** として **start** を、**Action Name** として **open process** の語彙を選択した。プロセス ID、親プロセス ID とファイルパスはそれぞれ関連するオブジェクトである **ProcessObj** と **FileObj** に記述した。コマンドラインはアクションを起こした際の引数と考えて、**Action\_Argument** に記述した。

```

<cybox:Observable id="example:Observable-6d9dad4e-6f82-4ad5-b99b-5207bde216c7">
  <cybox:Event>
    <cybox:Type xsi:type="cyboxVocabs:EventTypeVocab-1.0.1">Process Mgt</cybox:Type>
    <cybox:Actions>
      <cybox:Action timestamp="2014-12-13T16:05:46.095500">
        <cybox:Type xsi:type="cyboxVocabs:ActionTypeVocab-1.0">Start</cybox:Type>
        <cybox:Name xsi:type="cyboxVocabs:ActionNameVocab-1.1">Open Process</cybox:Name>
        <cybox:Action_Arguments>
          <cybox:Action_Argument>
            <cybox:Argument_Name xsi:type="cyboxVocabs:ActionArgumentNameVocab-1.0">Command</cybox:Argument_Name>
            <cybox:Argument_Value>C:\Windows\system32\SearchFilterHost.exe 0 548 552 560 65536 556 </cybox:Argument_Value>
          </cybox:Action_Argument>
        </cybox:Action_Arguments>
        <cybox:Associated_Objects>
          <cybox:Associated_Object id="example:Process-7fc5d47b-b0df-46f2-af47-cada36b570c9">
            <cybox:Properties xsi:type="ProcessObj:ProcessObjectType">
              <ProcessObj:PID>3120</ProcessObj:PID>
              <ProcessObj:Parent_PID>2620</ProcessObj:Parent_PID>
              <ProcessObj:Start_Time>2014-12-13T16:05:46.095500</ProcessObj:Start_Time>
            </cybox:Properties>
          </cybox:Associated_Object>
          <cybox:Associated_Object id="example:File-370f4374-9769-45b7-88d9-4f12c4f68061">
            <cybox:Properties xsi:type="FileObj:FileObjectType">
              <FileObj:File_Path>??C:\Windows\system32\SearchFilterHost.exe</FileObj:File_Path>
            </cybox:Properties>
          </cybox:Associated_Object>
        </cybox:Associated_Objects>
      </cybox:Action>
    </cybox:Actions>
  </cybox:Event>
</cybox:Observable>

```

図 3.12 プロセス起動の CybOX 記述例

次にプロセスの停止(PROCESS\_QUIT)の記述内容を図 3.13 に示す。Action Type として Kill を，Action Name として Kill Process を選択した。プロセス ID は関連するオブジェクトである ProcessObj に記述した。

```

<cybox:Observable id="example:Observable-d6853201-2972-428b-b17e-03790466330d">
  <cybox:Event>
    <cybox:Type xsi:type="cyboxVocabs:EventTypeVocab-1.0.1">Process Mgt</cybox:Type>
    <cybox:Actions>
      <cybox:Action timestamp="2014-12-13T16:05:53.002400">
        <cybox:Type xsi:type="cyboxVocabs:ActionTypeVocab-1.0">Kill</cybox:Type>
        <cybox:Name xsi:type="cyboxVocabs:ActionNameVocab-1.1">Kill Process</cybox:Name>
        <cybox:Associated_Objects>
          <cybox:Associated_Object id="example:Process-62e023ea-4023-431c-8e82-24a287f007ce">
            <cybox:Properties xsi:type="ProcessObj:ProcessObjectType">
              <ProcessObj:PID>3796</ProcessObj:PID>
            </cybox:Properties>
          </cybox:Associated_Object>
        </cybox:Associated_Objects>
      </cybox:Action>
    </cybox:Actions>
  </cybox:Event>
</cybox:Observable>

```

図 3.13 プロセス停止の CybOX 記述例

次にモジュール読み込み(PROCESS\_MODLOAD)の記述内容を図 3.14 に示す。Action Type として Load を， Action Name として Load Module を選択した。プロセス ID とファイルパスはそれぞれ関連するオブジェクトである ProcessObj と FileObj に記述した。

```
<cybox:Observable id="example:Observable-964bf491-00ae-49d0-bccf-db7eaf8560fb">
  <cybox:Event>
    <cybox:Type xsi:type="cyboxVocabs:EventTypeVocab-1.0.1">Process Mgt</cybox:Type>
    <cybox:Actions>
      <cybox:Action timestamp="2014-12-13T16:05:53.002400">
        <cybox:Type xsi:type="cyboxVocabs:ActionTypeVocab-1.0">Load</cybox:Type>
        <cybox:Name xsi:type="cyboxVocabs:ActionNameVocab-1.1">Load Module</cybox:Name>
        <cybox:Associated_Objects>
          <cybox:Associated_Object id="example:Process-ca6857d6-95a2-4396-8fa2-32dec61af471">
            <cybox:Properties xsi:type="ProcessObj:ProcessObjectType">
              <ProcessObj:PID>1844</ProcessObj:PID>
            </cybox:Properties>
          </cybox:Associated_Object>
          <cybox:Associated_Object id="example:File-3e04b7c2-09d3-4169-a03b-3ecf07dedbeb">
            <cybox:Properties xsi:type="FileObj:FileObjectType">
              <FileObj:File_Path>\PROGRA~1\COMMON~1\MICROS~1\IME14\IMEJP\IMJPTIP.DLL</FileObj:File_Path>
            </cybox:Properties>
          </cybox:Associated_Object>
        </cybox:Associated_Objects>
      </cybox:Action>
    </cybox:Actions>
  </cybox:Event>
</cybox:Observable>
```

図 3.14 モジュール読み込みの CybOX 記述例

次に、通信試行(NETWORKV4)の記述内容を図 3.15 に示す。項目はすべてネットワーク接続の情報なので関連するオブジェクトである NetworkConnectionObj に記述した。その中で、接続元 IPv4 アドレス、接続元ポート番号を Source\_Socket\_Address に、接続先 IPv4 アドレス、接続先ポート番号を Destination\_Socket\_Address に記述した。しかし、プロトコル番号自体を記述する箇所は存在しない。

```
<cybox:Observable id="example:Observable-8917278f-101a-4183-9859-840decf5c6b9">
  <cybox:Event>
    <cybox:Type xsi:type="cyboxVocabs:EventTypeVocab-1.0.1">Process Mgt</cybox:Type>
    <cybox:Actions>
      <cybox:Action timestamp="2014-12-13T16:06:28.079500">
        <cybox:Type xsi:type="cyboxVocabs:ActionTypeVocab-1.0">Connect</cybox:Type>
        <cybox:Name xsi:type="cyboxVocabs:ActionNameVocab-1.1">Connect to IP</cybox:Name>
        <cybox:Associated_Objects>
          <cybox:Associated_Object id="example:NetworkConnection-6f234887-b81a-4c85-a192-4584f365fdff">
            <cybox:Properties xsi:type="NetworkConnectionObj:NetworkConnectionObjectType">
              <NetworkConnectionObj:Layer3_Protocol>IPv4</NetworkConnectionObj:Layer3_Protocol>
              <NetworkConnectionObj:Source_Socket_Address xsi:type="SocketAddressObj:SocketAddressObjectType">
                <SocketAddressObj:IP_Address xsi:type="AddressObj:AddressObjectType">
                  <AddressObj:Address_Value>192.168.21.39</AddressObj:Address_Value>
                </SocketAddressObj:IP_Address>
                <SocketAddressObj:Port xsi:type="PortObj:PortObjectType">
                  <PortObj:Port_Value>56647</PortObj:Port_Value>
                </SocketAddressObj:Port>
              </NetworkConnectionObj:Source_Socket_Address>
              <NetworkConnectionObj:Destination_Socket_Address xsi:type="SocketAddressObj:SocketAddressObjectType">
                <SocketAddressObj:IP_Address xsi:type="AddressObj:AddressObjectType">
                  <AddressObj:Address_Value>210.130.0.1</AddressObj:Address_Value>
                </SocketAddressObj:IP_Address>
                <SocketAddressObj:Port xsi:type="PortObj:PortObjectType">
                  <PortObj:Port_Value>53</PortObj:Port_Value>
                </SocketAddressObj:Port>
              </NetworkConnectionObj:Destination_Socket_Address>
            </cybox:Properties>
          </cybox:Associated_Object>
          <cybox:Associated_Object id="example:Process-f40f397b-482d-4d18-8ccd-729b2400404d">
            <cybox:Properties xsi:type="ProcessObj:ProcessObjectType">
              <ProcessObj:PID>356</ProcessObj:PID>
            </cybox:Properties>
          </cybox:Associated_Object>
        </cybox:Associated_Objects>
      </cybox:Action>
    </cybox:Actions>
  </cybox:Event>
</cybox:Observable>
```

図 3.15 通信試行(NETWORKV4)の CybOX 記述例

次に、通信試行(NETWORKV6)の記述内容を図 3.16 に示す。項目はすべてネットワーク接続の情報なので関連するオブジェクトである NetworkConnectionObj に記述した。その中で、接続元 IPv6 アドレス、接続元ポート番号を Source\_Socket\_Address に、接続先 IPv6 アドレス、接続先ポート番号を Destination\_Socket\_Address に記述した。しかし、プロトコル番号自体を記述する箇所は存在しない。

```

<cybox:Observable id="example:Observable-3d3e6302-723c-4017-9a21-47b608394709">
  <cybox:Event>
    <cybox:Type xsi:type="cyboxVocabs:EventTypeVocab-1.0.1">Process Mgt</cybox:Type>
    <cybox:Actions>
      <cybox:Action timestamp="2014-12-13T16:06:32.026100">
        <cybox:Type xsi:type="cyboxVocabs:ActionTypeVocab-1.0">Connect</cybox:Type>
        <cybox:Name xsi:type="cyboxVocabs:ActionNameVocab-1.1">Connect to IP</cybox:Name>
        <cybox:Associated_Objects>
          <cybox:Associated_Object id="example:NetworkConnection-4ec6ba77-9df7-4961-af9c-e7f00eb18cf4">
            <cybox:Properties xsi:type="NetworkConnectionObj:NetworkConnectionObjectType">
              <NetworkConnectionObj:Layer3_Protocol>IPv6</NetworkConnectionObj:Layer3_Protocol>
              <NetworkConnectionObj:Source_Socket_Address xsi:type="SocketAddressObj:SocketAddressObjectType">
                <SocketAddressObj:IP_Address xsi:type="AddressObj:AddressObjectType">
                  <AddressObj:Address_Value>fe80:0:0:0:1001:a03e:947a:518a</AddressObj:Address_Value>
                </SocketAddressObj:IP_Address>
                <SocketAddressObj:Port xsi:type="PortObj:PortObjectType">
                  <PortObj:Port_Value>63567</PortObj:Port_Value>
                </SocketAddressObj:Port>
              </NetworkConnectionObj:Source_Socket_Address>
              <NetworkConnectionObj:Destination_Socket_Address xsi:type="SocketAddressObj:SocketAddressObjectType">
                <SocketAddressObj:IP_Address xsi:type="AddressObj:AddressObjectType">
                  <AddressObj:Address_Value>ff02:0:0:0:0:1:3</AddressObj:Address_Value>
                </SocketAddressObj:IP_Address>
                <SocketAddressObj:Port xsi:type="PortObj:PortObjectType">
                  <PortObj:Port_Value>5355</PortObj:Port_Value>
                </SocketAddressObj:Port>
              </NetworkConnectionObj:Destination_Socket_Address>
            </cybox:Properties>
          </cybox:Associated_Object>
          <cybox:Associated_Object id="example:Process-69e6f06b-27fc-473f-b677-cd40e16ee342">
            <cybox:Properties xsi:type="ProcessObj:ProcessObjectType">
              <ProcessObj:PID>356</ProcessObj:PID>
            </cybox:Properties>
          </cybox:Associated_Object>
        </cybox:Associated_Objects>
      </cybox:Action>
    </cybox:Actions>
  </cybox:Event>
</cybox:Observable>

```

図 3.16 通信試行(NETWORKV6)の CybOX 記述例



以上の結果からプロセスログを記述する CybOX の構造は図 3.17 のようになる。背景色がピンクの箇所は CybOX Schema から語彙を選択して記述する場所で、青字がプロセスログの各項目の対応場所である。また、プロセスログから CybOX の各項目へ機械的に挿入し XML 文書で出力するプログラムを作成した。

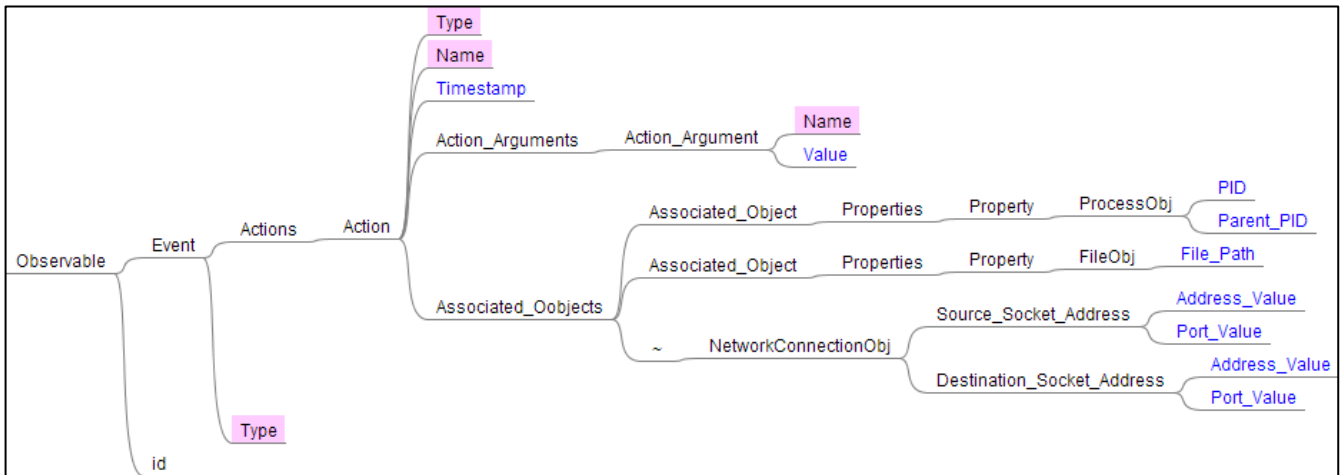


図 3.17 プロセスログ記述用 CybOX の概要

ほかにも、動的解析ソフトから得られた情報を CybOX で記述した結果、図 3.18 のよう作成できた。これは動的解析ソフトから得られた API Call のログを記録している。CSV 形式でログは記録され、その項目は記録時間、PID、ファイル名、API call, Status, Return Value, Prameter1, Prameter2, ... となっている。図からわかるように、このログもプロセスログと同様に CybOX で記述可能であった。

```
<cybox:Observable id="example:Observable-f080e022-421b-479d-b357-506a7eb53402">
  <cybox:Event>
    <cybox:Actions>
      <cybox:Action timestamp="2011-11-29T05:16:22.317000" action_status="SUCCESS">
        <cybox:Action_Arguments>
          <cybox:Action_Argument>
            <cybox:Argument_Name xsi:type="cyboxVocabs:ActionArgumentNameVocab-1.0">API</cybox:Argument_Name>
            <cybox:Argument_Value>lpClassName-&gt;AdobeReaderSpeedLaunchCmdWnd</cybox:Argument_Value>
          </cybox:Action_Argument>
          <cybox:Action_Argument>
            <cybox:Argument_Name xsi:type="cyboxVocabs:ActionArgumentNameVocab-1.0">API</cybox:Argument_Name>
            <cybox:Argument_Value>lpWindowName-&gt;(null)</cybox:Argument_Value>
          </cybox:Action_Argument>
        </cybox:Action_Arguments>
        <cybox:Associated_Objects>
          <cybox:Associated_Object id="example:Process-4b0fc48c-12b4-47c6-820e-9ddbfeff4c360">
            <cybox:Properties xsi:type="ProcessObj:ProcessObjectType">
              <ProcessObj:PID>1376</ProcessObj:PID>
              <ProcessObj:Image_Info>
                <ProcessObj:File_Name>AcroRd32.exe</ProcessObj:File_Name>
              </ProcessObj:Image_Info>
            </cybox:Properties>
          </cybox:Associated_Object>
          <cybox:Associated_Object id="example:API-8421f05b-466c-45de-9429-331d32d7f38d">
            <cybox:Properties xsi:type="APIObj:APIObjectType">
              <APIObj:Function_Name>FindWindowW</APIObj:Function_Name>
            </cybox:Properties>
          </cybox:Associated_Object>
        </cybox:Associated_Objects>
      </cybox:Action>
    </cybox:Actions>
  </cybox:Event>
</cybox:Observable>
```

図 3.18 動的解析結果の CybOX 記述

以上のように、基本的な項目は CybOX の構造を適切に与えることでそのまま表現することができた。しかし、プロトコル番号は記述できる箇所がなかった。また、コマンドラインは記述に問題はなかったが、1つの文字列内に実行ファイルとそれに渡す引数というように異なる意味を持つ情報が一緒に記述されているため、解析時には専門家の知識が必要になると懸念される。

### 3.4 プロセスログを用いたマルウェア動的解析

対象としたマルウェアは **ShinoBOT** と呼ばれる **RAT** シミュレータである[33]. **ShinoBOT.exe** の実行時に起動する画面を図 3.19 に示す. **ShinoBOT** を実行すると, 図 3.20 のように自動的に **PC** のネットワーク構成やシステム情報などが収集され **ShinoBOT** サーバに送信される. また, **ShinoBOT** サーバにアクセスすると **ShinoBOT** を実行した端末の情報を閲覧し, 端末に対して自由にコマンドを実行させることができる. ただし, 情報の閲覧とコマンドの実行には **ShinoBOT** 実行時にランダムに生成されるパスワードが必要である. そのため, 他のユーザによって端末の情報閲覧やコマンド実行がなされるおそれがない. そのため安全に **RAT** の解析が可能である.

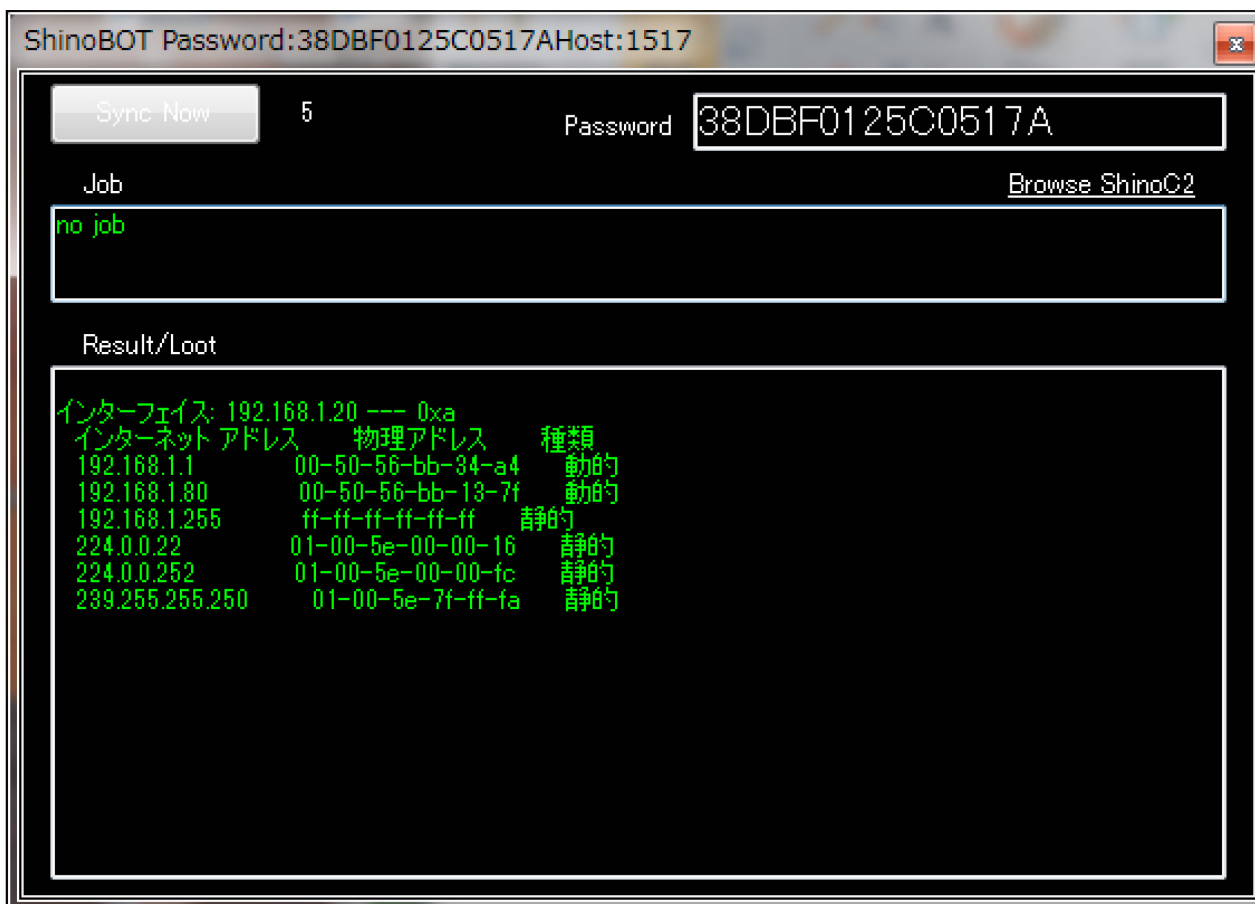
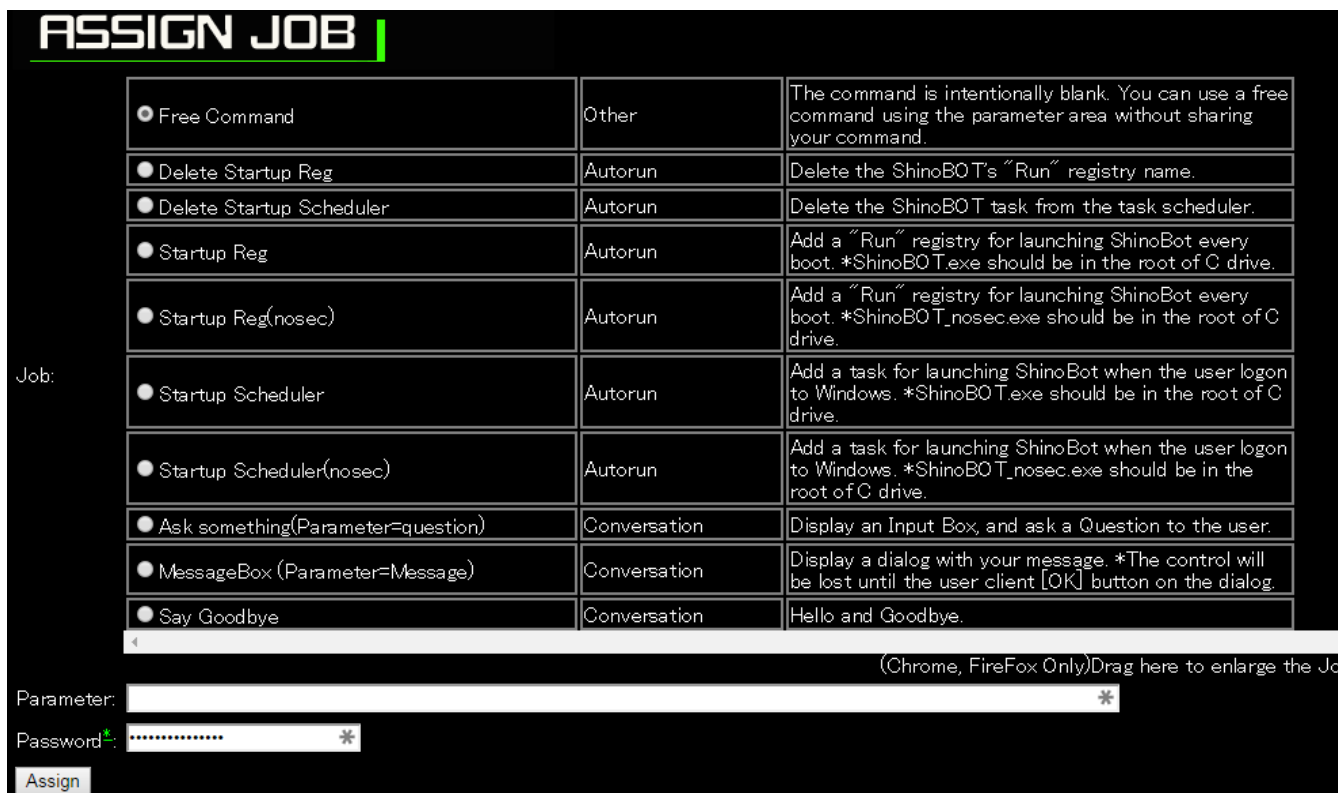


図 3.19 ShinoBOT.exe の起動画面



7938	Screen Shot (Default)	SBOTshot	Done	Screen_Shot
7937	Systeminfo (Default)	systeminfo	Done	<pre> ホスト名: K-W7X64A OS 名: Microsoft Windows 7 Professional OS バージョン: 6.1.7601 Service Pack 1 ビルド 7601 OS 製造元: Microsoft Corporation OS 構成: スタンドアロン ワークステーション </pre>
7936	Systeminfo (Default)	systeminfo	Done	<pre> ホスト名: K-W7X64A OS 名: Microsoft Windows 7 Professional OS バージョン: 6.1.7601 Service Pack 1 ビルド 7601 OS 製造元: Microsoft Corporation OS 構成: スタンドアロン ワークステーション </pre>
7935	Download PsExec	SBOTwget:http://shinoc2.shinosec.cloudns.org/files/PsExec.exe	Done	File Downloaded Saved Path: C:\Users\Yui\Downloads\PsExec.exe
7934	Download PsExec	SBOTwget:http://shinoc2.shinosec.cloudns.org/files/PsExec.exe	Done	File Downloaded Saved Path: C:\Users\Yui\Downloads\PsExec.exe
7933	Free Command		Done	
7932	Find Neighborhood from ARP (Default)	arp -a	Done	<pre> インターフェイス: 192.168.21.26 --- 0xa インターネット アドレス 物理アドレス 種類 192.168.21.1 00-a0-d6-b6-4b-ec 動的 192.168.21.20 00-50-56-bb-43-ce 動的 192.168.21.28 00-50-56-bb-9c-ef 動的 </pre>
7931	Show Local User List (Default)	net user	Done	<pre> ¥¥K-W7X64A のユーザー アカウント ----- Administrator      Guest                Yui </pre>
7930	Task List (Default)	tasklist /svc	Done	<pre> イメージ名 PID サービス ----- System Idle Process 0 M/A </pre>

図 3.20 ShinoBOT のリモートコード実行用画面

ShinoBOT 実行時のプロセスログを Onmitsu で収集した。ShinoBOT.exe を起動した瞬間に記録されたプロセスログの一部を図 3.21 に示す。赤字がプロセスを起動したファイルを，青字が入力されたコマンドラインを示している。この図から，ShinoBOT が実行されると，まず SerchProtocolHost が実行され，ShinoBOT が必要としている情報を収集する。次に，ShinoBOT のプロセスが netsh を実行してファイアーウォールを停止させ，net を実行して WindowsUpdateService を停止させる。このようにリモートアクセスができるように準備を進めていることがわかる。

```
5128,3672,¥??¥C:¥Users¥Yui¥Downloads¥ShinoBOT.exe,"C:¥Users¥Yui¥Downloads¥ShinoBOT.exe" ,,,,,
6064,2620,¥??¥C:¥Windows¥system32¥SearchProtocolHost.exe,"C:¥Windows¥system32¥SearchProtocolHost.exe" Global¥UsGthrFltPipeMssGthrPipe99_
Global¥UsGthrCtrlFltPipeMssGthrPipe99 1 -2147483646
"Software¥Microsoft¥Windows Search" "Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT; MS Search 4.0 Robot)"
"C:¥ProgramData¥Microsoft¥Search¥Data¥Temp¥usgthrsvc"
"DownLevelDaemon" ,,,,,
1444,5128,¥??¥C:¥Windows¥SysWOW64¥netsh.exe,netsh firewall set opmode
mode=disable,,,,,
4184,5128,¥??¥C:¥Windows¥SysWOW64¥net.exe,net stop wuau serv,,,,,
```

図 3.21 ShinoBOT 起動時のプロセスログの一部

上記のようにプロセスログからプロセス起動時の情報を抽出して ShinoBOT の動作を解析した結果、以下の動作手順でリモートアクセスをしているとわかった。

- 1 SearchProtocolHost.exe で必要なファイルを検索
- 2 ファイアウォール等のサービスを停止
- 3 レジストリキーの追加
- 4 システム情報やプロセスリスト、IP アドレスなどの情報を取得
- 5 DllHost.exe を実行して任意の.dll ファイルをロード
- 6 ShinoBOT サーバとの通信開始

プロセスログによる動的解析の結果、このマルウェアは通常は実行されない異常なコマンドを実行していることがわかった。そこで、異常なコマンド実行をプロセスログから発見できればマルウェア攻撃を検知できると考えられる。

## 3.5 プロセスログを用いたマルウェア攻撃検知手法

### 3.5.1 提案手法

CybOX に変換したプロセスログを用いてマルウェア攻撃を検知する手法を検討した。検知手法は上記の結果から得られたコマンドラインのブラックリストとプロセスログを比較する方法である。コマンドラインのブラックリストはマルウェア実行時のプロセスログから特異なコマンドラインを取捨選択して作成した。今回は異常なコマンドラインとして、ShinoBOT の解析から以下の3つコマンドラインをブラックリストとして組み込んだ。

- `netsh firewall set opmode mode=disable` : ファイアーウォールの無効化
- `net stop wuauclt` : WindowsUpdate サービスの停止
- `net stop McShield` : McShield(McAfee)の停止

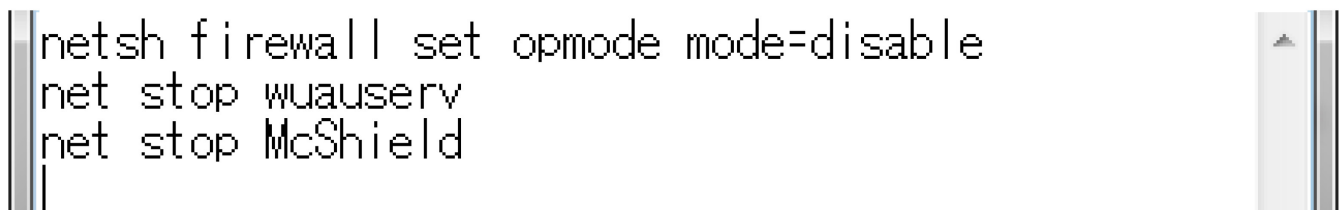
ブラックリストとプロセスログのマッチング手法はそれぞれ1つずつ取り出して比較する手法である。具体的な手順を以下に示す。このとき、プロセスログはCybOXに変換しているものとする。

- 1 CybOX から `Observable` を1つずつ読み込む
- 2 `ActionName` が `OpenProcess` ならば、コマンドラインの情報を取得
- 3 ブラックリスト内の情報を1行ずつ読み込む
- 4 コマンドラインとブラックリストの文字列比較
- 5 一致したなら、アラートを出す

このような手順で提案手法を実際に処理するプログラムを作成した。

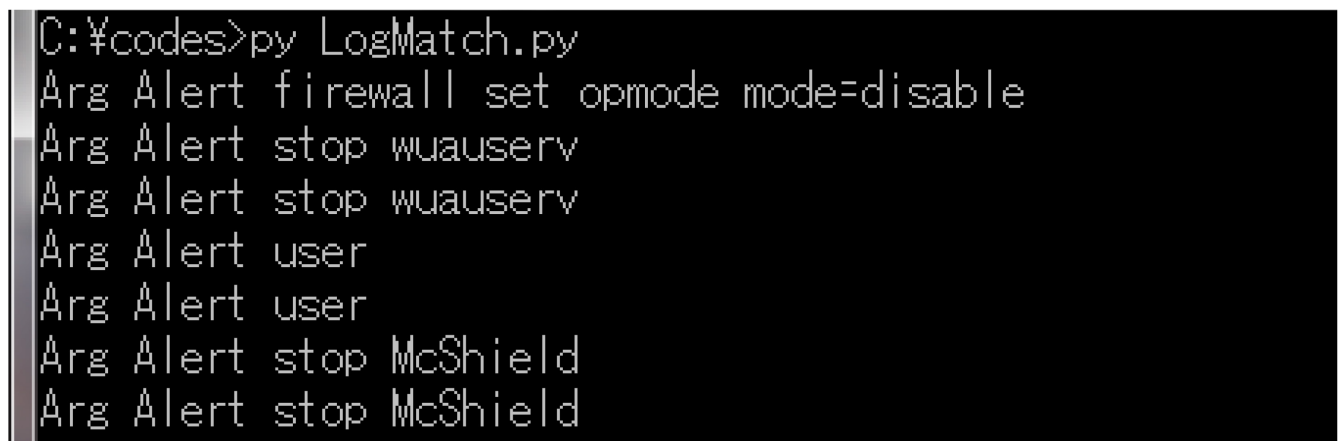
### 3.5.2 検証実験

作成したプログラムの検証実験を行う。ShinoBOT のプロセスログを対象とし、適切にアラートが表示されるか評価する。コマンドラインブラックリストを図 3.22 に示す。ブラックリストは前項での検討結果と同じである。プログラムの実行結果を図 3.23 に示す。この図からブラックリスト内のコマンドはすべて検知していることがわかる。ただし、**user** のように部分一致しているコマンドも検知している。



```
netsh firewall set opmode mode=disable
net stop wuauerv
net stop McShield
```

図 3.22 作成したコマンドラインブラックリスト



```
C:\¥codes>py LogMatch.py
Arg Alert firewall set opmode mode=disable
Arg Alert stop wuauerv
Arg Alert stop wuauerv
Arg Alert user
Arg Alert user
Arg Alert stop McShield
Arg Alert stop McShield
```

図 3.23 プロセスログとブラックリストとの比較結果



### 3.5.3 考察

前項の結果からプロセスログを用いたマルウェア攻撃検知の実現可能性が得られた。ただし、マルウェアの特定は困難であり、この手法だけではアラートを出した後の対策には結びつかない。その理由は特定のマルウェア類「だけ」が使うコマンドライン入力のごくごく少数だと考えられるためである。ただし、ファイアーウォールの停止など一部のコマンドラインはユーザにアラートを表示した後にユーザにサービスの起動を促すなどの対策は可能だと考える。

これらの結果からプロセスログは攻撃検知のシグネチャとして有用であると考えられる。ブラックリストとの比較はプロセスログのみでも可能である。そのため、マルウェア検知のトリガーとしての利用も期待される。

### 3.6 プロセスパターンを用いたマルウェア検知手法

1つのプロセスログから特異なコマンドを発見することは専門家の知識がなければ困難である。その理由はコマンドライン自体の意味解釈が困難で、また同時にマルウェアが実行したものかも判断する必要があるためである。プロセスログを表記したCybOXは共有が容易である。そこで、図3.24のようにCybOXで表記したマルウェアのプロセスログを何度も比較していけば、マルウェア独自のプロセスの起動順序(プロセスパターン)が作成できるのではと考えた。そこで、次にプロセスパターンによるマルウェア検知を検討した。

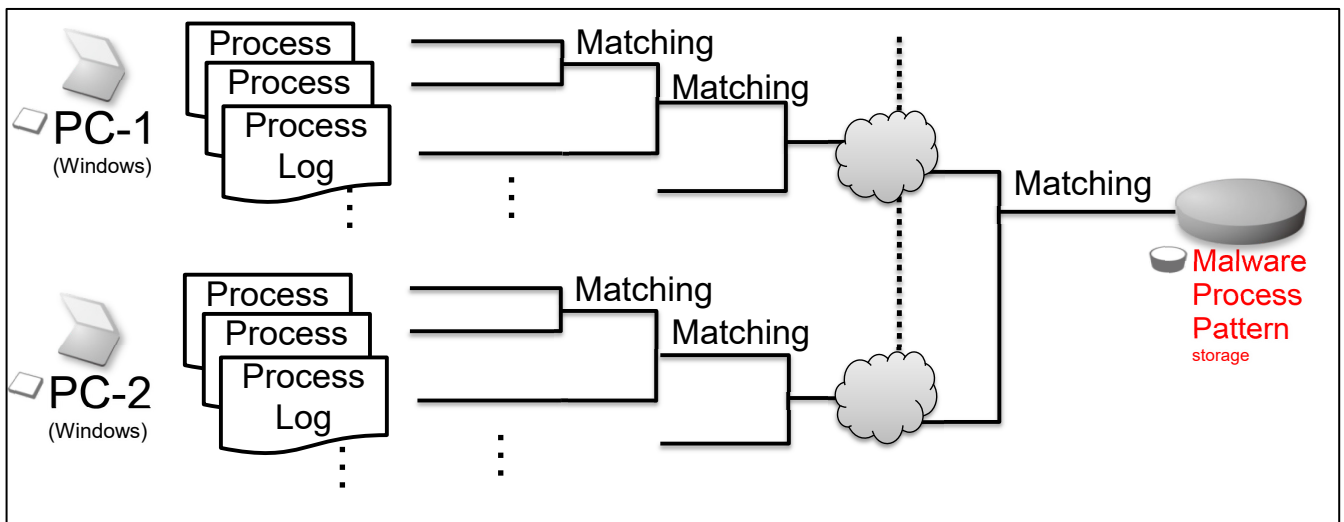


図 3.24 マルウェアプロセスパターン作成手法の概要

### 3.6.1 マルウェアプロセスパターンの作成

CybOX で記述したマルウェアのプロセスログを繰り返し比較してマルウェア独自のプロセスパターンを作成する。今回はプロセスの起動順序に着目した。具体的には複数のプロセスログを比較した際に一致しないプロセス起動を除去していく方法である。比較対象はファイルパスを採用した。プロセス ID やコマンドラインで比較しない理由は、プロセス ID やコマンドラインの引数は実行する端末によって異なる場合が多いためである。プロセスパターン作成手順を以下に示す。

- 1 それぞれの CybOX からプロセス起動情報を持つ **Observable** を取得
- 2 **Observable** からコマンドラインとファイルパスを取得
- 3 ファイルパス同士を比較し、一致したプロセスログを表示
- 4 CybOX から不一致の **Observable** を除去

手順 1~3 までを機械的に行うプログラムを作成した。

次に、作成したプログラムを使って **ShinoBOT** のプロセスパターンを作成した。プロセスパターンの作成環境は **Windows7** で、**WindowsUpdate** 適用済みと未適用の環境である。プロセスログは各環境で時間をずらして 3 つずつ作成した。プロセスの記録は 3.4 説で記述した動作手順の内 1~5 まで、**ShinoBOT** の実行から **DllHost.exe** の実行までを対象とした。図 3.25 に手順 3 で得られた結果の一部を示す。「**Log :**」の行は比較するプロセスログを、「**Mal :**」の行はマルウェアプロセスパターンを示している。この図から表示されたプロセスログは **ShinoBOT** のプロセスパターンになり得ることがわかる。プロセスパターンとして 12 のプロセス起動ログが得られた。

```

Log: ['C:\\Windows\\system32\\SearchProtocolHost.exe Global\\UsGthrFltPipeMssGthrPipe167_Global\\UsGthrCtrlFltPip
Mal: ['C:\\Windows\\system32\\SearchProtocolHost.exe Global\\UsGthrFltPipeMssGthrPipe99_Global\\UsGthrCtrlFltPip
Log: ['netsh firewall set opmode mode=disable'] [{'parent_pid': 6088, 'start_time': '2015-02-19T19:03:15.030300'}
Mal: ['netsh firewall set opmode mode=disable'] [{'parent_pid': 5128, 'start_time': '1900-01-29T10:13:05.046500'}
Log: ['net stop wuauclt'] [{'parent_pid': 6088, 'start_time': '2015-02-19T19:03:15.031100', 'pid': 5840, 'xsi:ty
Mal: ['net stop wuauclt'] [{'parent_pid': 5128, 'start_time': '1900-01-29T10:13:05.046500', 'pid': 4184, 'xsi:ty
Log: ['\\??\\C:\\Windows\\system32\\conhost.exe "-1115788352279676201778129907-120947242-127227860510493849931121
Mal: ['\\??\\C:\\Windows\\system32\\conhost.exe "-6196549262128730366-1638676944-11783615594806282531237689228-49
Log: ['net stop McShield'] [{'parent_pid': 6088, 'start_time': '2015-02-19T19:03:15.032200', 'pid': 5272, 'xsi:ty
Mal: ['net stop McShield'] [{'parent_pid': 5128, 'start_time': '1900-01-29T10:13:05.048100', 'pid': 5296, 'xsi:ty
Log: ['\\??\\C:\\Windows\\system32\\conhost.exe "-67636407233439699-177585641017181257861560794863-11427643603609
Mal: ['\\??\\C:\\Windows\\system32\\conhost.exe "-419521295-1156185441161548583138362471462075811-1223113396988589

```

図 3.25 マルウェア起動プロセスログ同士の比較結果の一部

### 3.6.2 提案手法

前項で作成したプロセスパターンを使用したマルウェア検知手法を検討した。検知手法は 5.3 節で述べたアプローチと同様にプロセスパターンのマッチングを行う。マッチングは標準である **CybOX** を用いて行うので、外部で作成されたプロセスログとの比較も容易である。

検知手順を以下に示す

- 1 検知対象となる端末のプロセスログを表記した **CybOX** からプロセス起動情報を持つ **Observable** を取得
- 2 取得した **Observable** からコマンドラインとファイルパスを取得
- 3 マルウェアプロセスパターンの **Observable** からコマンドラインとファイルパスを取得
- 4 2 で取得したファイルパスとマルウェアプロセスパターンのファイルパスを比較
- 5 比較結果が
  - 5.1 一致したなら、マルウェアプロセスパターンから次の **Observable** を取得
  - 5.2 一致しなければ、**CybOX** から次の **Observable** を取得

- 6 手順 2~5 を繰り返し行い，一致した割合がしきい値を超えたらアラートを表示

手順 1~5 を機械的に行うためのプログラムを作成した。

### 3.7 実験

提案手法の有効性検証するために，端末上のプロセスログと複数のマルウェアプロセスパターンを比較して **ShinoBOT** が特定可能か実験する．各マルウェアプロセスパターンと一致したプロセスの割合を比較することで評価を行う．

#### 3.7.1 実験環境

実験を行う環境は **VMWare vSphere** 上に仮想ネットワーク環境を構築して行った．システム構成を図 6.1 に示す．実験には 3 台の端末を使用した．それぞれ，マルウェアを実行する被攻撃 **PC**，検知を行う解析 **PC**，ネットワークの監視を行う **IDSPC** である．被攻撃 **PC** はマルウェアの実行とプロセスログの記録を行う．解析 **PC** は被攻撃 **PC** のプロセスログを **CybOX** に変換し，提案手法でマルウェアの特定を行う．**IDSPC** では不正通信の検知を行って被攻撃 **PC** から解析 **PC** へプロセスログを送信する際のトリガーを作る．

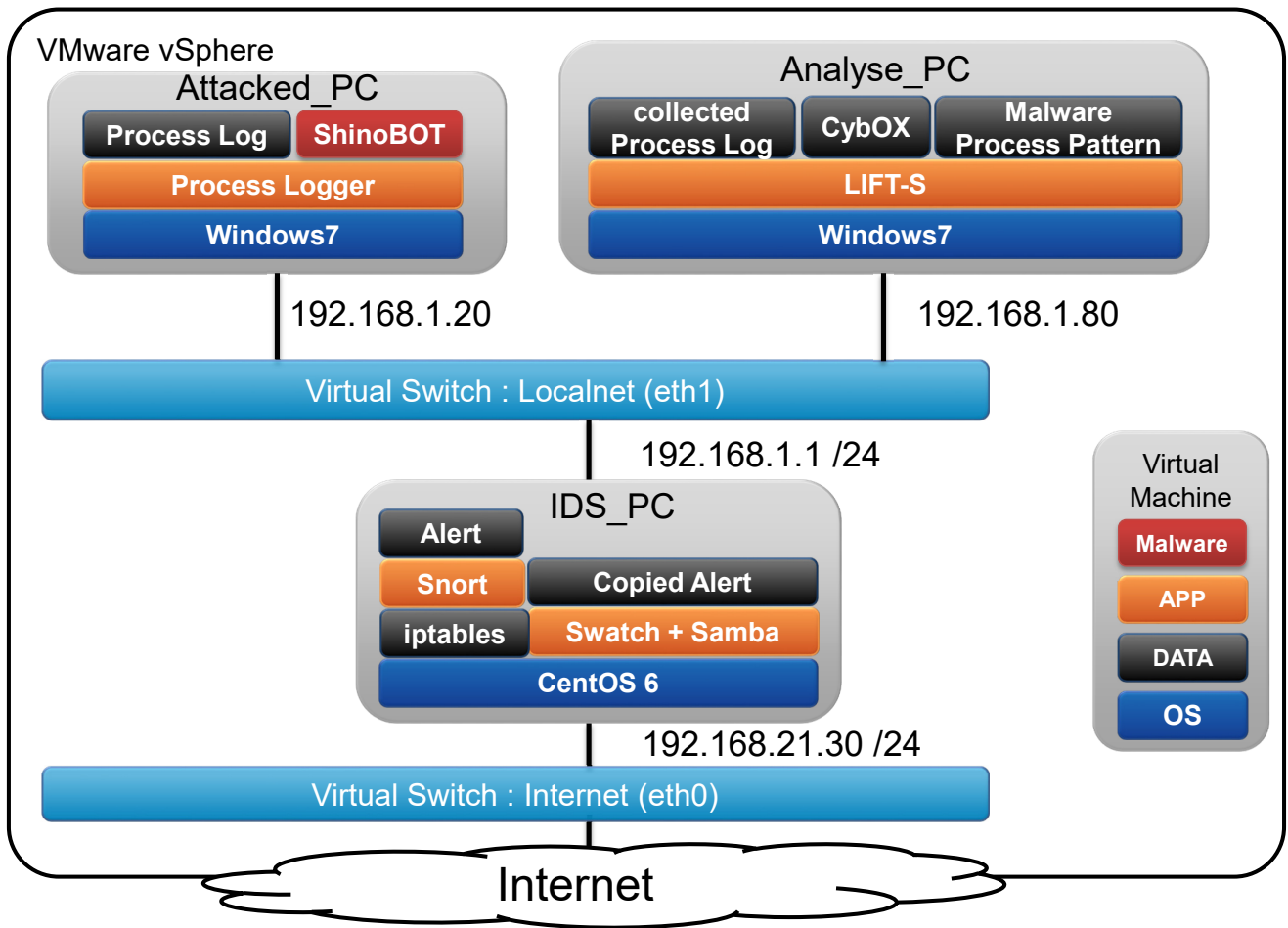


図 6.1 実験用システム構成概要

### 3.7.2 事前準備：比較用マルウェアプロセスパターンの作成

比較用のマルウェアプロセスパターン作成する。マルウェアは ViruseShare から入手した。プロセスパターンの作成手順は 3.6.1 項と同様である。作成した結果は以下のようになった。

0cf9e999c574ec89595263446978dc9f : プロセス起動ログは 4

001dd76872d80801692ff942308c64e6 : プロセス起動ログの数は 9

これらのマルウェアの解析結果は、サンドボックスでマルウェアを解析した結果を保存しているサイトである **Malwr** から取得した。その解析結果によると、どちらのマルウェアもスタートアップフォルダにマルウェアをコピーし、個人情報をインターネットブラウザから盗む働きをする。なお、実験用マルウェアプロセスパターンの作成は被攻撃 PC 上で行った。

### 3.7.3 実験手順

実験概要を図 3.26 に示す。実験手順を以下に示す。

- 1 被害 PC で **ShinoBOT.exe** を実行
- 2 **Snort** が **ShinoBOT** サーバとの通信を検知
- 3 **Snort** が不正通信アラートを解析 PC へ通知
- 4 通知を受けた解析 PC が被害 PC にプロセスログを送信するよう要求送信
- 5 要求を受けた被害 PC が解析 PC へプロセスログを送信
- 6 解析 PC がプロセスログを **CybOX** 文書に変換
- 7 解析 PC が **CybOX** 文書とマルウェアプロセスパターンを比較

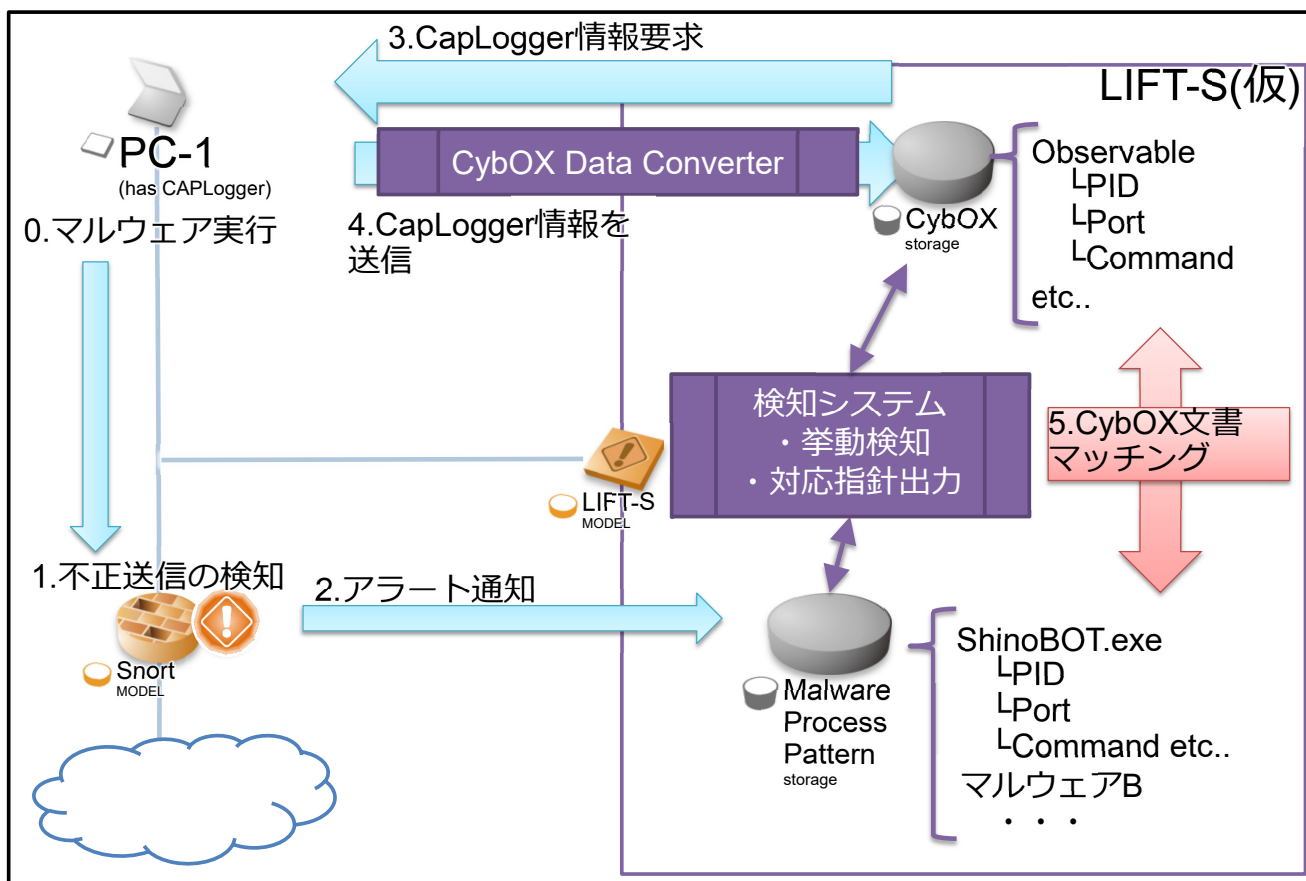


図 3.26 実験手順の概要

手順 1 の結果は被害 PC の表示画面から確認する。手順 2~5 は各 PC のログから結果を確認する。手順 6 は手順 5 で得られたプロセスログと変換した CybOX を比較して確認する。手順 7 は実行結果から確認する。

手順 7 の結果から提案手法の有効性を評価する。また、手順 1~7 の結果から提案手法を実フィールドへ展開した際の実現可能性を評価する。



### 3.7.4 実験結果

ここでは特に、手順 6 と手順 7 の結果を記述する。手順 6 の結果を図 3.27 と図 3.28 に示す。図 3.27 は被害 PC から取得したプロセスログの内容を、図 3.28 は CybOX に変換した結果を示している。図 3.27 の 1 行目と図 3.28 の Observable を比較することでプロセスログが正しく CybOX に変換されたとわかる。

```
2015,02,26,14,24,04,0395,PROCESS_LAUNCH,4664,2068,??C:\Users\Yui\Downloads
¥ShinoBOT.exe,"C:\Users\Yui\Downloads¥ShinoBOT.exe" ,,,,,
2015,02,26,14,24,04,0395,PROCESS_MODLOAD,2068,,¥Windows¥System32¥sfc.dll,,,,,,
2015,02,26,14,24,04,0395,PROCESS_MODLOAD,2068,,¥Windows¥System32¥sfc_os.dll,,,,,,
2015,02,26,14,24,04,0411,PROCESS_MODLOAD,4664,,¥SystemRoot¥System32¥ntdll.dll,,,,,,
2015,02,26,14,24,04,0411,PROCESS_MODLOAD,4664,,¥SystemRoot¥SysWOW64¥ntdll.dll,,,,,
2015,02,26,14,24,04,0411,PROCESS_MODLOAD,4664,,¥Device¥HarddiskVolume2¥Users¥Yui
¥Downloads¥ShinoBOT.exe,,,,,,
```

図 3.27 記録したプロセスログの一部

```
xmlns:AddressObj="http://cybox.mitre.org/objects#AddressObject-2"
xmlns:cyboxVocabs="http://cybox.mitre.org/default_vocabularies-2" xmlns:cybox="http://cybox.mitre.org/cybox-2"
xmlns:cyboxCommon="http://cybox.mitre.org/common-2" xmlns:example="http://LIFT-S.com/"
- <cybox:Observable id="example:Observable-e5446cbc-339c-4751-87fa-d6309e67af38">
- <cybox:Event>
  <cybox:Type xsi:type="cyboxVocabs:EventTypeVocab-1.0.1">Process Mgt</cybox:Type>
- <cybox:Actions>
  - <cybox:Action timestamp="2015-02-26T14:24:04.039500">
    <cybox:Type xsi:type="cyboxVocabs:ActionTypeVocab-1.0">Start</cybox:Type>
    <cybox:Name xsi:type="cyboxVocabs:ActionNameVocab-1.1">Open Process</cybox:Name>
  - <cybox:Action_Arguments>
    - <cybox:Action_Argument>
      <cybox:Argument_Name
        xsi:type="cyboxVocabs:ActionArgumentNameVocab-1.0">Command</cybox:Argument_Name>
      <cybox:Argument_Value>C:\Users\Yui\Downloads\ShinoBOT.exe </cybox:Argument_Value>
    </cybox:Action_Argument>
  </cybox:Action_Arguments>
- <cybox:Associated_Objects>
  - <cybox:Associated_Object id="example:Process-c95fe0e8-f333-4aad-846f-f3bb50533d40">
    - <cybox:Properties xsi:type="ProcessObj:ProcessObjectType">
      <ProcessObj:PID>4664</ProcessObj:PID>
      <ProcessObj:Parent_PID>2068</ProcessObj:Parent_PID>
      <ProcessObj:Start_Time>2015-02-26T14:24:04.039500</ProcessObj:Start_Time>
    </cybox:Properties>
  </cybox:Associated_Object>
  - <cybox:Associated_Object id="example:File-fa0e1289-56e3-44ed-8a11-55b7f00feab8">
    - <cybox:Properties xsi:type="FileObj:FileObjectType">
      <FileObj:File_Path>??C:\Users\Yui\Downloads\ShinoBOT.exe</FileObj:File_Path>
    </cybox:Properties>
  </cybox:Associated_Object>
</cybox:Associated_Objects>
```

図 3.28 CybOX で記述したプロセスログの一部

手順 7 の結果を図 3.29 に示す。パターン一致数とは被害 PC のプロセスログと各マルウェアを比較した際に一致したプロセスログの数である。マルウェア\_プロセス数とは、比較しているマルウェアプロセスパターン内のプロセスログの数である。ログ\_プロセス数とは被害 PC のプロセスログの総数である。パターン一致割合とはパターン一致数とマルウェア\_プロセス数の割合である。図からわかるように ShinoBOT のパターン一致割合は 100%。0cf9e999c574ec89595263446978dc9f のパターン一致割合は 0.0%。001dd76872d80801692ff942308c64e6 のパターン一致割合は 44.4%であった。

```
>>> ===== RESTART =====
>>>
----- VirusShare_0cf9e999c574ec89595263446978dc9f.xml -----
パターン一致数: 0
マルウェア_プロセス数: 4
ログ_プロセス数: 125
パターン一致割合: 0.0
----- shinoBOT.xml -----
パターン一致数: 12
マルウェア_プロセス数: 12
ログ_プロセス数: 125
パターン一致割合: 1.0
----- VirusShare_001dd76872d80801692ff942308c64e6.xml -----
パターン一致数: 4
マルウェア_プロセス数: 9
ログ_プロセス数: 125
パターン一致割合: 0.4444444444444444
>>> |
```

図 3.29 CybOX 文書とマルウェアプロセスパターンとの比較結果

### 3.7.5 考察

手順 7 の結果からわかるように被害 PC のプロセスログと ShinoBOT のマルウェアプロセスパターンの一致した割合は 100%であった。また、他のマルウェアプロセスパターンと一致した割合は高々 44.4%であった。したがって、適切なしきい値を設定することでプロセスパターンによるマルウェアの特定は可能であると考えられる。この考察と手順 1~6 の結果から、提案手法をシステムとして実フィールドでの有効性を示す結果であると考えられる。

実験後に 001dd76872d80801692ff942308c64e6 と一致したプロセスログを表示させた結果を図 3.30 に示す。この図から DllHost.exe が連続して起動していることがわかる。DllHost.exe は Windows の.dll ファイルをロードするための基本的な実行ファイルである。そのため、このコマンドラインだけでは、このあとに.dll ファイルがロードされることしか判断できない。したがって、検知精度をさらに高めるには DllHost.exe を起動した後の Process\_ModLoad のログまで考慮する必要がある。また、その際には DllHost.exe が起動した.dll ファイルを特定するためにプロセスツリーの情報も必要となる。

```
----- VirusShare_001dd76872d80801692ff942308c64e6.xml -----
Log[ 14 ]: ['C:\\Windows\\system32\\DllHost.exe /Processid:{AB8902B4-09CA-4BB6-B78D-A8F59079A8D5}']
Mal[ 0 ]: ['C:\\Windows\\system32\\DllHost.exe /Processid:{F9717507-6651-4EDB-BFF7-AE615179BCCF}']
Log[ 15 ]: ['C:\\Windows\\system32\\DllHost.exe /Processid:{AB8902B4-09CA-4BB6-B78D-A8F59079A8D5}']
Mal[ 1 ]: ['C:\\Windows\\system32\\DllHost.exe /Processid:{F9717507-6651-4EDB-BFF7-AE615179BCCF}']
Log[ 16 ]: ['C:\\Windows\\system32\\DllHost.exe /Processid:{AB8902B4-09CA-4BB6-B78D-A8F59079A8D5}']
Mal[ 2 ]: ['C:\\Windows\\system32\\DllHost.exe /Processid:{F9717507-6651-4EDB-BFF7-AE615179BCCF}']
Log[ 17 ]: ['C:\\Windows\\system32\\DllHost.exe /Processid:{AB8902B4-09CA-4BB6-B78D-A8F59079A8D5}']
Mal[ 3 ]: ['C:\\Windows\\system32\\DllHost.exe /Processid:{AB8902B4-09CA-4BB6-B78D-A8F59079A8D5}']
パターン一致数: 4
マルウェアプロセス数: 9
ログプロセス数: 125
パターン一致割合: 0.444444444444
```

図 3.30 001dd76872d80801692ff942308c64e6 のプロセスパターンとの比較結果

## 第4章 侵入検知ツールの開発

本章は拙著「マルウェアによるネットワーク内の挙動を利用した標的型攻撃における感染経路検知ツールの開発と評価」[34]によるものである。

IPA の報告書[1]によると、標的型メール攻撃の典型的な手法は、偽装メールと不正プログラム(マルウェア)を組み合わせて行う方法である。この攻撃の実施は、次の6段階で定義されている。

- (1)計画立案段階：攻撃対象を決定し情報を収集
- (2)攻撃準備段階：偽装メールやC&Cサーバを用意
- (3)初期侵入段階：偽装メールによるマルウェア感染
- (4)基盤構築段階：感染端末の情報を窃取し環境を調査
- (5)内部侵入・調査段階：端末間での侵害を拡大
- (6)目的遂行段階：窃取した情報を外部へ送信

攻撃者は段階(3)で特定のユーザに対して特定の行為をとるよう誘導することで侵入し、有用な情報の窃取するために段階(6)を目指す。

現在の標的型メール攻撃の対策は主に入口対策、内部対策と呼ばれるものである。入口対策とは段階(3)で攻撃の侵入を防止する対策である。また、内部対策とは段階(3)～段階(5)において、攻撃者による侵入拡大を防止する対策である。

このとき、入口対策や内部対策において異常検出された際に、その異常が標的型メール攻撃に起因する否かを判断するのは困難である。さらに、検知された端末が段階(3)で感染した端末か、段階(5)で感染した端末か判断する必要がある。なぜなら、段階(5)で感染した端末なら、その感染元の端末が存在するため、感染元端末からさらに感染が拡大していく恐れがあるためである。したがって、感染経路を調査しその感染源を迅速に発見する必要がある。

感染経路の追跡は、従来は専門家のチームが不正プログラムや各端末のログを解析することで追跡している。そのためには高度な専門知識が必要であり、また長い時間を要するため感染源の迅速な発見が困難であった。

このとき、感染端末を特定後に、端末内のログを組み合わせることでその原因となる不正プログラムや、その起動原因となった通信を特定する場合、マルウェアがプロセスを隠蔽すると、マルウェアの起動原因となる通信を特定することが困難になる恐れがある。その際に、端末内のプロセスとそれによる通信を同一のログで取得できれば、端末内のログを照会する手間が減り感染経路の把握が迅速にできる。

これまでネットワークトラフィック等を用いることで感染端末を特定する手法は存在している。しかし、内部侵入・調査段階に焦点をあて、その攻撃の原因となるプロセスもあわせて感染経路を追跡する手法は、著者らの調査した範囲では存在しない。

そこで、本研究では内部侵入・調査段階に焦点をあて、複数の端末のプロセスログを用いることで感染経路を追跡する手法を提案する。このとき、プロセスログの記録には後述する Onmitsu というツールを用いた。そして、この特徴を用いた感染経路検知ツールを開発した。処理時間を短縮するためのアプローチとして、プロセスログの情報を適切に抽出する手法をとる。そこで、情報を抽出し、様々な粒度で表現することが容易であるオントロジーを採用した。

## 4.1 感染経路検知ツールの開発

### 4.1.1 標的型攻撃における内部侵入・調査段階の挙動

本節では内部侵入・調査段階の挙動を IPA[2]と FireEye[35]の報告書をもとに説明する。この段階での目的はネットワーク内の認証情報を窃取しながら侵害範囲を拡大することである。

具体的には、次のような手順で侵入を拡大する。

- (1)初期感染した端末で管理者権限を窃取
- (2)FTP サービスなどを使用して、近隣端末へ内部攻撃用ツールを転送
- (3)転送したツールをリモート実行し他端末へ侵入拡大
- (4)イベントログなどの攻撃痕跡を消去

端末間で侵害を拡大する際には、基盤拡大用端末や情報収集用端末、情報送信用端末など役割を各端末にもたせている。役割を分散させることで攻撃の全容を把握しづらくさせていると考えられている。

FireEye の報告書によると、ネットワーク内の各端末は同一のローカル管理者パスワードを持っていたため、手順(2)が容易となった。また、その際に攻撃者は転送ツールとして PsExec を使用していることがわかっている。PsExec とは Microsoft が提供しているソフトウェアである[36]。あわせて、JPCERT によると、攻撃者は内部侵入・調査段階において Windows コマンドを多用していることがわかっている[37]。特に、at コマンドや wmic コマンドがリモート端末上でマルウェアを実行するために利用されている。

以上のことから、本論文ではまず侵入拡大時の活動に焦点をあてるため、手順(3)までを主な対象とする。また、侵害拡大に使用するツールとして PsExec と at コマンド、wmic コマンドを用いることとした。

### 4.1.2 内部通信時の挙動解析

本論文で提案する感染経路検知手法のアプローチは内部通信とその通信を行っているプロセスの関係性を明確化することで他端末侵入の挙動を追跡することである。そこで、本項では PsExec, at コマンド、wmic コマンドの内部通信時の特徴的な挙動をプロセスログから抽出し、関係性を明確化する。このとき、抽出した特徴的な挙動の正否は、WireShark と Process Monitor のログとプロセスログを比較することで判断した。その結果、内部ネットワーク通信時のクライアント端末とリモート端末のそれぞれの特徴を表 4.1 に記述する。表 4.1 から、内部通信時には必ず特徴的なプロセス、ポート番号を用いられると考えられる。

表 4.1 内部通信時の特徴

ツール	クライアント端末		リモート端末	
	起動プロセス	通信試行時の宛先ポート番号	直前の通信試行時の実行プロセス	親プロセス
PsExec	psexec	135	RpcSs に属する svchost	PSEXECsvc.exe
at	at	445	Schedule に属する svchost	taskeng.exe
wmic	wmic	135	Schedule に属する svchost	wmicprvse.exe

```

2015,08,03,17,33,40,0664,PROCESS_LAUNCH,3084,2108,
¥??¥C:¥Users¥Yui¥Downloads¥PsExec.exe,"PsExec.exe
-accepteula -d ¥¥k-w7x64a -u Yui -c
-f ""C:¥Users¥Yui¥downloads¥ShinoBOT.exe"",,,,,
2015,08,03,17,33,42,0817,NETWORKV4,348,,,
,192.168.21.32,58986,224.0.0.252,5355,17
2015,08,03,17,33,42,0817,NETWORKV4,348,,,
,192.168.21.32,60921,224.0.0.252,5355,17
2015,08,03,17,33,43,0911,NETWORKV4,4,,,
,192.168.21.32,49451,192.168.21.30,445,6
2015,08,03,17,33,43,0911,NETWORKV4,348,,,
,192.168.21.32,54718,224.0.0.252,5355,17
2015,08,03,17,33,43,0927,NETWORKV4,348,,,
,192.168.21.32,49561,224.0.0.252,5355,17
2015,08,03,17,33,52,0039,NETWORKV4,348,,,
,224.0.0.252,5355,192.168.21.30,63415,17
2015,08,03,17,33,52,0335,NETWORKV4,4,,,
,192.168.21.255,137,192.168.21.30,137,17
2015,08,03,17,34,05,0316,PROCESS_QUIT,3084,,,,,,,,,

```

図 4.1 PsExec を実行した際のクライアント端末のプロセスログ

```

2015,08,03,17,33,43,0812,NETWORKV4,368,, ,↓
,192.168.21.30,5355,192.168.21.32,49561,17↓
2015,08,03,17,33,51,0908,NETWORKV4,816,, ,↓
,192.168.21.30,68,255.255.255.255,67,17↓
2015,08,03,17,33,51,0908,NETWORKV4,368,, ,↓
,192.168.21.30,63415,224.0.0.252,5355,17↓
2015,08,03,17,33,52,0220,NETWORKV4,4,, ,↓
,192.168.21.30,137,192.168.21.255,137,17↓
2015,08,03,17,33,54,0514,NETWORKV4,368,, ,↓
,192.168.21.30,58589,210.130.0.1,53,17↓
2015,08,03,17,33,54,0592,NETWORKV4,1248,, ,↓
,192.168.21.30,49202,111.221.29.253,443,6↓
2015,08,03,17,33,54,0810,NETWORKV4,368,, ,↓
,192.168.21.30,53577,224.0.0.252,5355,17↓
2015,08,03,17,34,04,0841,PROCESS_LAUNCH,3556,536,↓
¥??¥C:¥Windows¥PSEXESVC.exe↓
,C:¥Windows¥PSEXESVC.exe,, , ,↓
2015,08,03,17,34,05,0059,PROCESS_LAUNCH,4092,3556,
¥??¥C:¥Windows¥ShinoBOT.exe,ShinoBOT.exe,, , , ,↓
2015,08,03,17,34,05,0059,PROCESS_QUIT,3556,, , , , ,

```

図 4.2 PsExec を実行した際のリモート端末のプロセスログ

ここで、表 4.1 のうち、PsExec の特徴を説明する。PsExec を実行するクライアント端末で記録されたプロセスログのうちプロセス起動と通信試行の一部を図 4.1 に示す。図 4.1 はクライアント端末からリモート端末へマルウェアをコピーし実行する引数を PsExec に与えて実行している様子を示している。また、リモート端末で記録されたプロセスログのうち、プロセス起動と通信試行の一部を図 4.2 に示す。図 4.2 はクライアント端末から通信を受け取り、コピーされたマルウェアが実行されている様子を示している。

図 4.1 の上段と下段を比較すると、PsExec を実行すると PsExec のプロセスがリモート端末へ通信していることがわかる。次に、図 4.2 から、通信を受け取ったリモート端末はクライアント端末と通信をした後に PSEXESVC.exe を実行する。このときリモート端末はクライアント端末間で通信したことは図 4.1 の下部と図 4.2 の上部の IP アドレス・ポート番号を比較することでわかる。また、その際リモート端末が通信を行っているプロセスは RpcSs に属する svchost であることもわかる。その後、リモート端末では起動された PSEXESVC.exe が子プロセスを起動することによってクライアント側で与えられたコマンドを実行する。このような流れで PsExec はリモート端末で任意のコマンドが実行される。

以上の結果から、PsExec が内部通信を行う際には次の特徴があることがわかった。

- (1) クライアント端末が PsExec のプロセスを起動し、PsExec がリモート端末へ向けて通信
- (2) リモート端末が PsExec による通信試行と同一の IP・ポート番号を持つ通信を RpcSs に属する svchost でクライアント端末へ向けて通信
- (3) リモート端末で PSEXESVC が起動

(4)PSEXESVC が親プロセスとなりリモートコマンドを実行



### 4.1.3 感染経路特定手法

前節で述べた特徴を用いて感染端末内のプロセス挙動も含めた感染経路を検知する手法を提案する。基本的には、端末のプロセスログに表 4.1 の特徴が存在するか検索する手法である。

- (1)不審な通信またはプロセスの検知
- (2)検知したプロセスの特定
- (3)親プロセスの起動とその直前の通信試行が表 1 のリモート端末の特徴を持つか調査
- (4)調査結果から通信元の端末を調査
- (5)特定した端末のプロセスログが表 1 のクライアント端末の特徴を持つか調査
- (6)調査結果から内部通信を行ったプロセスの特定
- (7)そのプロセスの親プロセスを調査していくことで不審なプロセス起動を発見
- (8)手順 2 をその端末で繰り返す

以上より、感染経路検知プログラムを開発した。機能要件は次の 2 つである。1 つめは、上記手順の自動的な処理である。このとき、手順(1)は既存のマルウェア検知手法や、IDS によるアラートなどが利用できる。2 つめは、情報処理機器間の関係と端末のログを統合することである。そのために、情報処理機器間の関係を RDF で表現した。ここで、ここで、情報処理機器間の関係等を RDF で表現するために定義した語彙と関係性の定義を図 4.3 に示す。あわせて、グラフ描画ツールである Graphviz を用いて RDF で記述された感染経路を可視化した。さらに、手順(2)～手順(7)はプロセスログだけでも実現可能だが、処理時間短縮のためにプロセスログを RDF に変換し、その結果を保存した DB に対して問い合わせることで実現した。

ネットワーク構造の表現に利用 * 語彙：CybOXから引用		内部侵入段階の表現に利用 * 語彙：独自定義、関係性：独自定義	
ネットワーク語彙	プロセス語彙	語彙	定義
network interface	process name	host name	CybOXと同じ
ipv4 address	process id	status	マルウェア感染状態を示す
ipv6 address	parent process id		
mac address	service groups name		
default gateway			
host name			
port number			
		関係性	使用目的
		Penetration	ホスト名間をつなぐ
		infect process	statusとプロセス名をつなぐ

図 4.3 本手法で定義したオントロジーの一部

## 4.2 実験

この節ではシミュレーション実験について記述する。この実験では、内部侵入段階における攻撃者の行動をシミュレートする。その後、提案した感染経路検知手法を適用し、その有効性と開発したプログラムの性能を評価する。

### 4.2.1 実験環境

実験環境はVMWare ESXi 5.5 を用いて仮想ネットワーク環境を構築した。構築したシステムの構成を図 4.4 に示す。最低限の組織ネットワークを作成し、ローカルネットワークと DMZ に分割した。IDS とプロキシサーバは標的型メール攻撃のトリガーを発生するために設置した。ローカルネットワークには業務を行う端末として Windows 端末を 10 台設置した。この Windows 端末にはこれまでの事例から、各端末で同一のローカル管理パスワードを設定した。これにより 1 つの端末の管理者情報を窃取すれば、他端末へ容易に侵入できる。同様に、各端末は PsExec を利用可能な環境を構築した。

次に、標的型メール攻撃をシミュレートするために利用した攻撃ツールについて述べる。標的型メール攻撃をより現実的にシミュレートするため、初期感染に用いる RAT として ShinoBOT と Metasploit を用いた。ShinoBOT を実行すると、自動的に PC のネットワーク構成やシステム情報などが収集され ShinoBOT サーバに送信される。また、ShinoBOT サーバにアクセスすると ShinoBOT を実行した端末の情報を閲覧し、端末に対して自由にコマンドを実行させることができる。同様に、Metasploit を用いることで感染端末の権限昇格が可能であり、侵入を拡大することが可能である。今回は Metasploit を用いて RAT を埋め込んだ Word ファイルを作成した。RAT を埋め込んだ Word ファイルの作成には Word の脆弱性、特に「MS10-087」[38]を利用した。

そして、内部侵入段階でリモート端末に送信するマルウェアとして、上記と同様に ShinoBOT と Metasploit を用いた。ここで、Metasploit は「Meterpreter reverse https」を用いてマルウェアを作成した。これをリモート端末で実行することで、攻撃者が初期感染端末と同様にリモート端末を制御可能となる。

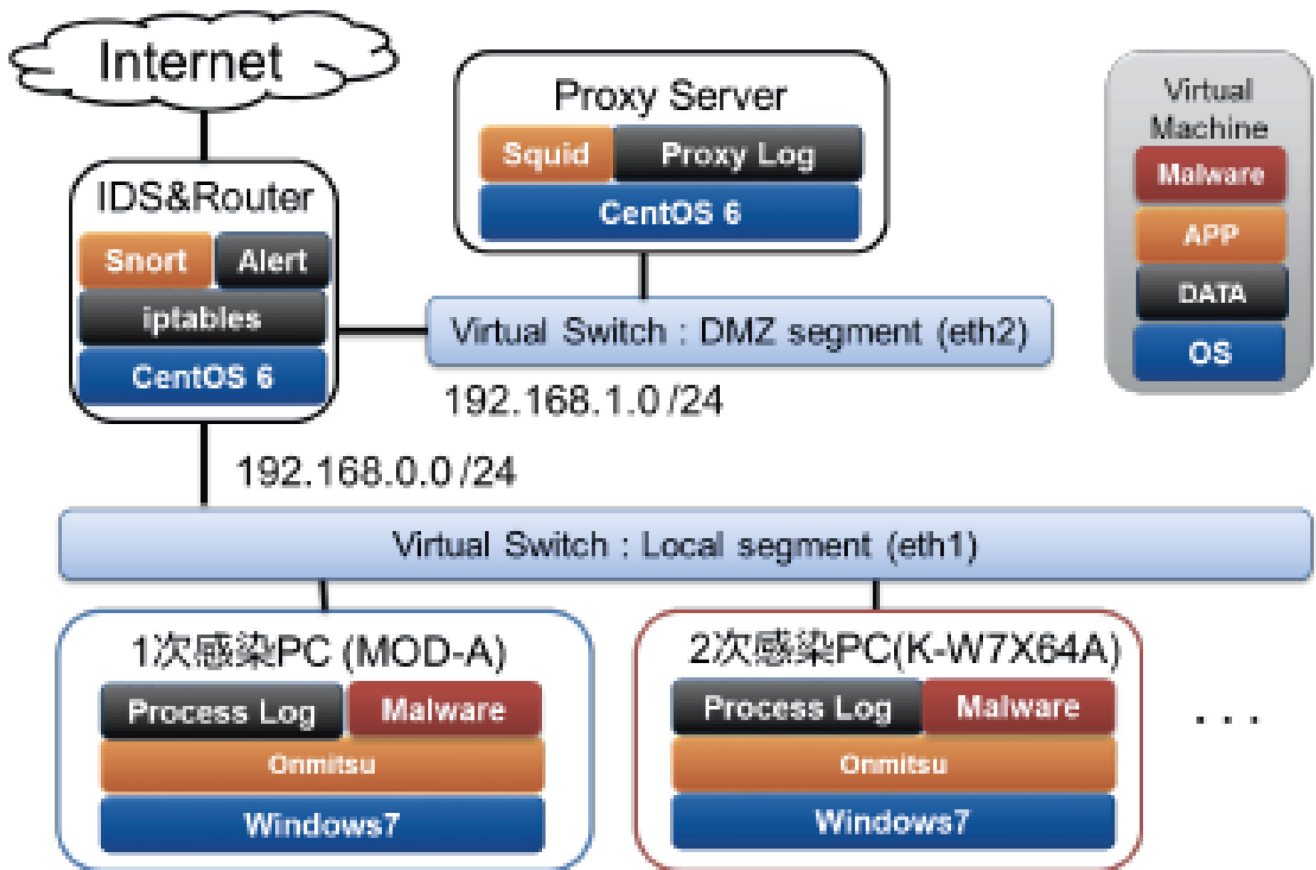


図 4.4 実験システムの概要

## 4.2.2 実験手順

標的型メール攻撃をシミュレートするために次の手順で内部調査段階までの攻撃を実施した。このときシミュレートする感染経路は5次感染までを実験対象とした。その理由は、FireEyeの報告書など、これまでの標的型メール攻撃の事例を調査した結果、内部調査段階で5次感染以上に広がった事例がなかったためである。よって、今回の小規模な実験環境でも現実的な評価が可能だと考えている。また感染拡大をする際の攻撃者行動はC&Cサーバを逐次介するものとした。

- (1)感染源端末でマルウェアを実行
- (2)感染源端末の管理者情報を窃取
- (3)感染先端末へ向けて内部通信を実行
  - (a)感染先端末の情報収集
  - (b)感染先端末へマルウェアを転送し、実行

上記の手順を5次感染まで実施する。さらに、使用するマルウェアや内部通信に用いるツールを表4.2の場合に分けて記述する。

表 4.2 攻撃手法と検知手法の分類

攻撃ツール	内部通信	検知手法
ShinoBOT	PsExec	プロセスログのみ
Metasploit	at	RDF 集約あり
	wmic	RDF 集約なし

実験中、各感染PCはOnmitsuによりプロセスログが記録される。各端末のプロセスログをRDFに変換し、提案手法を適用する。このとき、RDFを用いた手法は「RDF集約あり」と「RDF集約なし」で適用した「RDF集約あり」とは、各端末のログを集約してから提案手法を適用することで感染経路を一度に検知する手法である。「RDF集約なし」とは、1端末のログに提案手法を適用し、次に調査すべき端末のログを特定していくことで追跡する手法である。以上の2つの結果とプロセスログのみで適用した結果の計3パターンの処理時間を比較することで、実世界に適用可能か評価する。

### 4.2.3 実験結果

各実験における感染経路検知結果を表 4.3 に示す. すべての実験において, 感染経路が検知できた. その中で, case1 における感染経路追跡結果の一部を図 4.5 に示す. 左端の RDF トリプル(MOD-A, Penetration, K-W7X64)は感染経路が検知されたことを示している. また, 上部で囲われた RDF トリプル群から感染源となるプロセス(ShinoBOT)が追跡できたとわかる.

表 4.3 感染経路検知結果

		経路追跡 (5 次感染)	感染源の特定
ShinoBOT	PsExec	○	○
	at	○	○
	wmic	○	○
Metasploit	PsExec	○	○
	at	○	○
	wmic	○	○

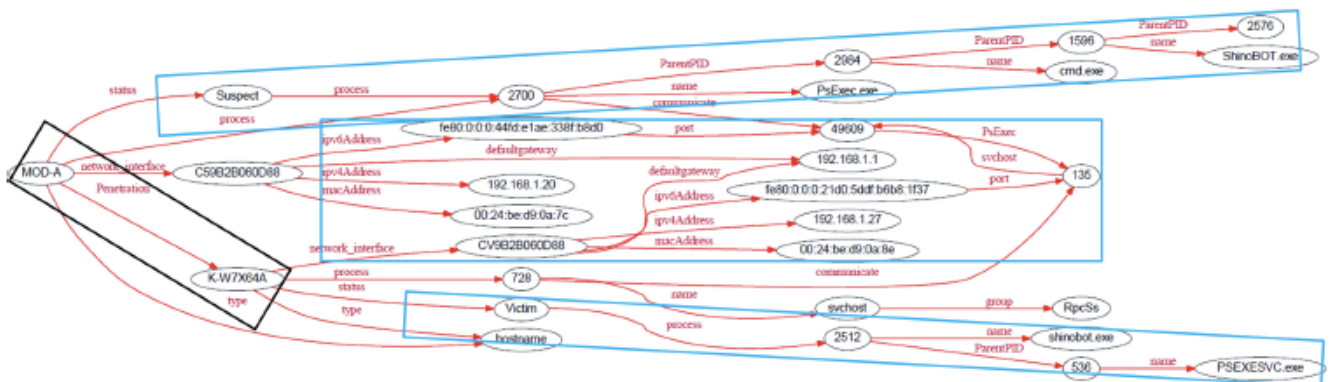


図 4.5 感染経路検知結果の一部(ShinoBOT)

次に、各手法における平均追跡時間のグラフを図 4.6 に示す。縦軸が追跡時間、横軸は追跡調査に用いた各端末を示している。各手法での平均追跡時間は CybOX では約 120 秒、RDF ログ集約ありでは約 20 秒、RDF ログ集約なしでは約 5 秒であった。図 4.6 から、RDF ログ集約なしの手法がより短時間で処理できたことがわかる。

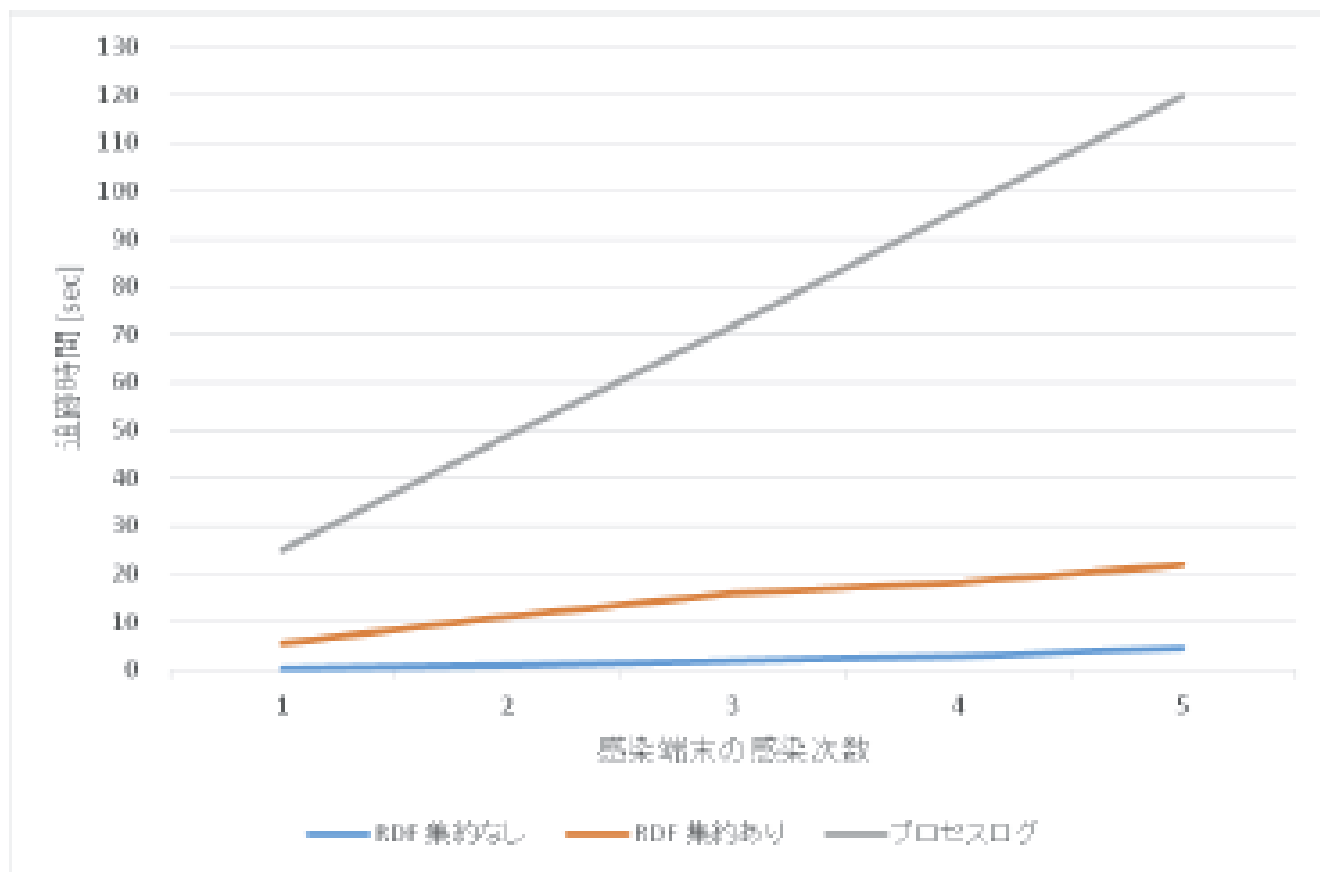


図 4.6 処理時間の平均[sec]

## 4.3 考察

### 4.3.1 感染経路検知手法の検証

はじめに、感染経路検知結果について考察する。

初期感染端末での手順 1 と手順 3 の結果の一部を図 4.7 に示す。このとき、手順 1 の結果が①を手順 3 の結果が②～④に対応している。

図 4.7 の上段のコマンドラインは ShinoBOT の実行を示している(①)。中段は ShinoBOT が PsExec を実行し(②,③)、2 次感染端末へ向けて通信した(④)ことを示している。このとき、リモート端末への命令は③のコマンドラインから ShinoBOT の転送であるとわかる。2 次感染端末での手順 3 の結果の一部を図 4.8

に示す。ShinoBOT の実行(④)とその直前の通信試行(③)を示している。なお、このとき実行された ShinoBOT が PsExec によって実行されたマルウェアで図 4.5 の下部の RDF トリプル群と図 4.8 の②と④を比較することにより 2 次感染端末で実行されたマルウェアの親プロセスがたどれていることがわかる。図 4.5 の中央の RDF トリプル群と図 4.8 の①、図 4.7 の④を比較することで内部通信が適切に関連付けられているとわかる。図 4.5 の上部の RDF トリプル群と図 4.7 の①を比較することにより初期感染端末で実行されたマルウェアがたどれていることがわかる。

```

① 2015,07,22,11,40,04,0216,PROCESS_LAUNCH,1596,2576,¥??¥C:¥Users¥Yui¥Downloads¥ShinoBOT.exe,"C:¥Users¥Yui¥Downloads¥ShinoBOT.exe",...
~~~~~
② 2015,07,22,12,07,34,0377,PROCESS_LAUNCH,2984,1596,¥??¥C:¥Windows¥System32¥cmd.exe,"C:¥Windows¥system32¥cmd.exe" /c C:¥Users¥Yui¥Downloads¥PsExec.exe -s ¥¥K-W7X64A -c "C:¥Users¥Yui¥Downloads¥ShinoBOT.exe"
③ 2015,07,22,12,07,34,0424,PROCESS_LAUNCH,2700,2984,¥??¥C:¥Users¥Yui¥Downloads¥PsExec.exe,C:¥Users¥Yui¥Downloads¥PsExec.exe -s ¥¥K-W7X64A -c "C:¥Users¥Yui¥Downloads¥ShinoBOT.exe"
④ 2015,07,22,12,07,34,0440,NETWORKV6,2700,...,fe80:0:0:0:44fd:e1ae:338f:b8d0,49609,fe80:0:0:0:21d0:5ddf:b6b8:1f37,135,6

```

図 4.7 初期感染 PC のプロセスログ(ShinoBOT)

```

① 2015,07,22,11,56,33,0753,NETWORKV6,728,...,fe80:0:0:0:21d0:5ddf:b6b8:1f37,135,fe80:0:0:0:44fd:e1ae:338f:b8d0,49609,6
② 2015,07,22,11,56,55,0359,PROCESS_LAUNCH,536,496,¥??¥C:¥Windows¥PSEXESVC.exe,C:¥Windows¥PSEXESVC.exe,....
~~~~~
③ 2015,07,22,12,07,34,0477,NETWORKV6,728,...,fe80:0:0:0:21d0:5ddf:b6b8:1f37,135,fe80:0:0:0:44fd:e1ae:338f:b8d0,49695,6
④ 2015,07,22,12,07,55,0537,PROCESS_LAUNCH,2512,536,¥??¥C:¥Windows¥ShinoBOT.exe,"ShinoBOT.exe"

```

図 4.8 2 次感染 PC のプロセスログ(ShinoBOT)

以上の結果から、感染経路が検知でき、さらに感染源となるプロセスも追跡できたことがわかった。

### 4.3.2 開発プログラムの検証

作成したプログラムの性能評価について考察する。

図 4.6 から、感染経路検知時間はプロセスログの手法は RDF 集約ありの手法と比べて約 6 倍の時間が必要であった。また、RDF 集約なしの手法と比べると約 60 倍の時間が必要であった。これは、RDF の手法では抽出し情報に対して問い合わせるだけで感染経路が導出できる。それに対し、プロセスログの手法では各端末のログのすべてを逐次読み込んで判断する必要がある。これにより、処理時間に差ができたと考えられる。さらに、RDF 集約ありの手法は RDF 集約なしの手法と比べると約 4 倍の時間が必要であった。これは、RDF 集約ありの場合では 10 端末の集約した情報を調査するのに対して、RDF 集約なしの手法では 1 端末の情報を調査するためだと考えられる。

これらの性能結果から、感染経路を検知するには RDF 集約なしの手法が有効であると考えられる。また図 4.6 から、感染経路を検知する際、ログのサイズに比例した時間を要すると考えられる。しかし、標的型攻撃において 10 次感染以上に感染を拡大することはほとんど考えられず、その場合でも 10 秒程で検知できると考えられる。そのため、この検知ツールを実世界に適用しても処理時間に大きな問題は生じないと考えられる。

しかし、今後は感染源のプロセスまで遡上の感染経路以外にも、どのような経路で拡散したのかという前進的な感染経路の追跡も検討する。この前進的な感染経路の追跡には、ネットワーク内の全端末を調査する可能性もある。このとき、RDF ログ集約なしの手法では 1 端末の調査あたり平均 1 秒程度であったが、これをネットワーク内の全端末に適用することを考えると、1000 端末では 1000 秒程度かかると予想され、より高速化が求められる。一方で、大規模な RDF データの問い合わせを高速化する研究が進められている [39]。これにより、前進的な感染経路の追跡でも RDF の手法はより短時間で検知できると期待される。

以上の結果から、内部通信の特徴を用いることでプロセスも含めた感染経路が適切に追跡可能だとわかった。これにより、標的型攻撃の内部侵入・調査段階で侵害範囲が拡大された際にその感染経路が追跡可能となる見通しが得られた。



## 4.4 開発プログラムの課題

本章では複数端末のプロセスログを解析することで標的型攻撃における内部侵入を検知し、その感染経路を検知する手法をについて検討した。このとき、機器間の関係をオントロジーで記述した。これにより各端末のログとネットワーク構造を統合でき、感染経路が検知可能となった。実験で感染経路検知手法の有効性を検証した。実験の結果、マルウェアを検知した 5 次感染端末から初期感染端末の感染源プロセスを発見することができた。また、開発プログラムでの処理時間は RDF 集約なしの手法を用いると高々 5 秒程度だった。これにより、感染経路を検知する手法に対する解決の見通しを得た。

今後は図 4.9 にある課題を検討しながら標的型攻撃における動的かつ総合的な攻撃検知手法についての検討を進めていく。

### 1. 内部侵入ツールのより多くの適用

内部侵入ツールとして Windows の標準ツールである Psexec を検討した。

しかし、攻撃者はこれらの標準ツール以外にも多様な内部侵入ツールを作成している。そのため、それらの挙動を調査しその結果を DB にして活用する手法を検討する必要がある。

### 2. 拡散経路を追跡する前進的な経路検知手法の検討。

このとき、遡上の経路と異なり、調査端末が増大することが予想される。そのため、感染経路の検知時間の短縮方法の検討が必要である。

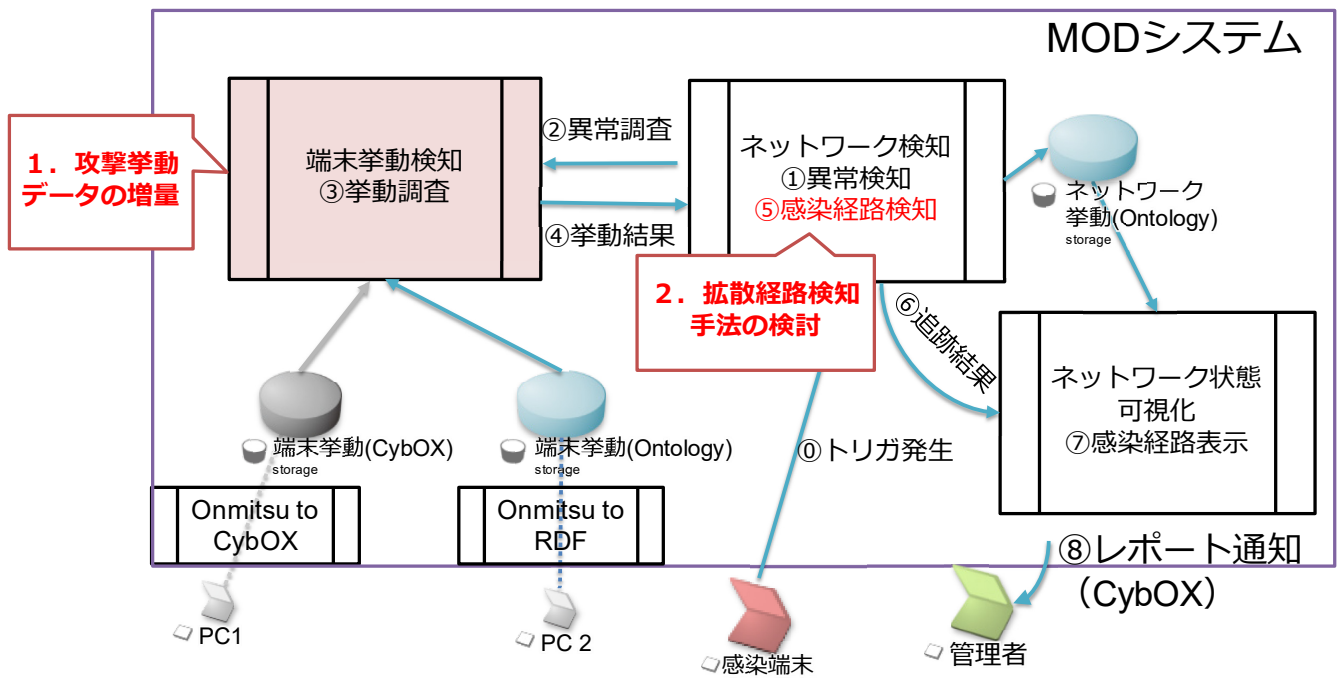


図 4.9 開発システムの課題の概要

## 第5章 今後の課題

本研究では第1章で述べた問題に対して、下記の通り検討した。

### ① 痕跡の削除が困難なログの記録：

1. **Onmitsu** を用いた揮発性情報の取得 (2.2.1 項)

### ② 事象情報の表現形式の統一：

1. **CybOX** の動的情報の表現可用性の検討 (3.3 節)
2. **Onmitsu** のログから **CybOX** 形式へ変換するツールの開発 (3.3 節)
3. **CybOX** を用いたログ情報と攻撃挙動の統一した表現の検討 (3.4 節)

### ③ 複数の事象を組み合わせた自動的な標的型攻撃の検知手法

1. プロセスログを用いた感染検知手法の検討 (3.5 節)
2. 感染源検知手法の提案とツールの開発 (第4章)

本論文では、それぞれの検討に対して解決の見通しを得たが今後の改良点として下記が挙げられる。

1. 様々なログを **CybOX** へ変換することによる表現可用性の向上
2. マルウェアプロセスパターンの増量
3. 開発プログラムの性能向上
4. 未知の攻撃に対する対応

具体的には、3.3 節の結果から **Onmitsu** で取得したプロセスログや他の動的解析ソフトから得られたプロセスログを **CybOX** で表現するには基本的には問題ないと分かった。しかし、3.5 節で述べた手法で分析する際には、コマンドラインの意味づけや、**Action\_Aergument** に記述したコマンドラインの実行ファイルと引数の切り分けなどができるように **CybOX** を拡張すればより表現可用性が高まると考えられる。そこで、今後も様々なログを対象として **CybOX** で記述し評価するなかで **CybOX** の拡張を検討していく必要がある。同時に、拡張した **CybOX**

で記述した文書を分析してセキュリティ専門家の知識を獲得可能できるかも評価する必要がある。

同様に、3.5 節で述べたようにマルウェアプロセスパターンではプロセスツリーも記述できれば検知精度が向上すると期待される。プロセスツリーを記述するためにはマルウェアの振る舞いを記述する仕様である **MAEC** と連携することが必要である。さらに、第4章で述べたように、実環境で適用するためには開発プログラムの性能問題も解決する必要がある。**Onmitsu** で記録されるログは軽量だが、**CybOX** で記述することでファイルサイズが 10 倍程度となり、現在の変換・比較処理ではある程度の時間を要する。そのため、動的検知のリアルタイム性が損なわれる恐れがある。さらに今後、マルウェアパターンや **CybOX** で記述されるログは膨大な量になると予想される。そのため、変換手法や比較手法の高速化は今後の重要な課題となる。さらに、標的型攻撃に用いられる脆弱性はゼロデイ攻撃のようにその対策がとられていない脆弱性をつかれて攻撃を受ける事が多い、そのため、未知の攻撃に対して即座に対応する手段または、その攻撃を予測する手法が重要となる。例えば、**VirusTotal** 等のマルウェアの収集サイトから自動的にマルウェアを収集し、**Lastline** のような動的挙動解析ツールで分析し、その結果から攻撃シナリオを予測し、そのシナリオを **CybOX** で記述することで最新の攻撃への対応または新たな攻撃の予測が可能となると期待される。

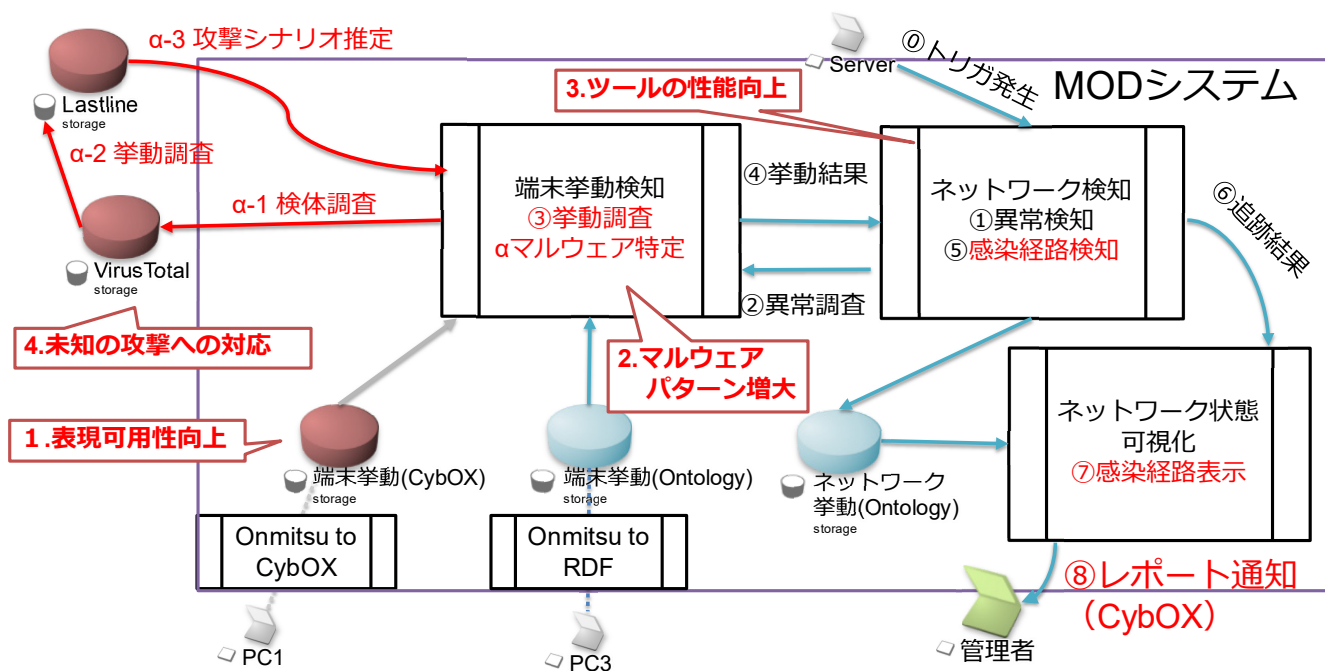


図 5.1 今後の課題の概要

## 第6章 結論

本研究では第1章で述べた問題に対して、下記の通り検討した。

### ① 痕跡の削除が困難なログの記録：

2.2.1項で攻撃者による痕跡削除が困難な情報の1つである揮発性情報を取得するためのツールである **Onmitsu** について述べ、そのログ形式とログを活用した攻撃検知について述べた。

### ② 事象情報の表現形式の統一：

第3章でサイバー攻撃表記仕様と分析方式について検討した。**CybOX** をサイバー攻撃表記仕様として、プロセスログを分析対象として検討を進めた。その結果、プロセスログを **CybOX** で表現するには基本的には問題なく、サイバー攻撃表記仕様として **CybOX** は可用性が高いことがわかった。さらに、コマンドラインの意味づけや **Action\_Aergument** に記述したコマンドラインの実行ファイルと引数の切り分け等ができるように **CybOX** を拡張したほうがより可用性が高まることもわかった。

### ③ 複数の事象を組み合わせた自動的な標的型攻撃の検知手法

3.5節で **CybOX** を用いた新たなマルウェア検知手法としてマルウェアプロセスパターンで検知する手法を提案し、その実現可能性を検証した。その結果、適切なしきい値を設定することでマルウェアの検知が可能となるとわかり、実フィールドでの有効性を示す結果が得られた。

第4章では複数端末のプロセスログを解析することで標的型メール攻撃における内部侵入を検知し、その感染経路を検知する手法をについて検討した。このとき、機器間の関係をオントロジーで記述した。これにより各端末のログとネットワーク構造を統合でき、感染経路が検知可能となった。実験で感染経路検知手法の有効性を検証した。実験の結果、マルウェアを検知し

た 5 次感染端末から初期感染端末の感染源プロセスを発見することができた。また、開発プログラムでの処理時間は RDF 集約なしの手法を用いると高々5秒程度だった。

これにより、感染経路を検知する手法に対する解決の見通しを得た。今後は第 5 章で述べた課題を検討し標的型メール攻撃における動的かつ総合的な攻撃検知手法についての検討を進めていく。

## 参考文献

- [1] 独立行政法人情報処理推進機構：『高度標的型攻撃』対策に向けたシステム設計ガイド,入手先  
<https://www.ipa.go.jp/security/vuln/newattack.html>（参照 2015-03-09） .
- [2] 独立行政法人情報処理推進機構：標的型サイバー攻撃の事例分析と対策レポート 2012, 入手先<https://www.ipa.go.jp/les/000014188.pdf>（参照 2015-03-09） .
- [3] 独立行政法人情報処理推進機構：サイバーレスキュー隊(J-CRAT) 技術レポート 2017, 技術報告(2018).
- [4] 独立行政法人情報処理推進機構：情報セキュリティ 10 大脅威 2018, 入手先<https://www.ipa.go.jp/les/000065376.pdf>（参照 2018-10-27） .
- [5] 特定非営利活動法人日本ネットワークセキュリティ協会：情報セキュリティインシデントに関する調査報告書, 技術報告(2018).
- [6] Mimura, S. and Sasaki, R.: Method for Estimating Unjust Communication Causes Using Network Packets Associated with Process Information, The International Conference on Information Security and Cyber Forensics (InfoSec2014), The Society of Digital Information and Wireless Communication, pp. 44-49 (2014).



- [7] MITRE: CybOX - Cyber Observable eXpression, MITRE (online), available from <<https://cybox.mitre.org/>> (accessed 2015-03-09).
- [8] Consortium, W. W. W. et al.: RDF 1.1 Primer, available from <<http://www.w3.org/TR/rdf11-primer/>> (accessed 2015-03-09).
- [9] Salahi, A. and Ansarinia, M.: Predicting Network Attacks Using Ontology-Driven Inference, arXiv preprint arXiv:1304.0913 (2013).
- [10] 藤巻伶緒：ホームネットワークにおけるトポロジ情報の記述に関する研究(2018).
- [11] Moser, A. and Cohen, M. I.: Hunting in the enterprise: Forensic triage and incident response, Digital Investigation, Vol. 10, No. 2, pp. 89-98 (2013).
- [12] Google: GRR Rapid Response: remote live forensics for incident response, available from <<https://github.com/google/grr>> (accessed 2015-03-09).
- [13] 満永拓邦, 松田亘, 藤本万里子：SDN と STIX を組み合わせた情報共有とインシデント対応の自動化, コンピュータセキュリティシンポジウム 2017 論文集, Vol. 2017, No. 2 (2017).
- [14] 高橋浩司, 花房比佐友, 堀口良太：GPU を用いた大規模ネットワーク交通流シミュレーションにおける経路探索計算の高速化(モバイルネットワークとアプリケーション), 電子情報通信学会技術研究報告: 信学技報, Vol. 117, No. 71, pp. 33-38 (2017).

- [15] 小野悟：柔軟に機能するキャンパスネットワークの実証的研究，博士論文，静岡大学(2017).
- [16] Slot, T.: Detection of APT Malware through External and Internal Network Traffic Correlation, Master's thesis, Univ. of Twente (2015).
- [17] Nakazato, J. et al: A Suspicious Processes Detection Scheme using Host Based IDS, Proc. of Symposium on Cryptography and Information Security, No. 2A1-5(2015).
- [18] Skrzewski, M.: System Network Activity Monitoring for Malware Threats Detection, Computer Networks, pp.138-146 (2014).
- [19] 田中功一ら：ログ解析によるマルウェア侵入検知手法の提案，マルチメディア、分散協調とモバイルシンポジウム 2014 論文集， Vol. 2014, pp. 522-529 (2014).
- [20] 川口信隆ら：不審活動の端末間伝搬に着目した標的型攻撃検知方式，情報処理学会論文誌， Vol. 57, No. 3, pp. 1022-1039 (2016).
- [21] Sato, M., Teshigawara, Y. and Sasaki, R.: Proposal for Knowledge Model Using RDF-based Service Control for Balancing Security and Privacy in Ubiquitous Sensor Networks, Informatics Society, p. 89.
- [22] Sato, M., Sugimoto, A., Hayashi, N., Isobe, Y. and Sasaki, R.: Proposal of a Method for Identifying the Infection Route for Targeted Attacks Based on Malware Behavior in a Network, Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec), 2015 Fourth International Conference on, IEEE, pp. 40-45 (2015).

- [23] Noguchi, H. et al: Search System for Behavior Time Segments from Accumulated Sensor Data in Room Environment, Future Generation Communication and Networking. (FGCN 2007), pp. 19-24 (2007).
- [24] K. Fujinami, and T. Nakajima, An Information Management Infrastructure for Sentient Artefact-based Smart Spaces (in Japanese), IPSJ Transactions on Computing System, Vol. 47, No. SIG12(ACS 15), pp. 399-410 (2006)
- [25] A. Held, S. Buchholz, and A. Schill, Modeling of Context Information for Pervasive Computing Applications, In Proceeding of the World Multiconference on Systemics, Cybernetics and Informatics, pringer (2002)
- [26] H. Noguchi, K. Tanaka, T. Mori, T. Sato, Room Situation Search System Based on RDF Describing Room Object as Target of Human Behavior, Technical Report of IEICE, Vol. 104, No. 725, pp. 31-36 (2005)
- [27] O. Sacco and A. Passant, A Privacy Preference Ontology (PPO) for Linked Data, Procs of the 4th Workshop about Linked Data on the Web (LDOW- 2011) (2011)
- [28] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thurainsingham, A Semantic Web Based Framework for Social Network Access Control, Proceedings of the 14th ACM symposium on Access control models and technologies, pp. 177-186 (2009)
- [29] P. Jagtap, A. Joshi, T. Finin, and L. Zavala, Preserving Privacy in Context-aware Systems, 2011 Fifth IEEE International Conference, pp. 149-153 (2011).

- [30] Carroll, J. et al: Jena: implementing the semantic web recommendations, Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, ACM, pp. 74-83 (2004).
- [31] Ellson, J. et al: Graphviz—open source graph drawing tools, International Symposium on Graph Drawing, Springer, pp. 483-484 (2001).
- [32] Compton, M. et al.: The SSN ontology of the W3C semantic sensor network incubator group, Web semantics: science, services and agents on the Worldwide Web, Vol. 17, pp. 25-32 (2012).
- [33] Shota, S.: ShinoBOT -the rat/bot malware simulator-, available from <<http://shinobot.com/>> (accessed 2015-03-09).
- [34] 佐藤信, 杉本暁彦, 林直樹, 磯部義明, 佐々木良一: マルウェアによるネットワーク内の挙動を利用した標的型攻撃における感染経路検知ツールの開発と評価, 情報処理学会論文誌, Vol. 58, No. 2, pp. 366-374 (2017).
- [35] Mandiant: M-Trends R 2015: A VIEW FROM THEFRONT LINES, Technical report, a FireEye Company (2015).
- [36] Microsoft: Microsoft TechNet Windows Sys-internals PSEXEC, Microsoft (online), available from <<http://technet.microsoft.com/ja-jp/sysinternals/bb897553.aspx>> (accessed 2015-03-09).

- [37] 朝長秀誠：攻撃者が悪用する Windows コマンド(2015-12-02),  
JPCERT/CC ( オンライン), 入手先  
<<https://www.jpccert.or.jp/magazine/acreport-wincommand.html>> (参照  
2016-01-01) .
- [38] Microsoft: マイクロソフトセキュリティ情報 MS10-087, Microsoft  
( オンライン) , 入手先<<https://technet.microsoft.com/ja-jp/library/security/ms10-087.aspx>> (参照 2016-01-01) .
- [39] 藤原浩司, 兼岩 憲：大規模 RDF グラフのための効率的なクエリ解決, 人工知能学会論文誌, Vol. 29, No. 4, pp. 364-374 (2014).

## 謝辞

本研究を進めるにあたり、様々な御意見・御指導を頂きました佐々木良一教授、猪俣敦夫教授に深く御礼申し上げます。また、日頃からお世話になっている勅使河原可海先生に感謝を申し上げます。

研究活動や論文執筆に際して、多大なご指導、ご助言をいただいた日立製作所の杉本暁彦様、林直樹様、磯部義明様、また LIFT プロジェクトメンバーに深く感謝申し上げます。

公私にわたり多大なご助言をいただきました高橋雄志さんに深く感謝申し上げます。

研究室において日々様々な支援をしてくれた情報セキュリティ研究室の皆様に深く感謝いたします。

最後に温かく見守り、心から応援し支えてくれた両親をはじめ、家族の皆様に深く感謝申し上げます。