NAIST-IS-MT1651125

修士論文

仮想マシン内部観察に基づく
マルウェア悪性機構抽出の自動化手法の提案

与那嶺 俊

2018年3月15日

奈良先端科学技術大学院大学
情報科学研究科

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

与那嶺 俊

審査委員：

門林 雄基 教授　（主指導教員）
藤川 和利 教授　（副指導教員）
林 優一 教授　（副指導教員）

仮想マシン内部観察に基づく
マルウェア悪性機構抽出の自動化手法の提案*

与那嶺 俊

内容梗概

マルウェア解析の目的の一つに攻撃者の意図（悪性機構）を明らかにすることが挙げられる。マルウェア解析はマルウェアが実行した API コールから機能を一つずつ明らかにしていくことが一般的だが、その作業は解析者によって手作業で行われることが多い。また、高機能化したマルウェアはデータの操作に難読化を組み合わせる等の手法で解析を妨害するが、マルウェアが実行した複数の API コールの関係性から機能を推定する自動化された方法はない。本研究ではファイルへのアクセスを試みたマルウェアがデータを処理していく過程を追跡し、悪性機構の識別の自動化を Virtual Machine Introspection によって実現する。開発した悪性機構の検出システムの Spaniel は監視対象ファイルの読み出しから派生した処理をテイント解析によって追跡し、不正活動に関する痕跡の収集を自動で行う。攻撃者のシステム侵入後の内部活動モデルである ATT&CK を用いた検証では 26%の攻撃モデルにおいて不正活動の指標を抽出できることが分かった。

キーワード

マルウェア、特徴抽出、テイント解析、仮想マシン内部観察

*奈良先端科学技術大学院大学 情報科学研究科 修士論文, NAIST-IS-MT1651125, 2018 年 3 月 15 日.

Automated Characterization of Malware's High-level Mechanism Using Virtual Machine Introspection*

Shun Yonamine

Abstract

One of the goals of malware analysis is to figure out the intention of the attacker (high-level mechanism). In traditional malware analysis, the experts monitor the APIs and other activities of the system in order to identify malicious functions. However, those tasks are manual and time-consuming. Additionally, the level of sophistication of recent malware renders traditional malware analysis almost obsolete. For instance, some malware can use obfuscation techniques to thwart malware analysis. There is no methodology for associating multiple APIs to identify malicious mechanisms in automated manners. In this research, we propose a virtual machine introspection-based method for automatically identifying high-level mechanisms. We developed Spaniel, a prototype of our proposed method, which performs taint analysis to track malicious processing that derives from file read of the watched file and collects traces of malicious activities. For evaluation, we used adversary behavior models defined in ATT&CK and Spaniel identified key indicators that cover 26% of those models.

Keywords:

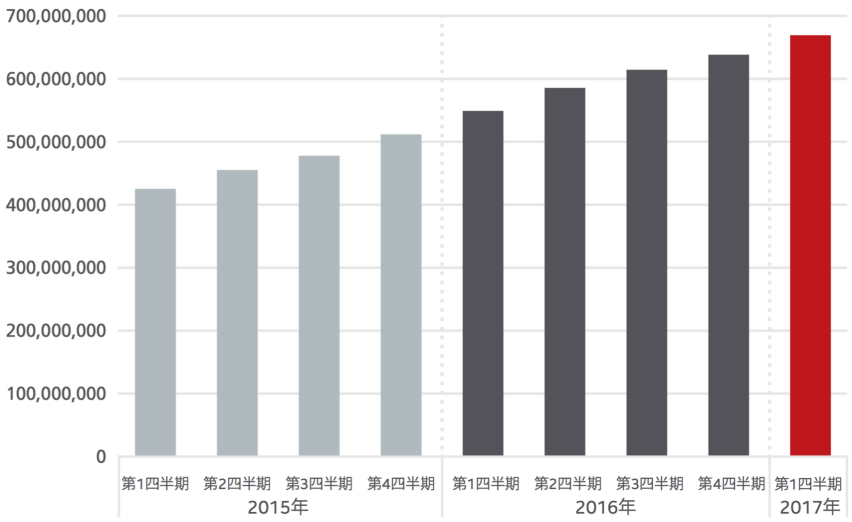
Malware, Characterization, Taint Analysis, Virtual Machine Introspection

*Master's Thesis, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT1651125, March 15, 2018.

目 次

図 目 次

表 目 次

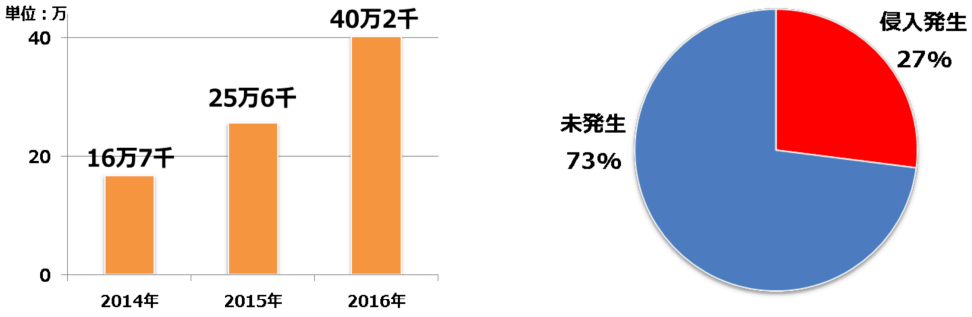


出典 : McAfee Labs 脅威レポート2017年6月

図 1 観測されたマルウェアの総数

1. 研究背景

サイバー攻撃のほぼ全てのケースにマルウェアが使用されている。情報処理推進機構 (IPA) が公開したレポート「情報セキュリティ10 大脅威 2017」は社会的に大きな影響を与えたセキュリティ事案を解説したものであり、個人ユーザーへの最大脅威に「インターネットバンキングやクレジットカード情報の不正利用」を、組織への最大脅威に「標的型攻撃による情報流出」を 2017 年のセキュリティ事案における注意事項として取り上げており、マルウェアへの注意喚起がなされている。Verizon 社は世界中の 65 個の組織を調査した結果に基づくデータ侵害に関するセキュリティレポート [1] を公開しており、攻撃者は金融情報をよく狙うことやインシデントの多くにマルウェアの関与が多いことを示している。NTT Security は脅威インテリジェンスプラットフォームによりインターネットトラフィクの 40%を分析した結果、政府や金融機関への攻撃やランサムウェアによる被害が多く発生していることを示している [2]。また、McAfee が提供する統計情報 (図



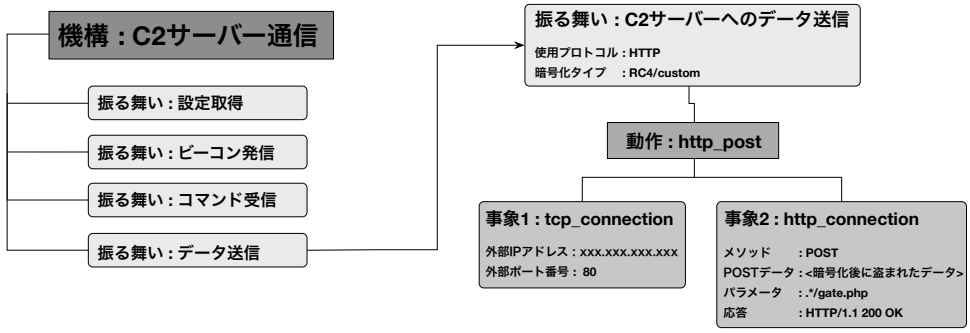
出典：トレンドマイクロ
「国内標的型サイバー攻撃分析レポート 2017年版」

図 2　標的型攻撃による侵入の疑いのあるネットワークトラフィック検出数 (左) と内部監視により確認された実際の侵入発生率 (右)

1) より、インターネット上で観測されるマルウェアの検体数は依然として増加の傾向にあることが確認できる。

インシデント発生時にはマルウェアの特徴や攻撃者の目的についての調査が必要となる。マルウェアは攻撃者の意図に応じて様々な振る舞いを示すため、確認された振る舞いについての分析を行わなければならない。攻撃者は特に自身の不正活動を隠蔽・消去する手口に力を注いでおり、マルウェア及び不正プログラムによる攻撃者の内部活動 (Lateral movement) は巧妙化と高度化を続けているため、解析はより困難になっている。図 2(左) はトレンドマイクロがネットワーク監視によって観測した標的型サイバー攻撃の疑いがあるトラフィックの検出数であるが、グラフは増加の傾向を示している。また、図 2(右) は 100 社を対象としたシステムの内部監視によって検出した侵入発生・未発生の比率である。同レポートは全ての「疑い」に反応して調査を行うことは現実的ではないとして「ファイル転送」、「痕跡除去」、「リモート実行」等の複数の内部活動の観測に着眼を置いた相関分析的なアプローチがインシデントの対応精度を向上するものとして期待の声を寄せている。本研究もまた、マルウェアの各機能に焦点を当てた対処法が有効であるという立場から取り組んでいる。

そのような中、マルウェアの特徴抽出 (Characterization) が注目されている。マ



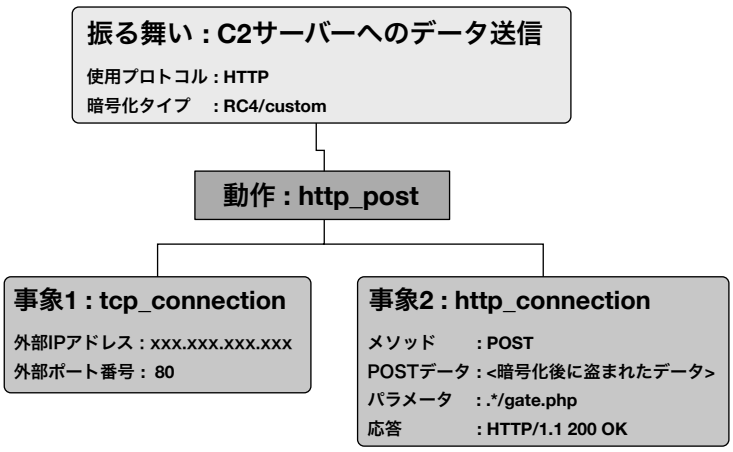
出典: The MITRE Corporation "Top Botnets and how MAEC can help keep you out of their clutches"

図 3 C2 サーバとの通信を意味する悪性機構 (左) とデータ送信における振る舞いの構成 (右)

ルウェアが持つ能力を記述し、組織間で共有することでマルウェアが発現させる脅威への対策を図ろうという試みである。例えばマルウェアは、情報の窃取や改竄を行う為にファイルへのアクセスを行うが、読み出されたデータは暗号化されたり、外部送信されたり等の形で多くの API に受け渡される。データが処理される過程の追跡は攻撃者の意図を特定する手がかりであり、マルウェアが開いたファイルや用いた API はマルウェアの悪意を示す特徴になる。

マルウェア解析は API コールの監視等の命令単位でマルウェアの挙動を一つずつ明らかにする手法が一般的であるが、人手で行うには時間と根気の要る作業である。一連の解析作業を自動で行うために Cuckoo Sandbox[3] 等のシステムが普及しているが、サンドボックスから得られる情報は API コールの観測結果についてのみの範囲に留まっており、複数の API コールの併用によって作動するマルウェアの振る舞いを解釈する術がない。

図 3(左) はマルウェアと C&C サーバ (C2 サーバ) との通信に関しマルウェアの悪意を象徴する挙動の概略とも呼べる機構 (High-level Mechanism) を 3 つの階層で表したものである [4]。図 3(右) は C2 サーバへのデータ送信の振る舞いの構成を表している。図 3 は MAEC(マイク) という標準化された規約を用いて表現したものであり、C2 サーバ通信に見られる主要な振る舞い各種に対して適切な術語



出典：The MITRE Corporation
"Top Botnets and how MAEC can help keep you out of their clutches"

図 4　C2 サーバの振る舞いの構成 (HTTP 通信と POST リクエストをデータ送信と紐付け)

を充てたものである。以降、MAEC が述べるマルウェアの機構について本論文では悪性機構と呼ぶ。

図 3 はコンポーネント単位に細分化された振る舞いの構成をマルウェアと C2 サーバとの通信に関する悪性機構であると示している。図 4 は被害端末からの外部へのデータ送信に関して振る舞いから動作単位に分割したものであるが、これらの情報はプロセスモニタ等から得られる API コール等の詳細情報に該当する。悪性機構を識別するにはそれぞれの振る舞いの関係付けを行う必要がある。

複数の API コールの動作が密接に関係する振る舞いの解釈は解析者の知識や見解に基づいている部分が大きく、手動による複雑な解析作業を解析者に要求する。近年、マルウェアの標的となっているプラットフォームは IoT デバイスにも及び、対処に必要な知識や技術の習熟コストは高まっており自動化されたアプローチが必要である。

本研究ではマルウェアが実行中に操作するデータの流れからマルウェアの意図に関する情報の自動化手法を提案する。テイント解析を用いて、特定の API コー

ルの組み合わせによって攻撃者の意図が発現したことをデータを介した結びつきから特定する。

本研究で述べるマルウェア解析とは事前に用意したマルウェアの実行による攻撃者の内部活動のシナリオの解析を指す。攻撃者の目的に焦点を当てたマルウェア解析において手動で行う解析では対処が難しい箇所を自動化されたアプローチで達成する手法を提案する。攻撃者の内部活動モデルの ATT&CK[5] で定義される Linux システムへの攻撃マトリクス [5] から選出したファイル窃取、ファイル暗号化、遠隔操作のシナリオを用いて、提案方式が悪性機構の指標の検出を行えることを示す。また、本提案方式が攻撃マトリクス中において対処可能な攻撃の種別について調査を行う。

最後に、本論文の構成について述べる。2 章では関連技術、3 章ではマルウェアの特徴に着目した関連研究について述べる。4 章では提案手法の設計と実装について述べる。5 章では提案方式によるマルウェア解析の実験を行い、6 章では提案方式の評価を行う。7 章では提案方式の課題点や展望について議論し、8 章でまとめを述べる。

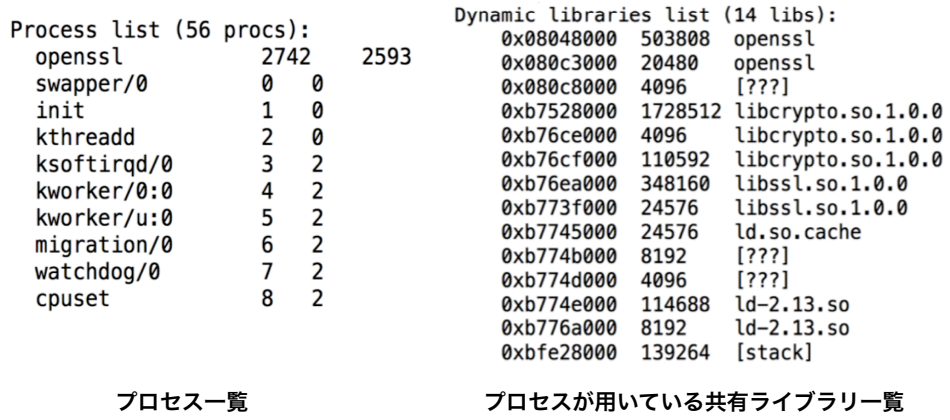


図 5 VMI によって取得可能な情報の例

2. 関連技術

本章では本研究におけるマルウェアの自動解析に用いる技術について説明を行う。

2.1 仮想マシン内部観察 (Virtual Machine Introspection:VMI)

仮想マシン内部観察 (VMI) とはハイパーバイザの外部から稼働中の仮想マシンの状態の参照や制御を行う技術である。実行中のゲスト OS からメモリ中のデータやカーネルのデータ構造を参照できるため、task_struct 構造体からプロセスに関する情報等 (図 5) を取得できる。VMI は OS にアンチウイルス等のアプリケーションを直接インストールすることなくハイパーバイザを通してシステムを内観することができる。このため、マルウェアに仮想環境内で実行されていることを検知されるリスクを回避するセキュアなセキュリティ機構を構築する手法として期待されている。

本研究で行うマルウェア解析は QEMU の仮想 CPU 上で実行されたコードの解析を行うため、取得できる情報は粒度が細かい。従って、仮想 CPU 上で実行された命令コードからプロセス名や共有ライブラリの情報の取得を行うために VMI

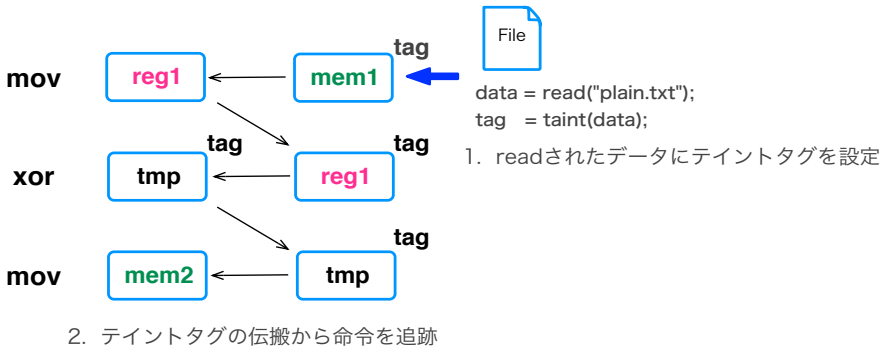


図 6 テイント伝搬の例

を用いる。粒度が細かい情報を OS レベルの情報に変換することをセマンティクスの復元と呼ぶ。セマンティクスの復元によって仮想 CPU 上で実行された命令コードからプロセス名やプロセス ID、プロセス空間にロードされた共有ライブラリ名等の取得が可能になる。

2.2 動的バイナリ計装 (Dynamic Binary Instrumentation:DBI)

動的バイナリ計装 (DBI) は実行中のプログラムにコードを挿入し、処理の書き換えや計測を行う技術である。解析対象のプログラム実行に挿入されるコードは解析者が事前にプログラムしたコールバック関数である。命令や基本ブロックの単位でコードが処理される毎にコールバック関数が呼び出され解析用の処理が挿入される。コールバック関数の処理の中で実行された命令の情報や命令が参照したメモリの内容を取得し、データフローの解析に用いる。

2.3 テイント解析 (Taint Analysis)

実行中のプログラムの動作の観測が可能な DBI を用いたプログラム実行中におけるデータフローの解析手法である。メモリ中に存在するデータにテイントタグと呼ばれるデータを設定し（これをテイントすると言う）、印が付いた監視対象

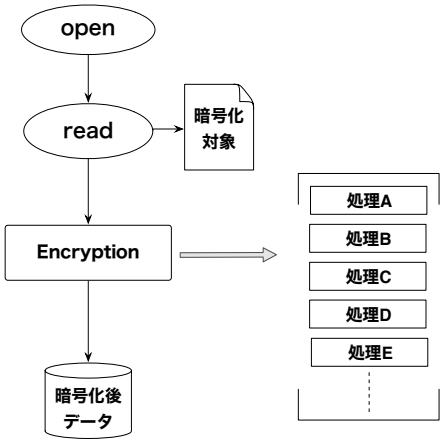


図 7 暗号化処理の実行フロー

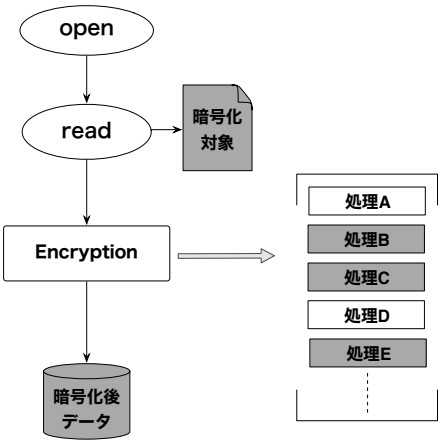


図 8 暗号化処理をテイント解析

のデータとする。テイントされたデータに対して算術等の処理が行われ他のレジスタやメモリ位置にデータのコピーや移動が発生した場合、テイントタグの伝搬 (テイント伝搬) が実行されて移動先のデータもテイントされる。テイント伝搬の例を図 6 に示す。監視対象のメモリにあるデータをテイントすると、テイントされたデータへの処理によって他のメモリ位置やレジスタにもテイントタグが設定される。テイントされたレジスタやメモリ位置に関する情報は専用のデータ構造体で管理する。従って、テイント伝搬の観測により監視対象のデータが実行中に処理される流れを捉えることができる。

本研究では監視対象のファイルデータをテイントし、テイントされたデータを処理した命令コードをテイントされた命令 (tainted 命令) として記録する。テイント伝搬を発生させた命令コードも同様に tainted 命令として記録する。命令コードが tainted 命令かどうかは命令中で処理されているレジスタやメモリのデータがテイントタグを持っているかを確認（テイントチェック）することで判定する。本研究ではテイント解析を用いることでマルウェアがシステム上に展開したデータやファイルから読み出したデータが処理される過程を追跡する。

テイント解析を用いた処理の追跡の例を説明する。図 7・8 は暗号化処理の実行フローを表しており、ファイル読み込み (read) と出力 (write) の間に実行された処理に焦点を当てている。マルウェア解析における暗号化処理の解析のモチベー

```
#define SIZE 128
int main(){
        FILE *in = fopen("plain.txt", "r");
        char data[SIZE];
        fgets(data, SIZE, in);
        for(int i=0; i<SIZE; i++){
                data[i] = data[i] ^ 0x25;
        }
        FILE *out = fopen("encrypted.txt", "w");
        fprintf(out, data, SIZE);
        fclose(in);
        fclose(out);
        return 0;
}
```

図 9　xor 暗号化の処理の例

ションの 1 つに暗号関数の情報の特定が挙げられる。図 7 はテイント解析適用前の実行フローであるが、read の後にファイルデータに対して適用された暗号化処理の詳細を知るには処理を全て１つ１つ (処理 a, b, c,...) 調べる必要がある。しかし、テイント解析を用いることで図 8 で示すように暗号化対象のファイルデータを取り扱った処理部分のみを抽出することができる。これはファイルデータが read によって読み込まれたことを API フックによって捉えた際にデータが読み込まれた先のバッファをテイントし、以降のテイント伝搬を観測した結果となる。図 8 のようにファイル暗号化において主要な役割を果たした処理中の命令コードを tainted 命令として抽出できる。

図 7・8 に関連し、テイント解析を用いた命令コード抽出の例を示す。図 9 はファイルに対し単純な xor 暗号化を行う処理である。図 10 はファイルデータにテイント解析を適用したことによって観測された tainted 命令の抽出を行った様子である。

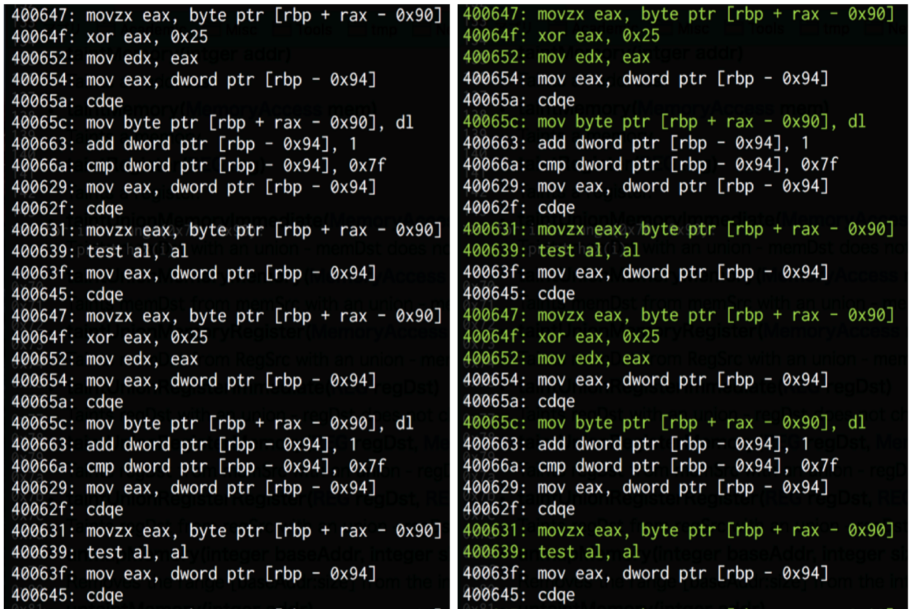
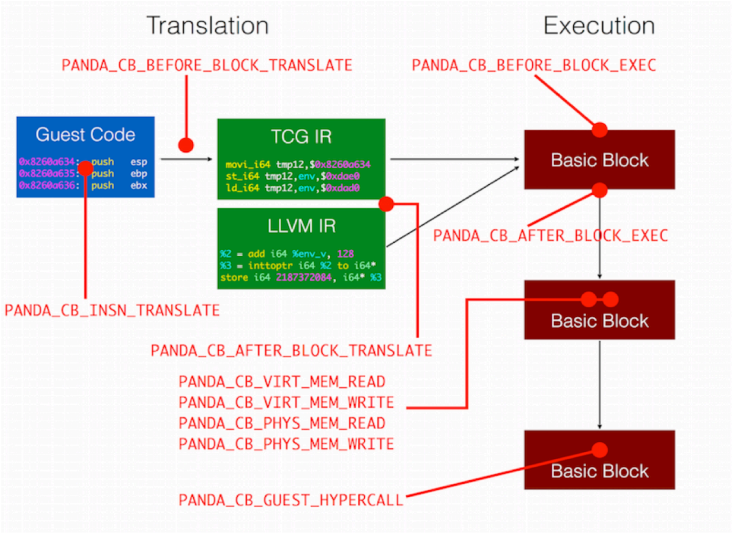


図 10　テイント解析による命令抽出 (左がテイント解析適用前の暗号化フロー、右はテイント解析による暗号化フローの解析によってテイントされた命令を緑色でハイライト表示したもの)

2.4 PANDA

PANDA[6] は CPU エミュレーターである QEMU 上で実装されたバイナリ解析プラットフォームである。DBI やテイント解析のための機能を備えており、オープンソースで公開されている。QEMU の TCG(Tiny Code Generator) がアーキテクチャ毎に異なる命令コードを LLVM の中間コードに変換することでマルチプラットフォームに対応した計装処理をサポートしており (図 11)、様々なプラットフォーム上の実行形式ファイルを対象とした解析を可能にしている。DBI による計装処理は中間コードに対して行われる。例えばテイント解析は計装処理対象の命令コードがデータの移動やコピー、もしくは算術命令を行うものであるかに応じて命令中で操作されているメモリ位置のデータを対象にテイント伝搬等の処理を適用する。

PANDA はプログラム言語 C++を用いたプラグイン作成によって機能拡張が可



出典：PANDA User Manual

図 11　コールバック関数の挿入による仮想 CPU 上におけるプログラム実行の計装処理

能であることと Record and Replay を用いたバイナリ解析をサポートしていることが主要な特徴として挙げられる。Record and Replay はマルウェア等の解析対象プログラムの実行中に、ゲスト OS のスナップショットと仮想マシンの稼働中に観測された命令コードや外部デバイスからの入力等の非決定性入力の record(記録) を行い、記録されたプログラム実行の replay(再生) によって再現性のある解析をサポートする (図 11)。

本研究では QEMU で動作する仮想マシン上でマルウェアを動作させて記録を行うが、マルウェア実行中の記録は.rr ファイルとして保存される。このファイルを本論文では「実行トレースログ」と呼ぶ。実行トレースログの取得後、解析は再生された実行トレース上で行われる (図 12)。本論文の提案方式の実装は PANDA プラグインの開発に取り組む。

図 12　record 機能による OS 上のプログラム実行の記録 (図は Windows を対象に解析した時のもの)

3. 関連研究

本章ではマルウェアが持つ特定の機能とそれらの特徴に着目した先行研究について述べる。

3.1 Panorama

Panorama[7] はテイント解析を用いてスパイウェア検出を行うシステムである。プロセスモニターを用いたマルウェア解析は API コール単体から確認できる痕跡 (例:特定のレジストリが書き換えられた) までしか収集できない。単体の API コールから確認可能な特徴だけではマルウェアと正規のアプリケーションとの区別が難しい。Yin ら [7] は保護対象データへの不正な参照とそのデータに対しマルウェアが行った処理が悪意ある振る舞いを示す特徴だと仮定している。この特徴にあたる処理を “Fundamental Trait” と定義しており、Panorama は解析者の補助ツールとしての立ち位置を取っている。また、テイント解析によって得られた解析結果にグラフによる可視化を試みている。

Panorama は保護対象プロセスであるテキストエディタや Web ブラウザに対してキーボード入力されたパスワード等の保護対象データにテイント解析を行い、検査対象のプログラムがテイントされたデータを参照した場合そのプログラムをスパイウェアと判定する。しかし評価の際の誤検知 (False positive) の理由の 1 つに、Panorama はテイントされたデータへの参照のみをスパイウェア判定の基準としていたことを述べている。また、考察にはテイントタグを付与されたデータへの参照から攻撃者の意図を読み取るルールがないことを取り挙げている。

3.2 ReFormat

ReFormat[8] はテイント解析を用いてマルウェアの暗号化関数の位置を特定する手法である。C2 サーバとマルウェアの間の暗号化通信における、受信パケットの復号化処理を行う関数の実行ファイル中の位置を特定する。

ReFormat は暗号パケット受信時にパケットの受信に用いられたバッファを監視対象としてテイントタグを設定し、以降の処理において監視対象のデータを操作した命令コードを抽出する。暗号化処理に使用される命令コードは算術命令が多いと仮定しており、計測した命令および関数呼び出しから一番多くの算術命令を発生させた関数を暗号化関数として検出する。テイント解析によって取得した命令コードに着目した解析方法である。バックドア機能を持つ実物のマルウェア検体である Agobot の暗号化された IRC 通信を用いた C2 サーバとの通信を対象に ReFormat を用いた解析手法を適用し、用いられた暗号関数と復号後のパケットデータが保存されたバッファの位置の特定に成功している。攻撃者から受け取った指令メッセージのフォーマットの復元に成功しているが、暗号化処理は算術命令が多用されるという前提で解析を行う手法であるため、正規のアプリケーションとの誤検出が発生する可能性や難読化を用いた手法により検出を回避されてしまう可能性について言及している。

出典： "Malware Attribute Enumeration and Characterization Concept Document"
The MITRE Corporation

図 13 既知と未知の検体とで似た振る舞いを分類整理

3.3 MAEC(“Malware Attribute Enumeration and Characterization”)

MAEC[9] はマルウェアの動作や振る舞い等、システム上で発現させる特徴や専門用語 (terminology) について統一された分類整理を行うための規約として定められている。MAEC 等の MITRE 社が支援する標準化された記述法へのニーズが生まれた背景には、未知の検体の対処に類似した既知の検体に関する情報を用いる (図 13) 等のモチベーションや、既存の検知手法だけでは限界が認知されていることが挙げられる。

既存の検知手法にはシグネチャ型とヒューリスティック型の 2 つの手法が主流となっている。過去に発見されたマルウェアを検知するシグネチャ型はデータベースに未登録の新種の検体や標的型攻撃用にカスタマイズされた検体に対して脆弱である。マルウェアは暗号を用いた自己書き換えや、コード圧縮やデッドコード挿入による自身のコード拡張を行うポリモーフィズムやメタモーフィズム等、シグネチャ検知を妨害するための多くの手法を用いる。

一方、ヒューリスティックベースによる検知手法はマルウェアから確認された振る舞いから悪意ある動作のパターンを検出する。これはデータベースに頼らない手法であり、新種の検体への対抗に有用だとされている。しかし、悪意ある動作の定義は実際は API コールの確認等に基づいており、正規のアプリケーション

によるものとの区別が難しいため、誤検知を発生させる要因となる。また、複数の動作が組み合わさって発現する振る舞いもあるため、悪意ある振る舞いの定義は難しい。この判定には解析者や解析者が使用したツール特有の出力結果に依存している部分が大きい。

MAEC はシグネチャ検知だけでは追いつかないほど数の増加や脅威が多様化したマルウェアに対処するために考案されており、各種解析ツールが生成したレポートから得られる情報について知識体系の分類整理に一貫性を与えることをモチベーションとしている。MAEC の課題点には悪意ある振る舞いの定義が不明瞭であることが挙げられている。API コール単位での処理の観測だけでは振る舞いを意味づけることが難しく、そのような振る舞いの定義付けに関するプロセスを補強するアイディアが必要であることが読み取れる。

4. 特徴抽出の自動化手法の提案

本章ははじめに提案手法について述べ、提案方式の設計におけるコンセプトと提案方式を用いた解析の流れについて説明する。

4.1 ファイルアクセス監視に基づくデータフロー解析による痕跡の収集による悪性機構の特定

本研究は攻撃者の意図の特定を目的とした解析の自動化手法を提案する。マルウェアによるファイルデータの処理に対してデータの流れを追跡 (Information Tracking) し、特定の悪性機構に関する情報をピンポイントで取得する。提案方式は監視対象のファイルが読み出され (read)、外部出力 (write,send) されるまでにマルウェアが適用した処理の解明作業を扱う。プロセス 1 つ 1 つを個別に精査する方式ではなく、システム全体を通し観測できるイベントから監視対象ファイルからの読み出しが発生した時に操作対象のデータに着目した解析モードへと移行する。

本研究はマルウェアの特定機能の解析にはデータのフローに着目したデータ指向型の解析が適しているという立場を取っている。従来のマルウェア解析でよく用いられる命令指向型の解析手法は、単体の API コールや 1 つ 1 つの命令コードの精査に基づいており、複数の手法が組み合わさった機能の対処には解析者の経験による勘が必要であり自動化は難しい。例えば攻撃者は、侵入検知を迂回する目的で暗号化された通信路上で情報窃取を行う等幾つかの手法を組み合わせるが、複合機能的な特徴の発見は困難な作業となる。ファイルへのアクセス自体は API コールの監視によって検知が可能であるが、読み出されたデータにどのような処理が適用されたのかは使用された関数や API 等のイベントを 1 つ 1 つ精査する必要がある。また、命令単位の監視から得られるログは膨大で手動で解析するには大変である。従って、複数の API コールの関係性を捉えるにはデータフローの解析によりデータを介した処理と処理の間の結びつきを特定することが効率的だと考えた。

提案方式はマルウェアが標的データへ適用した特定の処理におけるデータフローの観測をテイント解析を用いることにより実現する。テイント解析によって特定のデータに適用された命令コードを抽出し、抽出した命令コードに関係するプロセスや共有ライブラリ等のマルウェアが用いた手法に関する情報の取得を行う。API コール同士の処理上における結びつきを特定し、悪性機構に関する情報の抽出の自動化を行う。また、マルウェア解析はテスト用検体が発生させた攻撃シミュレーションの全体を記録した実行トレースを対象に行い、攻撃の意図の特定を試みる。

4.2 提案方式の利点

ファイル名を指定し、更にプロセス名等の事前情報を与えることにより感染端末上の特定の標的ファイルに視点を合わせた柔軟な解析を行うことができる。処理されるデータと API コールとの関係性からマルウェアの悪性機構の種類の判別へと繋がる情報を取得できる。解析から得た情報の可視化により直感的な理解を促す出力が生成される。提案方式のコンセプトを以下に列挙する。

- 自動化
  - API が参照したファイル名等の値を取得
  - I/O 処理の際にテイント解析を適用し、処理を追跡
  - テイントチェックにより悪性機構を示す痕跡を判定
  - VMI によりテイント解析で得た機械語命令から OS レベル情報を取得
- 柔軟性
  ファイル名の指定により、特定のデータに着目した解析をサポート
- 可視化
  標的ファイルからの読み出しと関係する処理についてグラフ出力

4.3 提案方式の実装

提案方式はPANDAフレームワークに組み込みが可能なプラグインであるSpaniel[10]として実装した。Spanielは標的マシン上でrecordしたマルウェアの実行トレースを対象に解析を行う。Spanielはマルウェアによる窃取や改竄等の標的となり得るファイルの指定により、そのファイルデータに適用された処理をテイント解析によって特定する。ファイルやネットワークに関するI/O処理をAPIフックで捕捉し、テイント解析用の処理を挿入する。テイントチェックによってテイントされたデータが操作対象となったネットワーク送信やファイル書き込み等を検出し、解析によって得た事実をマルウェアの悪性機構の判別に用いる。OS内で動作するプロセスやロードされた共有ライブラリ等のシステム情報をVMIで取得しセマンティクスを復元する。得られたセマンティクス情報からグラフ記述スクリプトを生成し、それを用いてグラフを出力し解析結果を可視化する。

プラグインの開発にあたってプログラム言語にC++、グラフ生成にはGraphviz[11]を用いた。また、本提案方式が用いるAPIフックとは実行トレースログの再生の中で特定のシステムコールが観測された際にコールバック関数を挿入することを指す。特定のシステムコールの呼び出し時に任意の解析用処理を挿入できる点において広義のAPIフックと同じである。

4.4 実装した機能

4.4.1 ファイル・ネットワークI/O処理のAPIフックによるテイントチェックの挿入

SpanielはAPIフックを用いてI/O処理の際にテイント解析に関する処理を挿入する。Input処理にあたる監視対象ファイルの読み込み(read)をAPIの引数に含まれているファイル名から確認した時、データの格納先のメモリ領域にテイントタグを設定して監視対象としてデータフローの追跡を開始する。Output処理であるファイル書き込みやネットワーク送信(write,sendto)が呼び出された時、出力対象のバッファがテイントされているかをテイントチェックで判定する。

テイントタグにはラベル番号という識別子が割り当てられている。Spaniel は監視対象のバッファのメモリ領域の全体に 1 つのラベル番号を与えている (例:読み出された/etc/passwd の格納先バッファに label-1 を与える)。I/O 処理に用いられるバッファはテイント解析の起点になる。API フックにより Input 処理は入力バッファにテイントタグを設定する処理が挿入され、テイント解析の開始口になる。Output 処理には出力バッファに対してテイントチェックを行う、出口検査の処理が挿入される。テイントチェックにより出力バッファ上にテイントタグの存在を確認した場合、それはマルウェアがファイルを開いた後に何らかの処理を行ったとし、監視対象のファイルが改竄もしくは窃取された痕跡と見なすことができる。このようにして得た痕跡を悪性機構の指標として用いる。

ファイルデータが格納されたメモリ領域をテイント後、メモリ領域は監視対象となりテイントされたメモリ領域に含まれているデータの操作を行った命令は tainted 命令として記録される。データ操作によって発生したテイント伝搬先のメモリもまたテイントされ、それらへのデータ操作も tainted 命令記録の対象となる。tainted 命令は仮想 CPU 上で観測された機械語命令であり粒度の細かい情報であるため、VMI を用いた OS コマンドや共有ライブラリ等の取得によってセマンティクスの復元を行う。VMI による OS レベルの情報の取得によって、その命令コードを実行したプロセス名や参照したライブラリ名の特定が可能になる。

解析結果はテキストベースの出力とテキストベースの解析結果を基に生成したグラフによる可視化結果からなる。テキスト出力からは悪性機構のカテゴリが読み取ることができる。出力される情報に以下がある。

- マルウェアが使用した API
- マルウェアが読み書きを行ったファイルの名前
- 書き込み処理に用いられた API
- 書き込み先に指定されたファイルもしくはその外部送信先
- マルウェアが標的データの処理に用いた共有ライブラリ情報
  - 暗号系の共有ライブラリを用いた形跡から暗号処理の使用が検知可能

```
digraph taintgraph{
        "<watched data>"[shape=doubleoctagon];
        "<sample file>"[shape=note];
        "<system_call>";
        "<system_call>" -> "<sample file>";
        "<system_call>" -> "<watched data>";
}
```

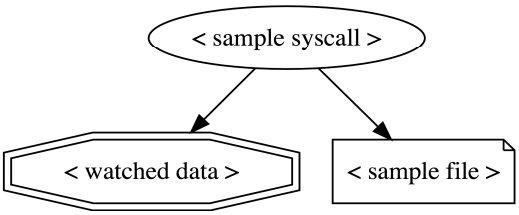


図 14 Graphviz のグラフ記述スクリプト (左) と生成したグラフ (右) の例

– ファイルデータが暗号化後に外部送信 (sendto) されたことを示す指標

4.4.2 グラフによる可視化

Spaniel は Graphviz による変換が可能なグラフ記述スクリプトの生成を行う機能を持つ (図 14)。データフローの解析で特定したマルウェアの悪意ある動作中の各機能の関係性と、マルウェアが用いたコマンドや共有ライブラリ・操作の対象になったファイル名等の情報を反映したグラフの生成を行った。

グラフ出力による可視化はマルウェアの動作の直感的な理解を促進することを意図している。グラフを構成するノードは、マルウェアやプロセス、共有ライブラリ名等の OS オブジェクトを示し、それらを繋ぐエッジはデータフローにおけるテイントタグの繋がりを示している。グラフに使用したアイコンには八角形や丸形等、解析から得られた情報の種類毎に形を使い分けており、tainted 命令に用いられた共有ライブラリや外部送信先の情報に関しては色を用いる等の工夫を行った。ファイル及びネットワークの I/O 処理に着目し、Input 処理をテイント解析の開始口とする。Output 処理時に行う出力バッファへのテイントチェックによる出口検査を、悪性機構判別における主要な箇所とする。

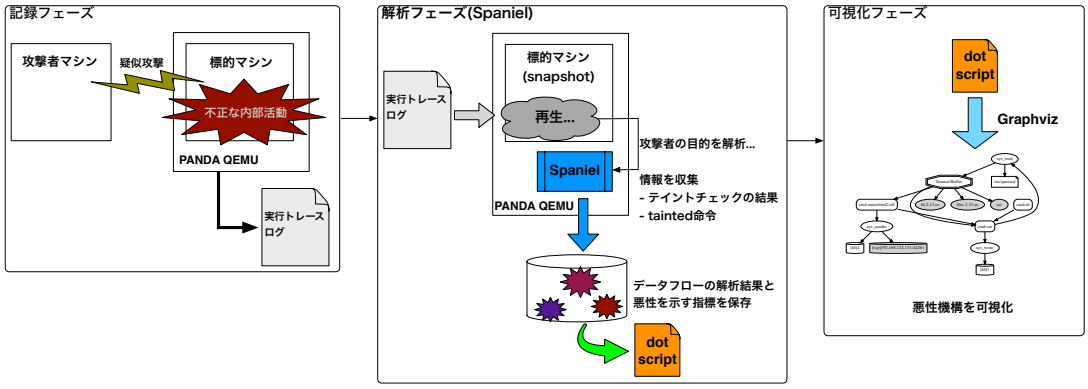


図 15　Spaniel を用いた解析の流れ

4.5　提案方式を用いたマルウェア解析の流れ

図 15 に提案方式を用いたマルウェア解析の手順を示す。解析は記録フェーズ・解析フェーズ・可視化フェーズの 3 つの段階で行われる。

4.5.1　攻撃シナリオの記録フェーズ

記録フェーズは QEMU の仮想マシン上でマルウェアを動作させ、システム内で発生した処理を実行トレースログに保存する。QEMU の record/replay 機能により実行トレースログにはマルウェアの実行による攻撃者の内部活動が記録される (図 15)。記録フェーズは解析対象のマルウェア検体が手元にあることを前提としている。

本研究が取り扱う悪性機構検出を目的としたマルウェア解析は、攻撃シナリオの解析という形で行われるため、記録フェーズは攻撃をシミュレートする間の実行トレースを記録する。記録フェーズで取得した実行トレースログは以降の解析フェーズ・可視化フェーズで使用する。

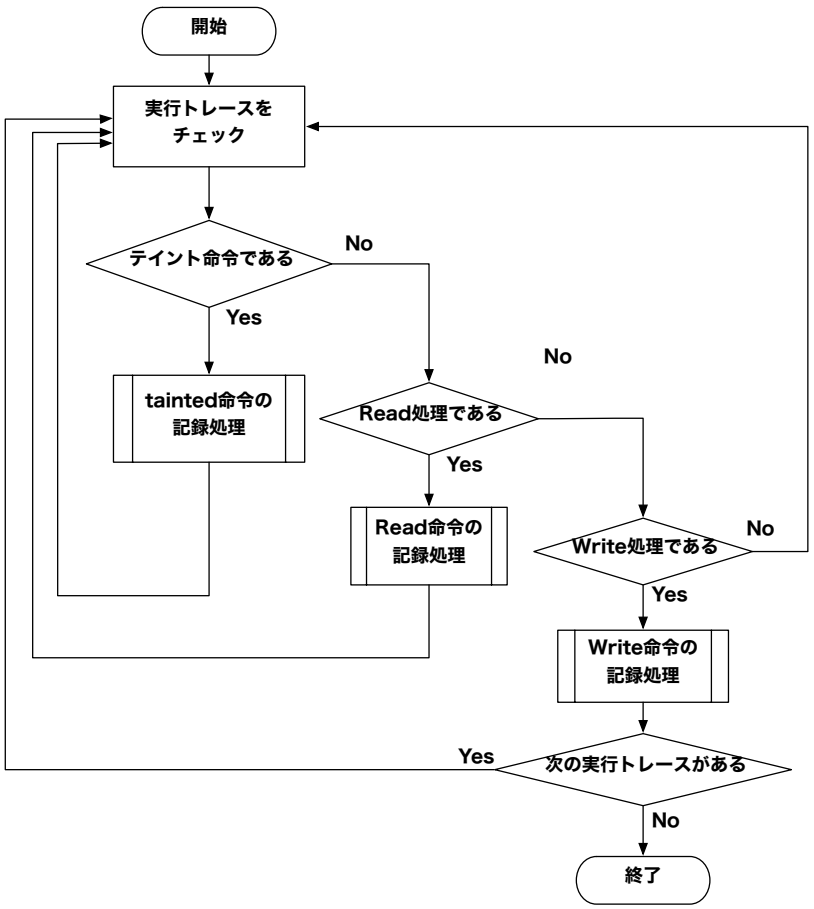


図 16 データフロー解析のフローチャート

4.5.2 悪性機構抽出における解析フェーズ

解析フェーズは実行トレースログを QEMU の replay 機能で再生し、再現された攻撃シナリオの解析を行う (図 15)。マルウェアの活動によりシステム内で実行された命令は実行トレースから取得でき、API コールの監視も可能である。更に監視対象ファイルのデータへの処理におけるデータフローをテイント解析で追跡し、マルウェアが用いたコマンドや共有ライブラリ等の OS ユーティリティ情報を取得する。

データフローの解析におけるフローチャートを図 16,17 に示す。監視対象ファ

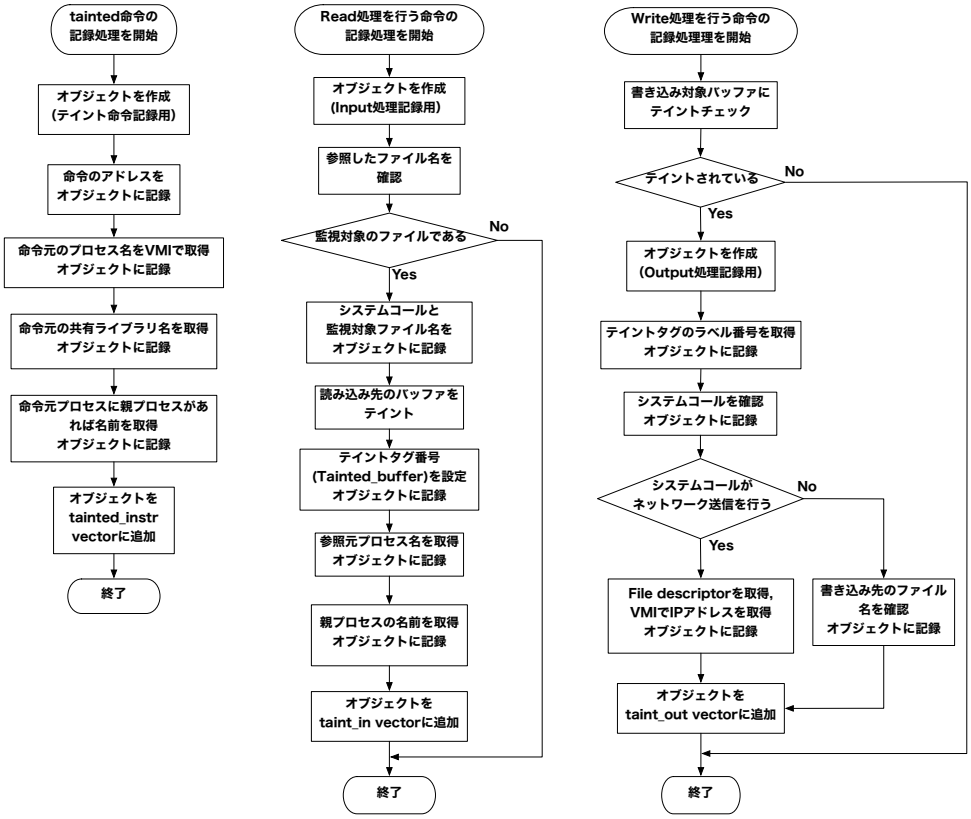


図 17　フローチャート内の関数 (図 16 内の各記録処理を分けて説明)

イルデータの読み出し (Read 処理) を API フックで検知した時、テイント解析を適用する。そしてデータの書き出し (Write 処理) を検知した時はテイントチェックを実行する。図 16 は実行トレース中の命令に応じてテイント解析の適用及びテイントチェックを実行する処理の流れである。図 17 は図 16 で用いる記録処理のフローチャートである。

解析フェーズは主に、tainted 命令を収集する処理、Read 処理時に行うテイント解析適用の処理、Write 処理時のテイントチェックの処理を中心に行う。tainted 命令から命令コードの実行元のプロセスやアクセス先のファイル等の実行中の情報を取得する。フローチャートに登場する vector とは解析時に取得した情報を保存するための配列であり、解析結果の出力やテイントタグを介したグラフ作成を

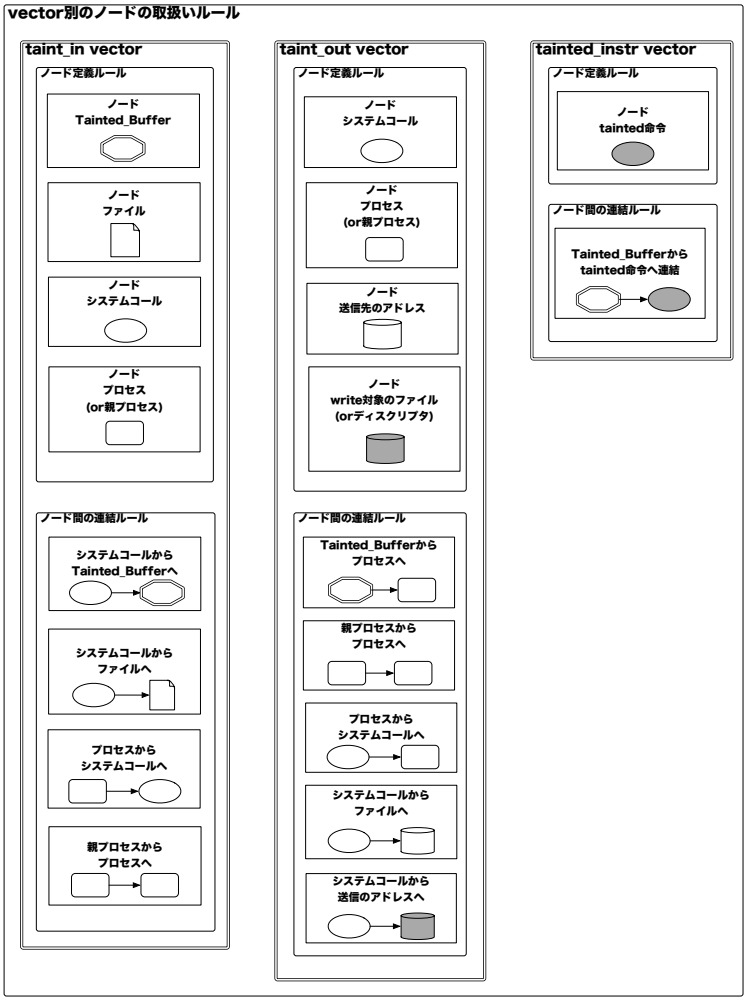


図 18 グラフ中のノードのアイコンの形状や結合に関するルール

行う際に用いる。グラフに用いる情報には親プロセス/子プロセス、呼び出された API/呼び出し元プロセス等の関係性が記述される。

4.5.3 グラフ化による可視化フェーズ

可視化フェーズは解析フェーズで得た解析結果をグラフ生成で可視化する。図 18 はグラフを構成する各 OS オブジェクト (プロセス、tainted 命令、ファイル等) に対応するアイコンを定義している。解析フェーズ (図 16) の各 vector の情報 (テ

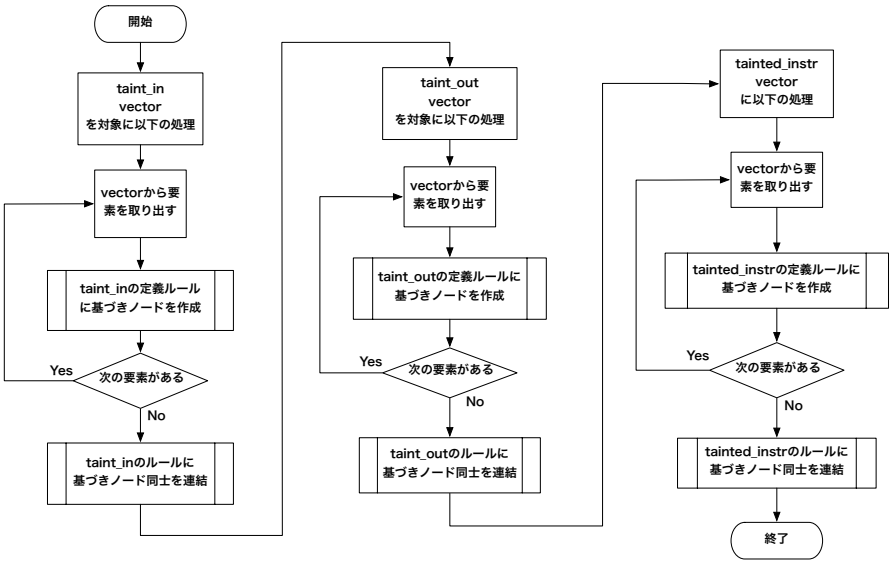


図 19 グラフ記述を定義する処理のフローチャート

イント解析の開始口、テイントされたデータの出口、tainted 命令) に関してそれぞれにグラフノードの定義ルールと接続ルールを用意している。表示する情報の種類に応じてアイコンの形と他のノードとの結合ルールを使い分け、グラフの記述方法を定義する処理（図 19）を実行している。

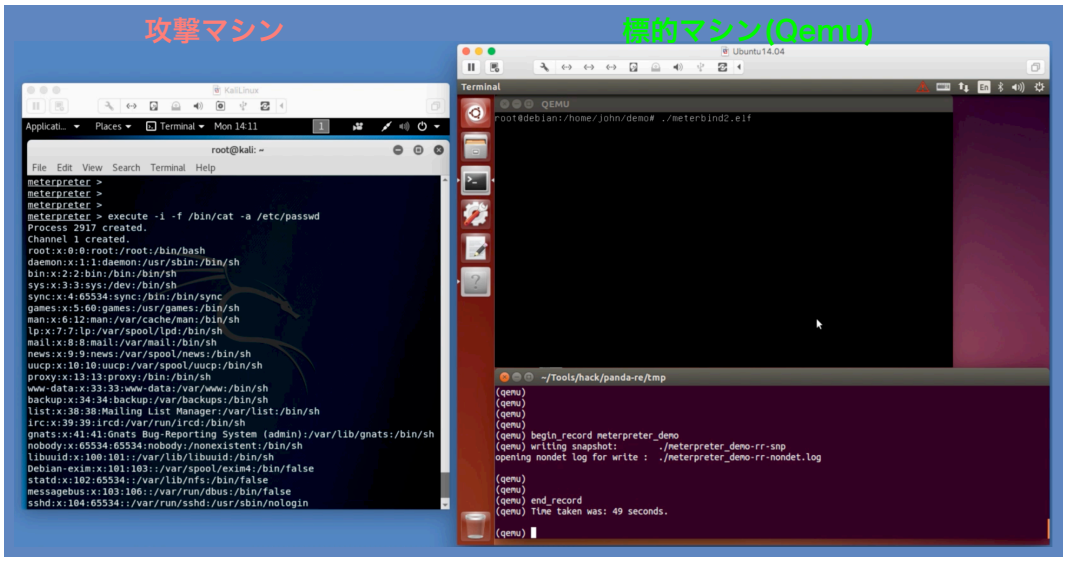


図 20　攻撃者の内部活動の実行トレースログを取得 (図は情報窃取をシミュレートしたもの)

5. ユースケースによる実験

本章は、提案方式を用いて「情報窃取」、「遠隔操作」、「ファイル暗号化」の攻撃シナリオの解析を行う。ファイルアクセス監視に基づくデータフローの解析による攻撃者の意図を特徴づける痕跡の自動抽出を行う。

実験で取り扱う攻撃シナリオの解析は 4.5 節で説明した手順で行う。攻撃者の内部活動を図 20 のようにシミュレートし、実行トレースログを記録する (図 21,22)。Spaniel による解析は図 23 に示すような実行トレースの replay の中で行う。

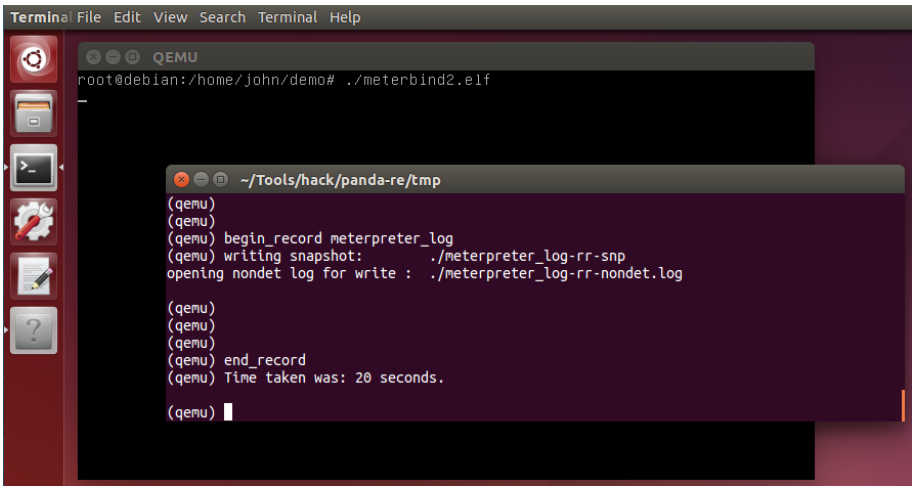


図 21 記録フェーズにおける record

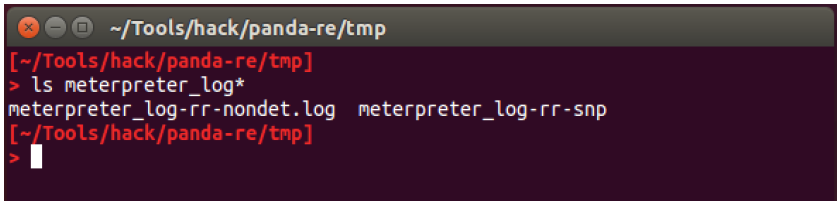


図 22 実行トレースログ (感染時のスナップショットと非決定性入力データ)

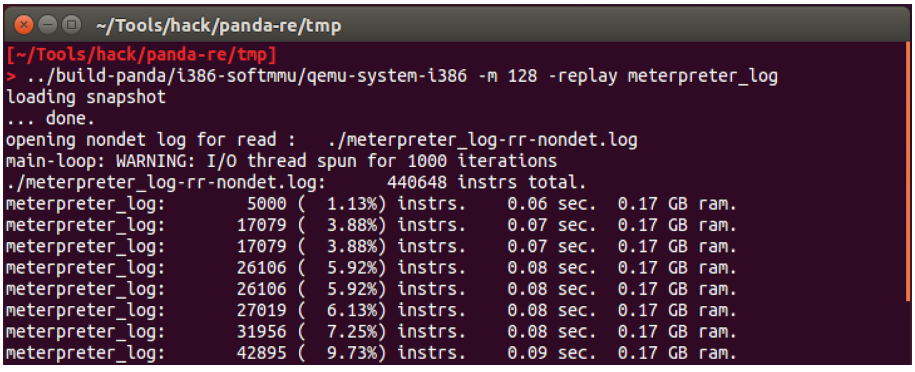


図 23 解析フェーズ時の実行トレースの再生

表 1 仮想環境内で動作する標的マシン構成

CPU	コア数:1
RAM	128MB
OS	Debian7 (3.2.88-1 i686)

表 2 解析サーバの物理マシン構成

CPU	Intel(R) Xeon(R) CPU E5-2630L v4 @ 1.80GHz (コア数:10)
RAM	62GB
OS	Ubuntu-14.04 64bit
Hypervisor	PANDA(QEMU emulator version 2.8.50)

5.1 実験環境のマシン構成

本章で述べる 3 つの攻撃シナリオの実験に同じマシン構成の環境を用いる。実験環境はマルウェアの活動を記録する用の標的マシンと、実行トレースの解析を行う用のサーバとで構成される。表 1 は QEMU の仮想環境内で動作する標的マシンの構成である。表 2 は実行トレースログの解析を行うサーバの物理マシン構成である。標的マシン内の処理を記録した実行トレースログを解析サーバに転送し、PANDA プラグインとして実装した Spaniel を用いて解析を行う。

PANDA フレームワークがメモリ中のテイントタグ情報の管理に用いるシャドウメモリの確保にはゲスト OS の RAM × 16 の容量を必要とした。TaintData 構造体の 16 バイト分のデータが仮想マシンのメモリバイトのサイズ分確保されるため、本実験が行う表 1 の仮想環境を対象にした解析には最低でも 2,147,483,648 バイトが必要とされる。

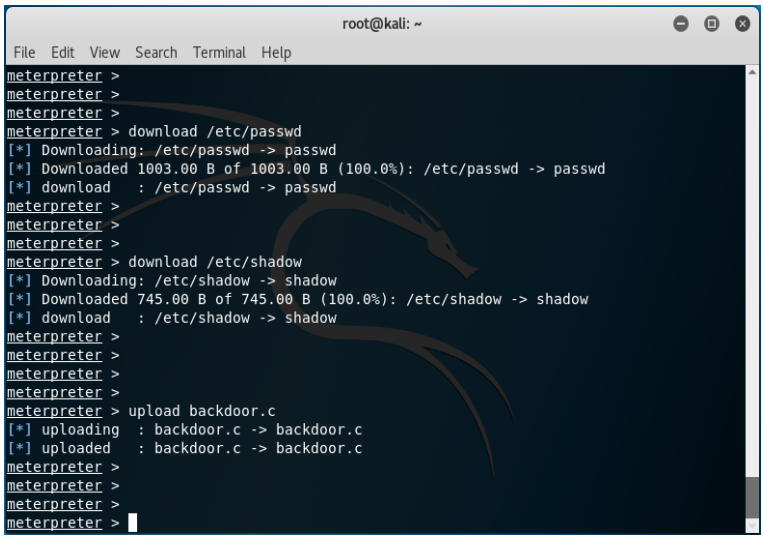


図 24　meterpreter による侵入後の内部活動

5.2　使用したテスト検体

5.2.1　情報窃取・遠隔操作の攻撃シナリオに用いたテスト検体 (meterpreter)

情報窃取や遠隔操作による攻撃を実行するテスト検体に meterpreter[12] を使用する。meterpreter は侵入テスト用のツールである Metasploit Framework[13] で生成可能な実行形式のシェルコードである。download コマンドや upload コマンド等の攻撃者が内部活動に用いる機能を備えている (図 24)。本実験では、meterpreter(ファイル名:meterbind2.elf) の実行後にバインドシェルから侵入した攻撃者による cat コマンドを用いたファイル窃取の攻撃シナリオを解析する。

5.2.2　暗号化処理の攻撃シナリオに用いたテスト検体 (OpenSSL)

暗号化処理を行うテスト検体として OpenSSL コマンド [14] を使用した。OpenSSL コマンド自体はマルウェアではないが、本実験中ではファイル暗号化を行うためのサンプルとして用いる。OpenSSL コマンド (ファイル名:openssl) が実行した暗号化処理を対象に実行トレースログを記録する。攻撃シナリオにおける暗号化は任意のテキストファイルに対して行われる。

```
3046  getpid()                          = 3046
3046  clock_gettime(CLOCK_MONOTONIC, {4454, 469150769}) = 0
3046  select(32, [0 4], [], NULL, {0, 166803}) = 1 (in [4], left {0, 166770})
3046  clock_gettime(CLOCK_MONOTONIC, {4454, 470466446}) = 0
3046  read(4, "\211m\327\230\343X\25\251\24-\225\31\"\372}?.\313\"\216\211m\327\230\211m\327\2\211m\327\230"..., 65535) = 178
3046  read(4, 0xbfeb4c81, 65535)        = -1 EAGAIN (Resource temporarily unavailable)
3046  open("/etc/passwd", O_RDONLY|O_LARGEFILE) = 5
3046  sendto(4, ")\20h\372C%\252\313\264P*{\202\207\302]\216\266\235\354)\20h\372)\20h\201)\20h\373"..., 147, 0, NULL, 0) = 1
47
3046  sendto(4, "", 0, 0, NULL, 0)      = 0
3046  getpid()                          = 3046
3046  clock_gettime(CLOCK_MONOTONIC, {4454, 486516503}) = 0
3046  select(32, [0 4], [], NULL, {0, 149437}) = 1 (in [4], left {0, 149399})
3046  clock_gettime(CLOCK_MONOTONIC, {4454, 487712174}) = 0
```

図 25　Linux における strace コマンドを用いたシステムコール観測

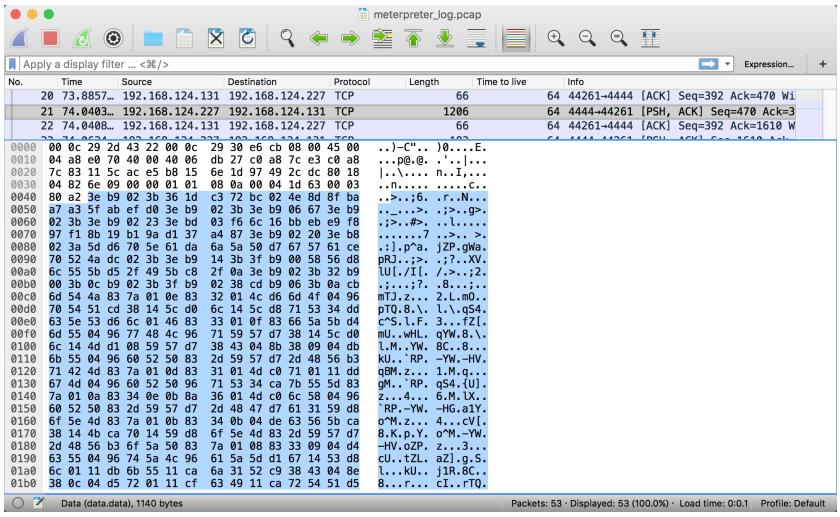


図 26　情報窃取を観測した際の暗号化パケット

5.3　情報窃取処理 (Exfiltration) の特徴抽出

仮想環境の Linux 上で実行された meterpreter(表 3) による不正活動のシナリオの中に、情報窃取の悪性機構が組み込まれていることを特定する。

マルウェアによるファイル窃取を解析する際に着目する点として、標的ファイルへのアクセスの監視が挙げられる。しかし、特定のプロセスがファイルがオープンしたからといってそれがファイルの外部送信を意図した動作とは限らない。設定ファイルやログファイルの閲覧等、ファイル操作は正規のプロセスも実行する処理であり、システム内で多く実行される。このため、操作対象のファイルが通常の動作ではアクセスされることが稀な場合を除き、ファイルアクセス自体を

表 3 Exfiltration の検証

検体ファイル	meterbind2.elf
窃取対象ファイル	/etc/passwd
テイントチェック対象 API	read(), sendto()

不正活動と見なすには根拠に欠ける。

図 25 は meterpreter が実行したシステムコールを strace コマンド (Windows におけるプロセスモニターと似た機能を持つ Linux ユーティリティ) で収集した際の出力である。meterpreter がプログラムの起動からファイル窃取に至るまでの間に発行したシステムコールの数は軽く 1000 を超える。ファイル窃取時のネットワーク活動をパケットモニタで監視した際 (図 26)、マルウェアが送信したデータは暗号化されているため、パケットを一見しただけではファイル窃取と断定するのは難しい。図 26 は認証情報を持つ passwd ファイルを meterpreter が窃取した際のパケット画面であるが、データのダンプには元の passwd ファイルのテキストデータの特徴を一切残していない。マルウェアはデータ送信に暗号等を用いた難読化処理を行うため、動作単位での事象を観測したのみでは送信されたデータとアクセスされたファイルデータを直接紐付けることはできない。

ファイル窃取を確認するには、マルウェアがネットワーク送信時にバッファに格納したデータが標的ファイルのデータが難読化されたものである可能性も考慮しなければならない。通常の解析において異なる API をまたいだデータへの処理を捉えるには手動による複雑な作業が要求される。

提案方式による本実験ではテイント解析を用いたデータのフローを追跡し、メモリ中に伝搬したテイントタグからファイル窃取の痕跡を自動収集する。

5.3.1 仮説 1「窃取対象ファイルのデータは難読化後にネットワーク API の送信バッファに格納される」

情報窃取の対象となったファイルデータは読み出された (read) 後に暗号や圧縮による難読化が適用される可能性があるものの、最終的にはネットワーク送信

API(sendto) の引数が指すバッファに格納されると考えられる。従って、データを介した処理の中で 2 つの API コールが関係していることを示す必要がある。

Spaniel を用いた解析はファイル読み出し API(read) が参照した標的ファイルデータへのテイント解析により、処理中のデータの流れを追跡することでファイル窃取の痕跡の収集を試みる。ファイルから読み出されたデータは最終的には外部送信を行う API(sendto) に受け渡されるため、sendto が呼び出された際に送信バッファを対象にテイントチェックを実行する。送信バッファのメモリ位置にテイントタグが登録されている時、この事実をファイル窃取の指標として扱う。本実験は読み出されたデータへの処理を追跡によって外部送信の対象のデータをファイルデータと紐付ける痕跡が得られるという前提で取り組む。

5.3.2 実験 1「窃取対象ファイルデータをテイント解析し、外部送信バッファを対象にテイントチェックを行う」

PANDA プラグインとして実装した Spaniel は Linux のシステムコール呼び出し時にコールバック関数を挿入する形で API フックを行う。本実験では標的ファイルの読み出し (read) を API フックし、read 先のバッファを監視対象のメモリとしてテイント解析を実行する。テイント解析で標的ファイルデータへの処理がテイント伝搬を通して観測する。

sendto や write 等のネットワーク送信を行う API が呼ばれる毎に送信バッファのテイントチェックを実行する。テイントチェックはメモリ中に伝搬したテイントタグを検出する処理である。送信バッファがテイントされていた場合、これをマルウェアがデータを読み出した後に外部送信した指標とする。

図 27 に標的ファイルから読み込まれたデータが難読化の後に外部送信される様子と Spaniel によるファイル窃取の検出の流れを図示している。標的ファイルから読み出されたデータにテイント解析を適用し、ネットワーク送信時に API フックで挿入されるテイントチェックの判定でファイル窃取を検出する。本実験は meterpreter が/etc/passwd ファイルの呼び出しを行った際にテイント解析を適用し、上記で説明した流れで解析を行う。送信バッファへのテイントチェック結果をマルウェアが情報窃取の悪性機構を含むかどうかの判定基準とする。

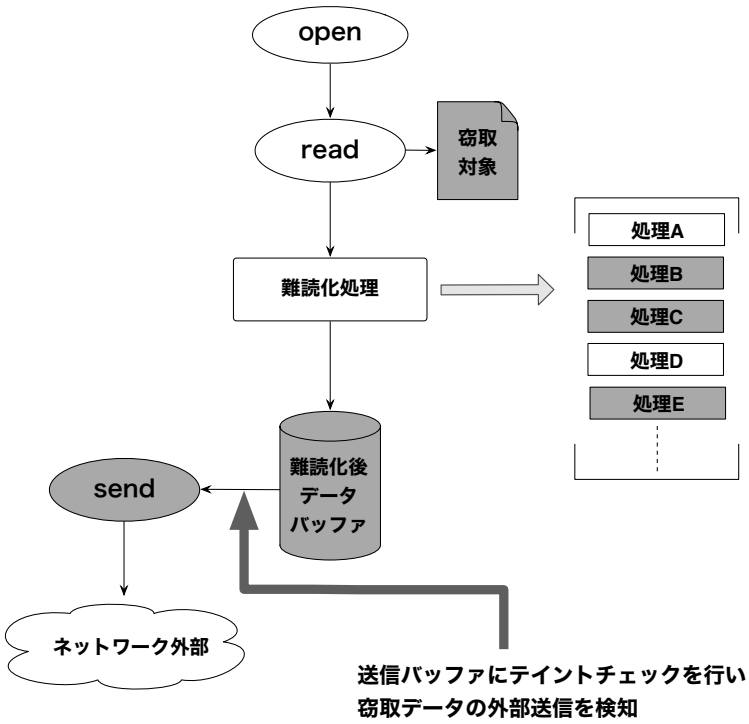


図 27 テイントチェックによるファイル窃取の検出

5.3.3 実験結果

図 28 に Spaniel の解析フェーズの出力結果を示す。出力結果は read によってファイルから読み出されたデータが何らかの処理を適用された後に外部送信されたことを示している。ネットワーク送信 API の sendto が呼び出された際に送信バッファにテイントタグが伝搬していることを確認した。read によって窃取対象ファイルから読み出されたデータはテイントタグ (label-1) で識別される。sendto の引数のバッファが同じテイントタグ (label-1) で紐付いていることが確認でき、テイントタグを介した read と sendto の関係性を示している。

また、図 29 に可視化フェーズの出力結果を示す。図 29 は情報窃取の活動をグラフ形式で可視化したものである。meterpreter が窃取対象データの/etc/passwd からデータを読み出した (read) 後に外部のホスト (IP アドレス 192.168.124.131) に送信されたことを示している。プロセスの親子関係やその他共有ライブラリの関

```
=============
taint_in:
        label-1
                taint_resource: /etc/passwd
                sys_call: sys_read
                        rr_instr: 3040551
                command: cat
                        parent_cmd: meterbind2.elf
                        parent_cmd: sh
                size: 1003

=============
taint_out:
        cmd: cat
                        parent_cmd: meterbind2.elf
                        parent_cmd: sh
        syscall: sys_write
                rr_instr: 3045056
                resource: (Isnull fd:1)
                                label-1, /etc/passwd compute:31851
taint_out:
        cmd: meterbind2.elf
        syscall: sys_sendto
                rr_instr: 3194324
                resource: (Isnull fd:4)
                        candidate_dst_host: 192.168.124.131:44261
                                label-1, /etc/passwd compute:31851
```

図 28　ファイル窃取の解析結果

係性についてはノードとエッジで結んでいる。本実験ではファイルやネットワークに関する I/O 処理をテイント解析の開始口、出口検査のポイントに見立てた上で情報窃取の指標を検出できることを確認した。

5.4 遠隔操作処理 (Command and Control) の特徴抽出

遠隔操作のシナリオはテスト用検体に meterpreter(表 3) を使用し、情報窃取のユースケースと平行で行われる。5.3 節で用いた攻撃シナリオと同じ実行トレースログの解析に取り組む。遠隔操作の解析では、マルウェアが不正活動の中で OS ユーティリティを用いた痕跡を悪性機構の指標とする。

攻撃者による標的ファイル (/etc/passwd) の読み出し以降の処理をテイント解析で追跡し、OS ユーティリティが使用された痕跡の収集を行う。tainted 命令に OS ユーティリティから実行されたコードが含まれていた場合、これを遠隔操作の指標とする。情報窃取の悪性機構は送信バッファへのテイントチェック結果か

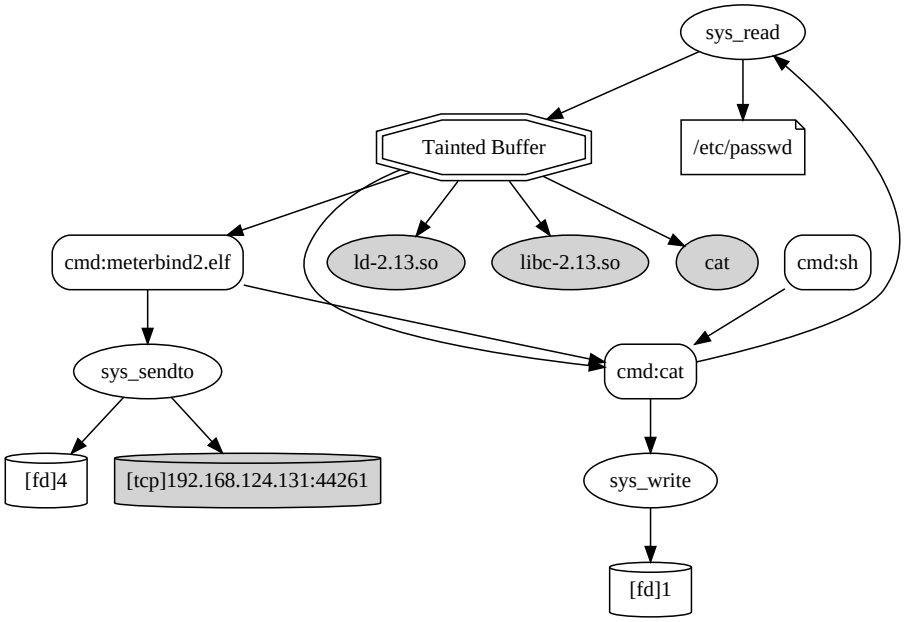


図 29　meterpreter が実行した情報窃取処理をグラフ形式で可視化

ら判定したが、遠隔操作の検出は tainted 命令の検査によって得た情報を判断材料とする。

5.4.1 仮説 2「観測された tainted 命令を発行したプロセス名から OS コマンドを特定できる」

本実験はマルウェアが不正活動に OS ユーティリティを使用することを前提としている。攻撃者が遠隔操作上で行うデータの窃取や改竄等の不正活動は感染マシン上の OS ユーティリティを用いて行う可能性が高い。OS ユーティリティを用いた不正活動は正規のアプリケーションの動作との区別が難しいため、侵入検知システムを回避する手段として攻撃者に悪用される。最近のマルウェアには不正活動に OS ユーティリティを用いる検体が多数存在し、Windows マルウェアの中には Powershell を悪用する検体が存在する [15]。

正規の操作とマルウェアによる操作とを区別するために、本実験では標的デー

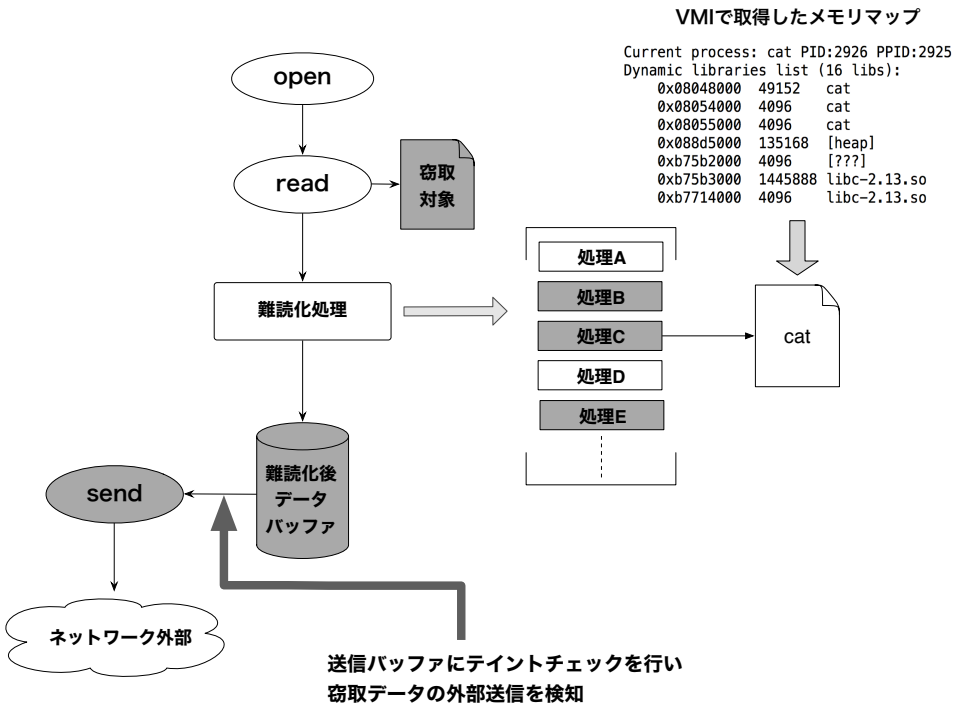


図 30　ファイル窃取に使用した OS コマンドの特定

タを対象にテイント解析を行う。標的データへのテイント解析で得た tainted 命令が OS ユーティリティから実行された命令コードであるとき、これを遠隔操作の指標に利用できると考えた。

5.4.2　実験 2「マルウェアによって読み込まれたファイルデータをテイント解析」

攻撃者はバインドシェル経由で実行した/etc/passwd からのデータ読み出しの解析を行うため、read が観測された際に読み込み先バッファを対象にテイント解析を適用して tainted 命令を抽出する。

抽出した tainted 命令に OS ユーティリティが実行したコードがないかを確認する。tainted 命令は粒度が細かい情報であるため、VMI で取得したメモリマップから命令コードのプログラムファイル名を特定して OS ユーティティの痕跡を取得する。VMI を用いた OS ユーティリティの痕跡の取得は図 30 のように行う。

tainted 命令から得た OS ユーティリティに関する情報を遠隔操作の悪性機構の指標として扱う。

5.4.3 実験結果

情報窃取の実験と同じく、テキストベースの出力が図 28 であり、可視化による出力結果は図 29 である。標的ファイルへのテイント解析から得た tainted 命令の検査により、攻撃者が用いた手段が cat コマンドであることを検出できた。プロセスの親子関係については VMI のプロセス表から参照を行い、cat コマンドが meterpreter によって呼び出されたことを確認した。結果、攻撃者が侵入後のファイル窃取の手段に cat コマンドを用いたことを特定できた。OS ユーティリティを用いた遠隔操作の痕跡と考えることができる。マルウェアの不正活動が標的ファイルへの処理と紐づいた場合、不正活動の指標をテイント解析によって自動的に特定できることを示した。

表 4 Encryption の検証

検体ファイル	openssl
暗号化対象ファイル	cryptme.txt
テイントチェック対象 API	read(), write()

5.5 暗号化処理 (File Encryption) の特徴抽出

標的マシン上で実行された処理に暗号化処理の悪性機構が含まれているかを特定する。openssl が実行した任意のテキストファイル (cryptme.txt) に対する AES 暗号化処理 (aes-cbc-256) を攻撃シナリオとして用いる (表 4)。

5.5.1 仮説 3「tainted 命令から暗号化処理に使用された共有ライブラリを特定できる」

マルウェアが行う暗号化処理は、ランサムウェアに見られるファイルシステムの破壊や C2 サーバとやり取りする通信内容を隠蔽する等の目的が背後にある。暗号を悪用するマルウェアの解析のゴールには、マルウェアが使用した暗号アルゴリズムの種類や暗号関数の位置特定、または実行時に使用される暗号鍵の特定が挙げられる [16]。

本実験は暗号化対象データへのテイント解析で抽出した tainted 命令には暗号化処理を特徴づける指標が含まれているという立場を取る。著者らが行った実験 [17] では、openssl コマンドが行った暗号化処理の対象のデータにテイント解析を適用してテイント伝搬を観測したところ、暗号化ライブラリの libcrypto.so 内の関数が tainted 命令を最も多くを発生させたことを確認した。標的データへのテイント解析で得た命令コードが暗号化を行う共有ライブラリのものであった時、これを暗号化の実行の指標とする。

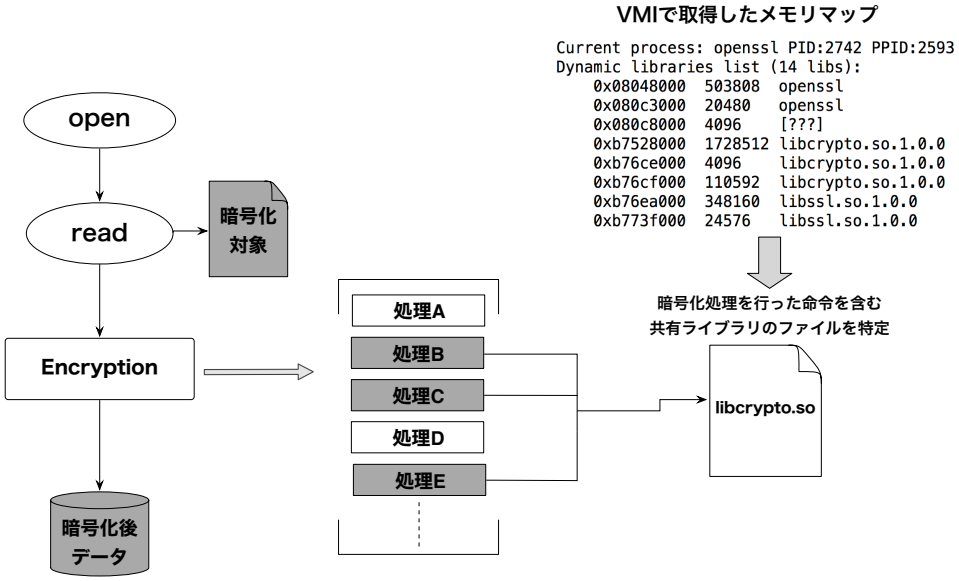


図 31 VMI によるメモリ空間参照による共有ライブラリの特定

5.5.2 実験 3「暗号化対象のファイルデータをテイント解析し、テイントされた命令コードを抽出」

表 4 中より、ファイルアクセスに関する API(read,write) への監視を軸にテイント解析を行う。標的ファイル (cryptme.txt) からの読み出し (read) 以降の処理にテイント解析を適用し、テイントされた出力バッファの書き出し (write) 処理が発生するまでの間に記録した tainted 命令の検査を行う。

Spaniel が行う解析は tainted 命令の検査から得た共有ライブラリに暗号に関するものがあることを暗号化の指標と判断する。tainted 命令の参照元の共有ライブラリファイル情報を VMI を用いてメモリマップから取得する。VMI による tainted 命令から共有ライブラリ名への参照は図 31 のように行う。

5.5.3 実験結果

解析フェーズの出力結果を図 32 に示す。暗号化対象のファイルデータ (cryptme.txt) が計算処理を経た後に excypted.txt というファイルに書き込まれたことを出力バッ

```
=============
taint_in:
        label-1
                taint_resource: /home/john/tmp/cryptme.txt
                sys_call: sys_read
                        rr_instr: 10925499
                command: openssl
                        parent_cmd: bash
                size: 36

=============
taint_out:
        cmd: openssl
                        parent_cmd: bash
        syscall: sys_write
                rr_instr: 10945738
                resource: /home/john/tmp/encrypted.txt
                                label-1, /home/john/tmp/cryptme.txt compute:38912
```

図 32　OpenSSL が行った暗号化処理の解析結果

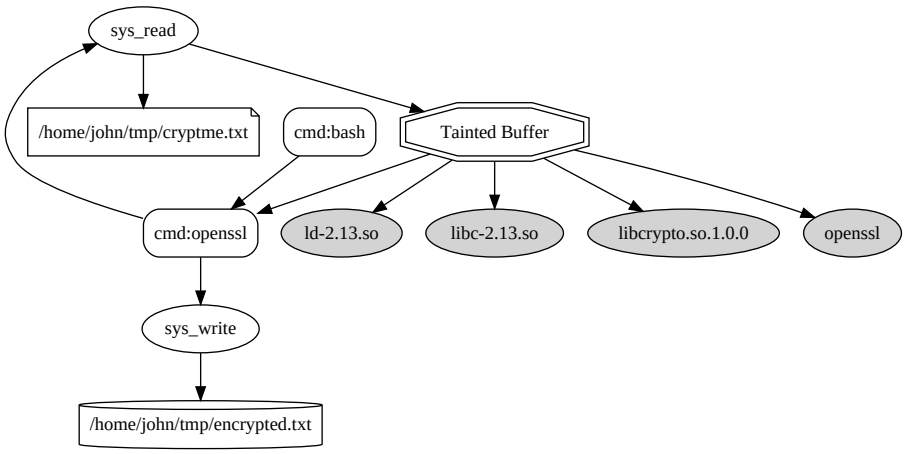


図 33　OpenSSL が実行した暗号化処理をグラフ形式で可視化

ファへのテイントチェックが検出した結果である。判明した共有ライブラリには libcrypto.so 等の暗号化に用いられるものが含まれており、テイント伝搬から暗号化の使用の痕跡を取得できた。

可視化フェーズで生成した図 33 のグラフは実行された API や共有ライブラリ間のテイントタグを介した関係性を示している。コマンド等の OS ユーティリティを悪用して暗号化を行うマルウェアの解析において、マルウェアが用いた共有ライ

ブラリの情報は攻撃者の意図を理解する上での助けとなる。本実験では標的ファイルへのテイント解析によって暗号等の特定の機構に関する情報を自動取得できることを示した。

6. 評価

本章は提案方式の評価を行う。6.1 節は提案方式による攻撃者の内部活動モデル一覧への悪性機構解析の調査結果を述べる。6.2 節は別の評価方法についての検討を述べる。

6.1 脅威モデルへの解析能力

攻撃モデル一覧をまとめた ATT&CK マトリクス [5] における、提案方式を用いた悪性機構解析の対応可能な範囲について評価を行った。ATT&CK マトリクスには MITRE 社が分類したサイバーキルチェーンにおける攻撃者の内部活動のモデルが列挙されている。

悪性機構の指標の判定基準に用いた条件について以下に述べる。

条件 1 外部出力 API が呼ばれた際の出力バッファがテイントされていること

条件 2 tainted 命令から参照した共有ライブラリ名に機能推定が可能な情報を含んでいること

条件 3 tainted 命令から参照した OS コマンドに機能推定が可能な情報を含んでいること

上記の条件で攻撃の指標の特定が可能な脅威モデルの調査を行った結果を表 5 に示す。自動化が可能な脅威モデルについては緑色を、そのモデルと同様のパターンを持つものを黄色で強調表示した。攻撃マトリクスのカバレッジの集計 (表 6) より、本提案方式のファイル名を基にした検出手法で解析可能な数は 95 個中 25 個となった。ファイルアクセスの監視に基づいたテイント解析によるマルウェア解析は、マトリクス中の 26%の攻撃について、それぞれの悪性機構の指標を自動抽出できることを示した。

情報窃取や暗号化処理の解析の実験で用いた手法で条件 1 に該当する指標を持つ攻撃モデルを検出することができた。ライブラリ名に暗号系ライブラリの名前が含まれていた場合、暗号に関係する攻撃モデルとして条件 2 から検出可能とな

Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Execution	Collection	Exfiltration	Command and Control
.bash_profile and .bashrc	Exploitation of Vulnerability	Binary Padding	Bash History	Account Discovery	Application Deployment Software	Command-Line Interface	Audio Capture	Automated Exfiltration	Commonly Used Port
Bootkit	Setuid and Setgid	Clear Command History	Brute Force	File and Directory Discovery	Exploitation of Vulnerability	Graphical User Interface	Automated Collection	Data Compressed	Communication Through Removal Media
Cron Job	Sudo	Disabling Security Tools	Create Account	Network Service Scanning	Remote File Copy	Scripting	Clipboard Data	Data Encrypted	Connection Proxy
Hidden Files and Directories	Valid Accounts	Exploitation of Vulnerability	Credentials in Files	Permission Groups Discovery	Remote Services	Source	Data Staged	Data Transfer Size Limits	Custom Command and Control Protocol
Rc.common	Web Shell	File Deletion	Exploitation of Vulnerability	Process Discovery	Third-party Software	Space after Filename	Data from Local System	Exfiltration Over Alternative Protocol	Custom Cryptographic Protocol
Redundant Access		HISTCONTROL	Input Capture	Remote System Discovery		Third-party Software	Data from Network Shared Drive	Exfiltration Over Command and Control Channel	Data Encoding
Trap		Hidden Files and Directories	Network Sniffing	System Information Discovery		Trap	Data from Removal Media	Exfiltration Over Other Network Medium	Data Obfuscation
Valid Accounts		Indicator Removal from Tools	Private Keys	System Network Configuration Discovery			Input Capture	Exfiltration Over Physical Medium	Fallback Channels
Web Shell		Indicator Removal on Host	Two-Factor Authentication Interception	System Network Connections Discovery			Screen Capture	Scheduled Transfer	Multi-Stage Channels
		Install Root Certificate		System Owner/User Discovery					Multiband Communication
		Masquerading							Multilayer Encryption
		Redundant Access							Remote File Copy
		Scripting							Standard Application Layer Protocol
		Space after Filename							Standard Cryptographic Protocol
		Timestamp							Standard Non-Application Layer Protocol
		Valid Accounts							Uncommonly Used Port
									Web Service

https://attack.mitre.org/wiki/Linux_Technique_Matrix 最終アクセス(2017/12/22)

実証済み	実証済みと同様に対処可能なもの

表 5 ATT&CK 攻撃マトリクス (Linux)

る。条件 3 は tainted 命令から取得した OS コマンド情報 (例:chmod コマンドを用いたアクセス権限の変更) から特定の機能の指標として確認することができた。表 6 より、対応が可能な範囲は暗号化やデータ転送を行う “Command and Control” 項目やシステム設定ファイルの書き換えによって達成される “Persistence” 項目が多い。これはマルウェアによるファイル処理に着目したテイントチェックと tainted 命令の解析に基づいているからである。

表 5 のカバレッジの範囲を広げる案としてテイント解析の対象のデータソース

キルチェーンのカテゴリ	モデル数	本提案で対処可能
Persistence (潜伏)	9	4
Privilege Escalation (権限昇格)	4	1
Defense Evasion (防御回避)	16	4
Credential Access (認証情報の取得)	9	2
Discovery (発見)	10	1
Lateral Movement (内部活動)	5	1
Execution (実行)	7	2
Collection (収集)	9	1
Exfiltration (窃取)	9	3
Command and Control (遠隔操作)	17	6
合計	**95**	**25**

表 6 キルチェーンのカバレッジ

(本研究ではファイル名) の種類を増やすことが挙げられる。提案方式はツールとしての使用性を考慮し、OS 上の実ファイルの名前を指定することでテイント解析対象のメモリ領域を設定していた。メモリ上に存在するデータはテイント解析の対象範囲内であるため、ファイル名以外のデータソースを検討する場合は監視対象データを一意に特定できる OS 側の識別子 (レジスタや API の名前等) を事前に設定する必要がある。

なお、ATT&CK は本論文執筆時点には更新されているため、比較のため付録 A の表 8 に最新版の ATT&CK マトリクスを掲載する。

6.2 その他の評価方法の検討

別の評価方法として、テストユーザーに提案方式を用いてマルウェア解析を行って貰いアンケート等のフィードバックから得た結果から有効性を評価する被験者実験が挙げられる。

Yakdan ら [18] は可読性の高いコードを生成するデコンパイラを開発し、解析ツールのユーザビリティの向上によってマルウェア解析者の支援を試みた研究を行っている。ユーザーへの被験者実験を採用した初のリバースエンジニアリング研究であり、ユーザーテストは難読化されたマルウェアのバイナリを配布された被験者達が提案ツールを用いてマルウェアの機能の分析を行う形式を取っている。マルウェア解析を終えた被験者達にマルウェアの機能や攻撃者の目的に関する理解度テストを受けて貰いテストの正答率を基に評価を行っている。既存のデコンパイラを用いた場合との正答率の比較から提案方式の有効性を示している。

本研究の評価に被験者実験を行う場合、テストユーザーにマルウェアの不正アクティビティの実行トレースログを配布して悪性機構のカテゴリーの特定やその根拠となる悪性機構の指標に関する情報を提示できるかといった評価項目が考えられる。

7. 議論

本章は提案方式の制約及び課題点、位置づけについて既存の解析手法との比較を通して説明し、さらに応用による展望について言及する。また、評価に用いたATT&CK に関するトピックと公開されているデータの分析から得た考察について述べる。

7.1 提案方式の制約及び課題点

7.1.1 提案方式に対するマルウェアによる解析妨害への考慮について

本研究ではマルウェアは Anti-VM による解析妨害は行わないという前提で実験を進めていた。マルウェアは Anti-VM 手法によって自身が仮想環境で動作していることを察知すると動作の変更を行い解析を困難にする。本研究では振る舞いが既知のテスト用検体を使用していたが、実際のマルウェアを対象に取り組む場合は Anti-VM の対処について考慮する必要がある。提案方式は QEMU の record/replay 機能を用いてマルウェアの動作の一部始終を観測することを前提としているため、マルウェアが QEMU の存在を検知して動作を停止する等の妨害を行った場合、解析に十分なデータが得られなない可能性がある。

Anti-VM は QEMU に限った話ではなく仮想環境を用いたマルウェア解析において悩まされる問題であり、多くの Anti-VM 手法が存在する [19]。特定の機械語命令の実行に対して仮想環境上と実機上とで異なる値が観測された事実を元に仮想環境を検知される例がある。また、特定のレジストリやファイルにハイパーバイザ固有の文字列が含まれている事実から検知される例もある。QEMU は仮想ハードディスク名に “QEMU HARDDISK” という文字列が含まれており、これは dmesg コマンド等を用いてカーネルが生成したログを参照することで確認可能である。文字列情報は仮想環境を検出する上での指標となる。ハードディスク以外の仮想デバイスにも仮想環境の指標が存在し、例えばネットワークデバイスは割り当てられた MAC アドレスにハイパーバイザ毎に固有のプレフィックスを持っていることが検知の指標となる。QEMU 等のエミュレーション技術を用いた仮想環境の解析妨害手法には、プロセッサ固有のレジスタへのアクセスによって実機

との挙動の差を指標に用いたりエミュレーションを行うためのバイナリ変換による速度低下を指標とする等のエミュレーションの性質や不備を突いたものがある。

Anti-VM 手法の全てに対処するのは難しいかもしれないがハイパーバイザ固有の文字列やプロセス名等のシグネチャを極力システム上に残さない等の最低限の配慮は必要である。例えば仮想ハードウェアの導入による QEMU 固有の文字列はソースコード内でハードコードされているため、これを別の値へ変更して再コンパイルしてから使用する等の手法が考えられる [20]。また、エミュレータは実機が行う処理を再現するため挙動における実機との差異をマルウェアに検出される可能性は前に述べたが、エミュレーションと実機の両方を組み合わせた解析環境が提案されている [21]。Anti-VM に関しては単体の解析環境だけでなく他のハイパーバイザや解析ツールの組み合わせによって対処することが最善であると考えられる。

7.1.2 マルウェア解析から得た分析結果の利用方法について

本研究で取り扱ったアプローチは検体が手元にあることを前提とするマルウェア解析であり、実環境での不正活動のリアルタイム検知ではない。解析結果を実際のシステムセキュリティの向上に用いる方法についても検討の余地がある。提案方式はマルウェア実行時のデータフローの側面から詳細な分析を行い、実行したシステムコールや操作対象のデータの関係性についての情報を取得できる。例えば、こうした情報をアンチウイルスや IDS のシグネチャとして利用する等の使い道が期待できる。実際の侵入検知に活かすには QEMU ベースの解析プラットフォームから得た分析結果をどのように実システム環境に橋渡しするかが課題になる。

7.2 既存の解析ツールによるマルウェア解析と提案方式との比較

マルウェア解析でよく用いられるプロセスモニタ、パケットモニタ、デバッガについて説明し、これらとの比較による提案方式の位置づけについて述べる。

7.2.1 プロセスモニタ

プロセスモニタはプロセスが実行した処理 (ファイルシステム、レジストリ、及びスレッドの活動) をリアルタイムで表示するツールであり、マルウェア解析によく使用される [22]。プロセスモニタは Windows のトラブルシューティングツールとして用いられる。Linux の管理用の監視ツールには、システムコールの監視を担当するものに strace コマンドがある。プロセスモニタによって記録されるシステムイベントは何千にも及ぶため、アナリストは解析に必要な情報をフィルタ等を用いて効率良く抽出作業を行う必要がある。マルウェア解析における確認事項には以下のような項目がある。

- ネットワーク活動における通信先に関する情報
- ロードされたモジュールの確認
- プロセスによるのファイル作成
- ファイルの書き換え・削除
- レジストリの変更・操作

ファイルやレジストリへのデータの追加等、潜伏や感染を示す兆候に対して事象ベースでの情報取得を行うのに用いられる。また、フィルタ以外にはスクリプトを用いたログファイルのパース作業も併用し C2 サーバのアドレス等の詳細な情報を引き出す用途で使用される。

7.2.2 パケットモニタ

マルウェアと C2 サーバの通信をプロトコルレベルで解析するために使用される。通信先の IP アドレスやポート番号、プロトコルの種別についての調査に使用される。マルウェアが感染マシン上に展開したデータをパケットから抽出できたとき、特定の検体の感染を特徴づけるシグネチャとして活用が期待できる。

しかし、マルウェアと C2 サーバとの間でやり取りされるデータに暗号化処理が適用されることは既に珍しいことではない。C2 サーバから受信した指令に基

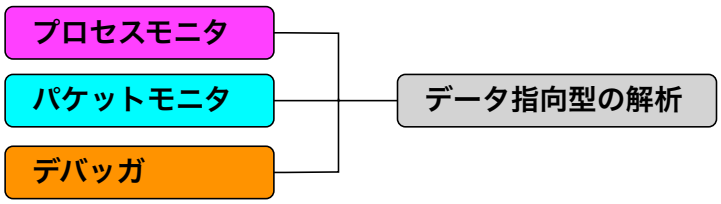


図 34　データの流れに基づく関係性の抽出

づく処理の機能の解明が必要な際、パケット解析とプログラム実行の動的解析が並行で行われる場合がある。

7.2.3　デバッガ

デバッガから得られる情報はプロセスモニタから取得可能な情報 (ファイルの名前・レジストリ名・設定値・ネットワーク接続先) よりも低レイヤーなシステム情報であり、プログラム実行中のメモリアクセスを検査することにより細かい粒度で悪意ある挙動を捉えることができる。マルウェアが API の引数に指定したバッファのアドレスからデータのダンプを行う際にはデバッガをによるブレークポイントの挿入が必要とされる。デバッガに与える情報は実行される命令のアドレスやヒープやスタックなどデバッグの最中に細かく設定しなければならないものが多く、手動での解析が一般的である。一方、DBI(Dynamic Binary Instrumentation) による解析手法は実行毎に挿入される計装処理のコードによりデバッガと同様な使用法が可能である。挿入されるコードはプログラマブルに設定できるため、自動化された解析が行える。

7.2.4　本研究で実装したツール利用における位置づけ

本研究の DBI を用いたデータフローの解析手法はプロセスモニタ、パケットモニタ、デバッガによる解析にデータフローの追跡を行うテイント解析を取り込ん

だものである (図 34)。命令単位で挙動を観測することによって得られる粒度が細かい情報と API コールやパケットデータ等の OS レベルの情報との関連付けを自動で行えるため、ヒューリスティックに悪性挙動を判定するにあたって説得力のある解析結果を得ることができる。API の処理の中で操作されているデータに着目した複数の振る舞いの関連付けを行うデータ指向型のリバースエンジニアリング手法としての活用を意図している。

7.3 提案方式の応用可能性

7.3.1 テイント解析によるアンチウイルス用のシグネチャの自動収集

動的解析から得られるマルウェアのシグネチャには API コールの引数に指定されたファイル名やレジストリに関する情報、格納される値等が考えられる。しかし、検体がシステムに与えた変更（書き換えたデータ）が意図する内容が判別できないことがある。ファイルとしての実体を持たないファイルレスマルウェア [23] の中にはレジストリ中にペイロードの取得先のホスト情報を格納し、再起動時に取得・再実行を試みる潜伏型 (Persistence) の機能を持つものが存在する。マルウェアが展開した正体不明なデータがどのような機能を担当しているのかを知るにはデータがプログラム中でどう扱われるかを観察する必要がある。マルウェアが操作した固有の文字列等のデータから派生した処理及び API コールの監視から得られた情報はデータ指向の解析から得られたヒューリスティックなシグネチャとして利用できるのではないかと考えられる。

7.3.2 record/replay ベースの解析手法の応用

record/replay は実行トレースログ (.rr ファイル) を記録・再生しプログラム実行を繰り返し観測できる手法であるが、攻撃者の内部活動 (Lateral movement) を観察するためのハニーポットとしての応用が期待できる。record された攻撃者の内部活動をテイント解析で細かく観察することにより攻撃者の動機や意図のマイニング (Intention mining) や侵入検知への活用が考えられる。テイント解析はメモリ間のデータのコピーや移動を絶えず観測するものであるため、速度面におけ

るパフォーマンス低下が憂慮されている。ネットワーク受信したデータを対象に行ったテイント解析は通常の replay 時の 27 倍以上の速度低下が確認されている [24]。しかし、record/replay はプログラム実行時の CPU やメモリの状態を再生するものであって実際にネットワーク通信を必要としないため、遅延によって C2 サーバとの通信がタイムアウトし解析が続行できない心配はない。

また、実行されるタイミングや環境によって異なる挙動を見せる検体も存在するため、そのような検体の解析に record/replay は適していると考えられる。ランサムウェアは実行された時間毎に異なる乱数から鍵を生成することもあればホスト固有の識別情報を鍵生成に用いることもある。ボット型マルウェアの機能である DGA(Domain Generation Algorithm) は接続先の C2 サーバのドメイン名を乱数を基に生成する。DGA は接続先の C2 サーバを切り替えることによってブラックリストに基づく URL フィルタリングを回避するために用いられる手法である。タイミング毎に異なるデータ生成やペイロード取得を行い様々な振る舞いを見せる機能の解析において record/replay による再現性のある動的解析は有用ではないかと考えられる。

```
{
    "type":"package",
    "id":"package--2d42dac8-c416-42c6-bc5c-7b6dcf576fc5",
    "schema_version":"5.0",
    "maec_objects":[
        {
            "type":"malware-instance",
            "id":"malware-instance--19863c16-503e-493f-8841-16c68e39c26e",
            "instance_object_refs":[
                "0"
            ],
            "labels":[
                "mass-mailer",
                "worm"
            ],
            "capabilities":[
                {
                    "name":"persistence",
                    "refined_capabilities":[
                        {
                            "name":"continuous-execution"
                        },
                        {
                            "name":"system-re-infection"
                        }
                    ],
                    "description":"The instance persists after a system reboot.",
                    "attributes":{
                        "persistence-scope":[
                            "self",
                            "other malware/components"
                        ],
                        "technique":"creates registry key"
                    },
                    "behavior_refs":[
                        "behavior--1",
                        "behavior--2"
                    ],
                    "references":[
                        {
                            "source_name":"ATT&CK",
                            "description":"Persistence",
                            "url":"https://attack.mitre.org/wiki/Persistence"
                        }
                    ]
                }
            ]
        }
    ]
}
```


図 35　MAEC 記述方式における ATT&CK の項目

7.4　ATT&CK について

7.4.1　ATT&CK の MAEC への組み込み

提案方式の概念実証に用いた ATT&CK モデルはマルウェアの不正活動に関する情報として MAEC の属性記述の項目に登録されている (図 35)。

ATT&CK は定期的に更新されており、図 36 のように Twitter で告知されている。本研究で用いた攻撃マトリクスは最新版では若干の変更が加えられていることが確認できる。

図 36　ATT&CK の定期的な情報更新の通知 (Twitter)

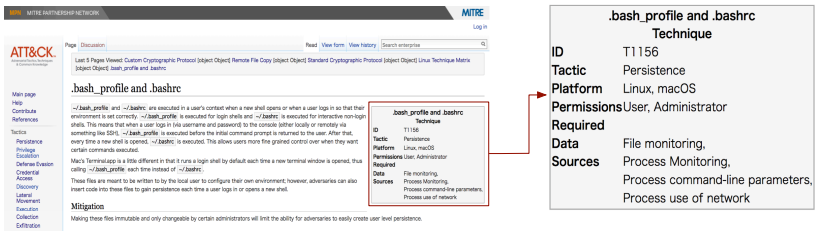


図 37　攻撃モデルの Data Sources を参照

7.4.2　ATT&CK モデルの分析と解析時の着眼点に関する考察

図 37 は ATT&CK モデル [5] の各解説ページであるが、マトリクス中の各攻撃モデルには図のように対処に必要な情報が記載されている。“Data Sources” は攻撃の解析における着眼すべきポイントが列挙されている。図 37 は .bashrc の書き換えによって攻撃者が潜伏を図るモデルであり、Data Sources には “File monitoring”(ファイル監視) が含まれている。ATT&CK[5] が現時点で定義している Data Sources は 32 種類であり、攻撃モデルは 89 種類である。

表 7 は ATT&CK モデル全 89 種類の Data Sources を確認した結果、最も多くの攻撃モデルの解析時の Data Sources として指定されていた項目の上位 7 個を抜粋したものである。“Process monitoring”(プロセス監視) が 63 種類の攻撃モデルにおいて解析時の着眼点として取り上げられている。また、“File monitoring”(ファイル監視) は 43 種類の攻撃モデルで Data sources に指定されており表中では第 2 位である。

本研究ではファイルアクセスへの監視に基づいてテイント解析によるデータフ

Data Source(解析時の着眼点)	合計
Process monitoring	63
File monitoring	43
Process command-line parameters	33
Process use of network	27
Packet capture	19
Netflow/Enclave netflow	19
Network protocol analysis	12

表 7 攻撃モデルの Data Sources によく取り上げられていた項目を調査

ローの追跡を行っていた。それはスパイウェアによる情報窃取やランサムウェアによるファイル暗号化等、マルウェアによる不正活動の多くはファイルアクセスを伴うものであるという仮説から行っていたからである。表 7 より、ファイルアクセス監視は解析の着眼点として妥当性はあると考えられる。

8. おわりに

本研究ではマルウェアが行うファイルデータへの操作の過程でマルウェアが用いたコマンドや手法に関する痕跡を収集し、動作単位で関連付けを行い、不正活動の特徴抽出を行った。解析によって得られたマルウェアの挙動の概略には暗号化を用いた形跡やデータの外部送信等の不正活動が発生した形跡が示されており、これらの指標を悪性機構を裏付ける指標として活用できることを示した。

単一の API コールに着目した従来の命令指向型のリバースエンジニアリング手法ではファイルの書き換え、消去、設定の変更等、システムに直接影響を与える動作の単位で不正活動の検知を行う。しかし、単一ではそう見えなくとも複数の動作の組み合わせによって発動する振る舞いがある。レジストリに追加された文字列データが実は C2 サーバとの通信に用いられており潜伏を意図したものであった等 [23]、ありふれたシステム操作が不正活動の下準備に用いられている。特に最近のマルウェアは自身の不正活動を隠蔽することに力を注いでいる。ファイルレスマルウェアは不正活動を行うにあたって感染端末上の OS ユーティリティを用いる等、正規の操作との区別を困難にする [15]。

提案方式はマルウェアによる情報窃取や改竄等の標的になるファイル名を指定することで、標的ファイルデータへの操作を起点として行われた不正活動の特徴抽出を自動化する手法を提案した。テイント解析を用いてデータの流れを追跡し、監視対象のファイルデータを処理した各 API の動作の関連付けを行い、1 つのファイル読み出し API から派生した他の動作の特定を行った。テイント解析で抽出した命令列からプロセス名やライブラリ名等の OS レベルの情報の取得する為に仮想マシン内部観察 (VMI) の機能を用いた。通常の解析手法では、1 つの API 操作から他の API 操作との関連付けを行ったり派生した処理から攻撃者が用いたツールやライブラリの特定は手間がかかる作業であるが、データフローに基づいた痕跡の収集によってそれらを自動化できることが分かった。

ユースケースを例に情報窃取・遠隔操作・暗号化を対象に行なった特徴抽出の実験では、各不正活動においてその悪性機構の指標を抽出できることを確認した。さらに解析によって得たセマンティクスの概要をグラフ出力によって可視化した。攻撃者の内部活動のモデルである ATT&CK マトリクスから攻撃モデルの 26%か

ら悪性機構の指標を抽出できることが分かった。攻撃者の標的となった情報に応じて解析時の着眼点は変化するため、本研究は重要度の高いファイルに着目したデータフロー解析により攻撃者の意図を柔軟に抽象化できることを示した。

謝辞

本研究と論文執筆を進めるにあたり、御指導を頂きました本学研究科の門林雄基教授、藤川和利教授、林優一教授に深く感謝致します。

主指導教員である門林教授は本研究において熱心な御指導と御鞭撻を頂いたほか、研究生活の中で様々な面で勉強になる話題を提供してくださり、また生活面や健康面にも気遣ってくださったことに、深く感謝致します。お陰で常に身の引き締まる思いで研究に邁進することができました。

サイバーレジリエンス構成学研究室の宮本大輔特任准教授、樫原茂助教、Doudou Fall 助教を始めとする皆様に深く感謝致します。宮本大輔特任准教授は研究内容に関する助言だけでなく美味しい食事に連れて行ってくださり私の健康を気遣ってくれました。樫原茂助教は定期的に声を掛けてくださり、ホワイトボードでの私の説明に根気強く付き合ってくれたり等、多くの助言をくださりました。Doudou Fall 助教は研究生活の中で私の英文の添削に付き合ってくれたり、また、論文執筆中の私を励ましてくれました。

また、博士前期課程 1 年目にお世話になりました情報基盤システム学研究室のスタッフの皆様と、互いに切磋琢磨した同期の皆様に感謝を申し上げます。

最後に、経済的、その他あらゆる面から私を支援してくださり、遠方の地から温かく見守ってくれた両親に心より感謝を申し上げます。

参考文献

[1] Verizon. 2017 Data Breach Investigations Report. Technical report, Verizon, 2017.

[2] NTTSecurity. Global Threat Intelligence Report 2017. Technical report, NTTSecurity, 2017.

[3] Cuckoo Sandbox - Automated Malware Analysis. `https://www.cuckoosandbox.org/` (accessed 2018-03-03).

[4] The MITRE Corporation. Top Botnets and How MAEC Can Help Keep You Out of Their Clutches Briefing Slides. `https://maec.mitre.org/about/docs/MAEC_Botnets_briefing.pdf` (accessed 2018-03-03).

[5] The MITRE Corporation. ATT&CK Linux Technique Matrix. `https://attack.mitre.org/wiki/Linux_Technique_Matrix` (accessed 2018-02-15).

[6] Platform for Architecture-Neutral Dynamic Analysis. `https://github.com/panda-re/panda` (accessed 2018-03-03).

[7] Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda. Panorama: capturing system-wide information flow for malware detection and analysis. In *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 116–127. ACM, 2007.

[8] Zhi Wang, Xuxian Jiang, Weidong Cui, Xinyuan Wang, and Mike Grace. Reformat: Automatic reverse engineering of encrypted messages. In *ESORICS*, Vol. 9, pp. 200–215. Springer, 2009.

[9] The MITRE Corporation. MAEC Language Overview, Version 4.1. `https://maec.mitre.org/about/docs/MAEC_Overview.pdf` (accessed 2018-03-03).

[10] PANDA plugin: spaniel. https://github.com/shun-yo/Spaniel (accessed 2018-03-03).

[11] Graphviz - Graph Visualization Software. https://www.graphviz.org/ (accessed 2018-03-03).

[12] About the Metasploit Meterpreter. https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/ (accessed 2018-03-03).

[13] Metasploit — Penetration Testing Software, Pen Testing Security. https://www.metasploit.com/ (accessed 2018-03-09).

[14] OpenSSL Cryptography and SSL/TLS Toolkit. https://www.cuckoosandbox.org/.

[15] ファイルレスマルウェアとは？〜非マルウェア型攻撃を理解する〜. https://www.cybereason.co.jp/blog/malware/2094/ (accessed 2018-03-03).

[16] Felix Gröbert, Carsten Willems, and Thorsten Holz. Automated identification of cryptographic primitives in binary programs. In *International Workshop on Recent Advances in Intrusion Detection*, pp. 41–60. Springer, 2011.

[17] 与那嶺俊, 門林雄基. DBI を用いたランサムウェアのファイルアクセス解析の一検討 (マルチメディア情報ハイディング・エンリッチメント). 電子情報通信学会技術研究報告= IEICE technical report: 信学技報, Vol. 117, No. 128, pp. 87–92, 2017.

[18] Khaled Yakdan, Sergej Dechand, Elmar Gerhards-Padilla, and Matthew Smith. Helping johnny to analyze malware: A usability-optimized decompiler and malware analysis user study. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pp. 158–177. IEEE, 2016.

[19] Anti-VM and Anti-Sandbox Explained. https://www.cyberbit.com/anti-vm-and-anti-sandbox-explained/ (accessed 2018-03-03).

[20] Modifying KVM (qemu-kvm) settings for malware analysis. http://blog.prowling.nu/2012/09/modifying-kvm-qemu-kvm-settings-for.html (accessed 2018-03-03).

[21] Lok-Kwong Yan, Manjukumar Jayachandra, Mu Zhang, and Heng Yin. V2e: combining hardware virtualization and softwareemulation for transparent and extensible malware analysis. *ACM Sigplan Notices*, Vol. 47, No. 7, pp. 227–238, 2012.

[22] Process Monitor for Dynamic Malware Analysis. https://blogs.technet.microsoft.com/motiba/2016/05/04/process-monitor-for-dynamic-malware-analysis/ (accessed 2018-03-03).

[23] トレンドマイクロセキュリティブログ: より高度な「ファイルレス活動」を実現した一連のマルウェアを確認. http://blog.trendmicro.co.jp/archives/15653 (accessed 2018-03-03).

[24] Brendan Dolan-Gavitt, Josh Hodosh, Patrick Hulin, Tim Leek, and Ryan Whelan. Repeatable reverse engineering with panda. In *Proceedings of the 5th Program Protection and Reverse Engineering Workshop*, p. 4. ACM, 2015.

付録

A. 追加資料 1

Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Execution	Collection	Exfiltration	Command and Control
.bash_profile and .bashrc	Exploitation of Vulnerability	Binary Padding	Bash History	Account Discovery	Application Deployment Software	Command-Line Interface	Audio Capture	Automated Exfiltration	Commonly Used Port
Bootkit	Process Injection	Clear Command History	Brute Force	File and Directory Discovery	Exploitation of Vulnerability	Graphical User Interface	Automated Collection	Data Compressed	Communication Through Removable Media
Browser Extensions	Setuid and Setgid	Disabling Security Tools	Credentials in Files	Network Service Scanning	Remote File Copy	Local Job Scheduling	Browser Extensions	Data Encrypted	Connection Proxy
Create Account	Sudo	Exploitation of Vulnerability	Exploitation of Vulnerability	Permission Groups Discovery	Remote Services	Scripting	Clipboard Data	Data Transfer Size Limits	Custom Command and Control Protocol
Hidden Files and Directories	Valid Accounts	File Deletion	Input Capture	Process Discovery	SSH Hijacking	Source	Data Staged	Exfiltration Over Alternative Protocol	Custom Cryptographic Protocol
Local Job Scheduling	Web Shell	HISTCONTROL	Network Sniffing	Remote System Discovery	Third-party Software	Space after Filename	Data from Local System	Exfiltration Over Command and Control Channel	Data Encoding
Rc.common		Hidden Files and Directories	Private Keys	System Information Discovery		Third-party Software	Data from Network Shared Drive	Exfiltration Over Other Network Medium	Data Obfuscation
Redundant Access		Indicator Removal from Tools	Two-Factor Authentication Interception	System Network Configuration Discovery		Trap	Data from Removable Media	Exfiltration Over Physical Medium	Domain Fronting
Trap		Indicator Removal on Host		System Network Connections Discovery			Input Capture	Scheduled Transfer	Fallback Channels
Valid Accounts		Install Root Certificate		System Owner/ User Discovery			Screen Capture		Multi-Stage Channels
Web Shell		Masquerading							Multi-hop Proxy
		Obfuscated Files or Information							Multiband Communication
		Process Injection							Multilayer Encryption
		Redundant Access							Remote File Copy
		Rootkit							Standard Application Layer Protocol
		Scripting							Standard Cryptographic Protocol
		Space after Filename							Standard Non-Application Layer Protocol
		Timestomp							Uncommonly Used Port
		Valid Accounts							Web Service

https://attack.mitre.org/wiki/Linux_Technique_Matrix 最終アクセス(2018/02/13)

表 8 最新版の ATT&CK 攻撃マトリクス (Linux)

表 8 は 6 章で用いた MITRE の ATT&CK の資料である。ATT&CK[5] は本稿執筆時点で更新されているため、比較のために掲載する。

B. 発表リスト

1. 与那嶺俊, 門林雄基. “DBI を用いたランサムウェアのファイルアクセス解析の一検討.” 電子情報通信学会 ICSS 研究会 2017 年 7 月

2. 与那嶺俊, 門林雄基, “マルウェア解析における Virtual Machine Introspection を用いた悪性機構抽出の自動化” 暗号と情報セキュリティシンポジウム (SCIS 2018) , 2018 年 1 月