脆弱性を利用した標的型攻撃のための解析ツール

鵜飼裕司　小林偉昭　中野学

(独)情報処理推進機構
セキュリティセンター　情報セキュリティ技術ラボラトリー

概要: 近年、特定の企業・組織を標的とした標的型攻撃が深刻化している。しかし、近年の標的型攻撃はセキュリティ脆弱性を悪用する例が増加しており、かつ、耐解析機能が多数実装されているため、迅速かつ効率的な解析は困難である。有効な対策分析のためには詳細なコードの解析が不可欠であり、迅速かつ効率的な解析手法の開発が求められている。そこで本稿では、新しい標的型攻撃を迅速かつ安定的に分析を行うためのツール整備について提案する。まず、近年の標的型攻撃に実装されている代表的な耐解析機能を自動的に検出・除去する手法について述べる。次に、それら手法に基づき開発を進めているツールの機能概要と実装の終了した部品の機能評価について述べる。

An analysis tool for targeted attacks using security vulnerabilities

Yuji Ukai　Hideaki Kobayashi　Manabu Nakano

Information Technology Promotion Agency, Japan
IT Security Center (ISEC), Security Engineering Laboratory

Abstract: Organizations are increasingly becoming the target of attacks following customized attack models. However in recent years attacks exploiting security flaws are increasing and anti-analysis techniques are widely implemented in malware thus making a quick and efficient analysis very difficult. Since performing effective steps to cope with attacks requires a detailed analysis of the code, we face a huge demand for techniques that allow quick and efficient coverage. This paper describes the exemplary implementation of techniques that automatically detect and cope with anti-analysis routines seen in recent years. Furthermore, it is an introduction to our implementation of these techniques - the "Customized Attack Analysis Tool". Our intention is to provide a tool to aid the quick and reliable analysis of anti-analysis techniques.

1. はじめに

近年、特定の企業・組織を標的とした標的型攻撃による被害が深刻化している。標的型攻撃では攻撃対象が限定的であるため、攻撃発生前に情報を入手し、適切な対応を取る事が難しい。セキュリティベンダーでも攻撃が見えにくいため情報が得られにくく、実態を正確に把握するのは困難である。このため、企業・組織は十分な情報や技術的解決策を得られないという状況にある。このため独立行政法人 情報処理推進機構（以降、IPA）では、近年の標的型攻撃に関する調査研究を実施し、2008 年 3 月に調査報告書（以降、IPA 標的型攻撃調査報告書）を公開した[1]。いくつかの標的型攻撃を詳細に解析した結果、それらについて有効な対策を見出したが、今後発生しうる標的型攻撃に対して迅速に対応するためには、脅威の継続監視と解析が必要であることが分かった。しかし同時に、近年の標的型攻撃には多数の解析を困難にする要素が存在しており、詳細解析には熟練した技術が必要である事も判明した。これは、迅速かつ安定的な脅威解析が困難な状況にあることを意味している。そこで本稿では、近年の標的型攻撃に実装されている代表的な耐解析機能を自動的に検出・除去する手法について述べる。また、現在、それら手法のツール化を進めており、その機能概要と実装の終了した部品

の機能評価について述べる。本提案は、新しい標的型攻撃を迅速かつ安定的に分析を行うためのツールを整備することで、有効な対策分析を支援する事を目的とする。

2. 標的型攻撃解析の現状

近年の標的型攻撃には多数の耐解析機能が実装されているため、詳細な解析を行うには手動でそれら耐解析機能を解除する必要がある。以下に、従来の一般的なマルウェア解析手法、および、それら手法を近年の標的型攻撃の解析に適用した際の問題点を示す。

2.1. 一般的なマルウエア解析手法

(1) 動的解析

プログラムの挙動に着目する解析手法である。プログラムを実際に実行し、ファイルへのアクセスや通信状況等を監視する。プログラムの実行された経路についての挙動を容易に把握できるが、プログラムの仕様、および実行されていない処理を解析することが困難である。

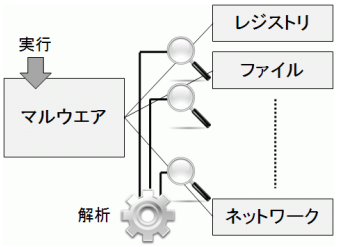

図 1 動的解析

(2) 静的解析

プログラムの構造、および仕様に着目する解析手法である。プログラムを逆アセンブルし、1 命令ずつ解析することで、プログラムの仕様を完全に把握することが可能である。しかし、完全な解析を行うためには相応の知識・経験、および時間が必要である。前述のように、動的解析はプログラムの概要を把握するためには有用でるが、正確な解析を行う事ができない場合がある。そのため、通常は動的解析を行った上で、静的解析による詳細な解析を実施する。

2.2. 標的型攻撃における耐解析機能

標的型攻撃では、動的解析および静的解析のいずれに対しても解析を困難にするための様々な耐解析機能を有している。主な耐解析機能は次の通りである。

(1) 実行コードの難読化（パッキング）

プログラム中の実行コードに対して圧縮、および暗号化等のエンコード処理を施すことで静的解析を妨害する機能である。処理が施されたプログラムは、実行時に元の状態に復元されて実行される。そのため、プログラムを実行していない状態での逆アセンブルが困難である。

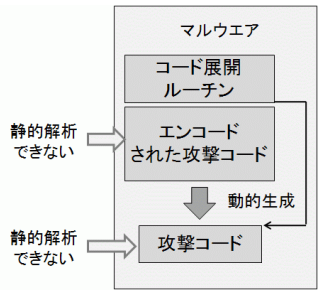

図 2 コード難読化

(2) 独自 IAT（Import Address Table）

通常、プログラム中に格納されている IAT と呼ばれる管理構造を確認することで、プログラムが呼び出す API を把握することが可能である。しかし、標的型攻撃では、プログラムの実行時に動的に IAT をメモリ上に生成し、プログラムが呼び出す API の隠ぺいを行うものが存在する。

(3) データ形式ファイルを介した攻撃

標的型攻撃では、Microsoft Office 製品や Adobe 製品等で利用される文章ファイル、表計算ファイル等のデータ形式ファイルを介した攻撃が数多く存在する。こうしたデータ形式ファイルを介した攻撃では、攻撃コードがファイル内の正常なデータに混入されている。そのため、解析を行うには、ファイル内に紛れた攻撃コードを特定する必要がある。

(4) 不完全な攻撃コードによる攻撃の未発動

標的型攻撃において利用される攻撃コードは、完成度が一定しておらず、中には完成度が低いため攻撃コードが発動しないものも存在する。こうした場合、攻撃コードの実行を捕捉することができず、解析が困難となる。

(5) 他プロセスへのコードインジェクション

標的型攻撃の多くは、実行時、プログラム中に保持している攻撃コードの本体部分を他の正常なプロセスに注入して実行させる。これにより、攻撃コードを実行する主体を偽装し、解析を困難にする。

(6) リモートホストからのコード受信と実行

標的型攻撃の多くは、攻撃コードが実行されるとネットワーク上のサーバと通信を行い、第二の攻撃コードを受信し、実行する。そのため、プログラムを完全に解析するためには、プログラムを実行し、実際のサーバと通信させる必要がある。また、サーバが閉鎖された場合、解析を行うことが非常に困難となる。

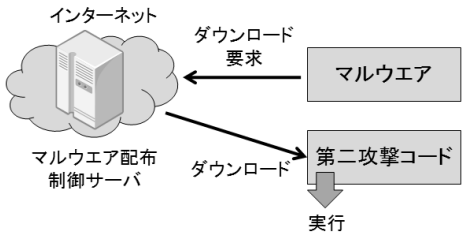

図 3 コード受信と実行

(7) デバッガ検出

静的解析を行う際、デバッガを利用してプログラムを実行することで、実行時におけるプログラムのメモリ、レジスタ等を参照することが可能となる。しかし、多くの標的型攻撃はデバッガ検出機能を備えており、デバッガの有無に応じてプログラムの挙動を変更するものが存在する。

3. 標的型攻撃解析の効率化

前述のように、標的型攻撃の解析においては、さまざまな耐解析機能により効率的な解析が難しくなっている。このため IPA では、前述の耐解析機能を解除し、効率的に解析を実施できるツールの開発を進めている。本節では、その解析ツールの各コンポーネントとその概要、および、実装について述べる。なお、近年の標的型攻撃は、Microsoft 社の Windows プラットフォーム用のものが大半である事から、本ツールは、Windows プラットフォームで動作する標的型攻撃の解析支援を目的とする。

3.1. コアエンジン

標的型攻撃を行うマルウェアを実行してプログラムに施された耐解析機能を自動的に解除し、従来と同様の解析が可能な実行形式ファイルを生成する。また、プログラムに施されていた難読化手法、デバッガ検出機能、および API 呼び出しについての情報をレポート出力する。本サブツールは、耐解析機能の解除に必要となるその他のサブツールや機能を包含する。コアエンジンを構成する各機能部、サブツールの機能詳細について述べる。

(1) インストラクショントレーサー

CPU および OS のデバッグ機能を利用し、対象プログラムを自身の監視下で実行する。これにより、対象プログラムの実行を制御する。具体的には、任意の箇所での実行の停止・再開、レジスタ・メモリの参照を行うことが可能となる。また、対象プログラムを停止した際に、他の機能部・サブツールの機能を実行する。

(2) 逆アセンブラ

インストラクショントレーサー部と連携し、対象プログラムの実行コードを逆アセンブルする。これにより、実行コード中における API 呼び出しを検出する。また、実行コード中における分岐命令を検出し、実行コードのプログラム構造を解析する。実行コード中の分岐箇所でのみ実行を停止することで、インストラクショントレーサー部の効率化が可能となる。

(3) PE ビルダー

対象プログラムのプロセスメモリイメージに基づいて、ファイルシステム上に実行形式ファイルを生成する。これにより、IDA[2]等の既存の解析ツールを利用した解析が可能となる。

(4) セクションパッキング解除

標的型攻撃を行うマルウェアの中には、マルウェアの本体コードが格納されたセクションに対して難読化を施しているものが存在する。セクションパッキング解除部は、インストラクショントレーサー部、および逆アセンブラ部と連携し、難読化が施されたセクションの実行時における復元、および当該セクシ

ョン内の実行コードの実行を監視する。これにより PE ビルダーと連携し、難読化が解除されたセクションのプロセスメモリイメージに基づいた実行形式ファイルを生成することが可能である。

(5) データパッキング解除

標的型攻撃を行うマルウェアの中には、プログラムの実行時に動的に確保したデータ用のメモリ領域に対して、難読化された自身のコードを復元し、実行するものが存在する。データパッキング部は、インストラクショントレーサー部、および逆アセンブラ部と連携し、データ用のメモリ領域へのアクセス、および当該セクション内の実行コードの実行を監視する。これにより、PE ビルダーと連携し、セクションパッキング同様、難読化が解除された状態でのプロセスメモリイメージに基づいた実行形式ファイルを生成する。

(6) シェルコードデコーダ

標的型攻撃を行うマルウェアの中には、複数の攻撃コードを段階的に実行するものが存在する。シェルコードデコーダ部では、実行時における難読化された攻撃コードの復元、および当該攻撃コードの実行を監視する。これにより、PE ビルダーと連携し、難読化が解除された状態での攻撃コードを実行形式ファイルとして生成する。

(7) API コールトレーシング

インストラクショントレーサー部、および逆アセンブラ部と連携することで、プログラムからの API 呼び出しを監視し、呼び出された API 名、引数、返り値を記録する。API 名の解決には、後述の「API 名解決ツール」を利用する。

(8) メモリトレーシング

データの読み込み・書き込みを行う API について、API 引数に指定された読み込み・書き込み用のメモリ領域の変更を追跡し、記録する。

(9) ハンドルトレーシング

ファイル操作やソケット操作を行う API について、API 引数に指定されたハンドルの変更を追跡し、記録する。

3.2. シェルコード展開ツール

データ形式ファイルを走査し、攻撃コードの可能性が高い部分を自動的に特定する。また、特定した攻撃コードから実行形式ファイルを生成する。これにより、データ形式ファイル中に混入れた攻撃コードを従来と同様の手法で解析することができる。攻撃コードの特定は、データファイルを先頭から順次逆アセンブルし、意味のある機械語命令ブロックを判定条件とする。

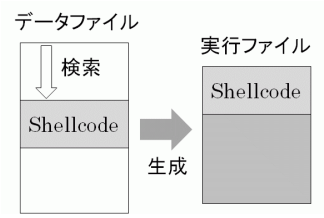

図 4 シェルコード展開ツール

3.3. プロセス毎パケットモニター

プロセス毎の TCP プロトコルによる通信を監視し、TCP セッション毎の通信内容を記録する。これにより、標的型攻撃を行うマルウェアの通信を記録し、通信プロトコルの解析を支援する。既存のパケットモニターは、ネットワークインターフェイス毎でのモニタリングが前提となっており、プロセス毎のモニタリングを行うことが困難である。プロセス毎パケットモニターは、TDI（Transport Driver Interface）フィルタドライバとして実装することで、プロセス毎でのパケットのモニタリングを実現する。

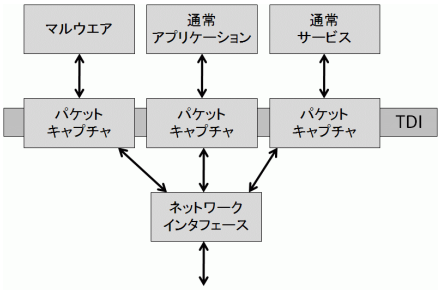

図 5 プロセス毎パケットモニター概要

3.4. マルウェアサーバエミュレーター

プロセス毎パケットモニターを利用して事前に取得した標的型攻撃を行うマルウェアとマルウェア配布・制御サーバとの通信記録に

基づいて、サーバの挙動をローカルマシン上でエミュレートする。対象プログラムが呼び出す TCP 接続用の API をフックし、記録しておいたサーバから送信されたデータを再生する。これにより、マルウェア配布・制御サーバが閉鎖された後も、繰り返し解析を実施することが可能となる。また、マルウェア配布・制御サーバが有効な間も、本ツールを利用して解析を行うことで、繰り返し実際のサーバに接続することを避けることができる。これは、サーバの運営者からマルウェア解析者の存在を隠ぺいするために有用である。

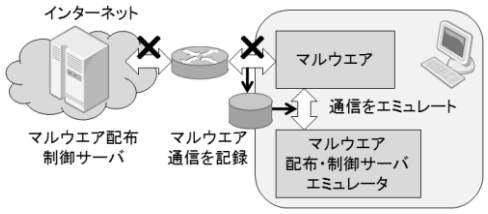

図 6 マルウェアサーバエミュレータ

3.5. デバッガでの「やり直し」と「バックトレース」

デバッガによる解析において、対象プログラムの実行状況を、事前にスナップショットを取得した任意のポイントに復元する。スナップショットの作成は、作成時点における対象プログラムのメモリ、レジスタの内容をファイルに保存する。また、復元時は、これらの情報を対象プログラムのメモリ、レジスタに書き戻す。これにより、プログラムのエントリポイントからデバッグのやり直しを避けることできるため、効率的な解析を行うことが可能となる。

3.6. コードインジェクショントレーサー

コードインジェクショントレーサー部は、以下の 2 つの機能部から構成される。カーネルモードインジェクショントレーサー部は、コアエンジンから分離された単独のサブツールとして実装する。

(1) ユーザーモードインジェクショントレーサー

対象プロセスから他のプロセスへのコードの注入、および当該コードの実行を監視し、注入され実行コードを抽出する。他のプロセスにデータの書き込みを行う WriteProcessMemory API、および他のプロセス中の実行コードを実行する CreateRemoteThread API をフックすることでこれらの機能を実現する。また、抽出した実行コードを、対象プロセスのセクションとして追加し、PE ビルダーと連携して実行形式ファイルを生成する。これにより、IDA 等の既存の解析ツールを利用した解析が可能となる。

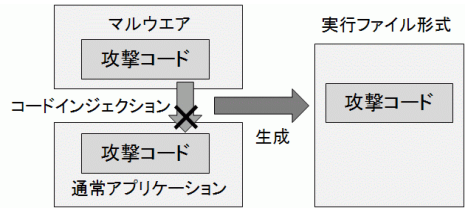

図 7 ユーザーモードインジェクショントレーサー

(2) カーネルモードインジェクショントレーサー

標的型攻撃を行うマルウェアの中には、実行時にシステムファイルを生成してカーネルに組み込むことで、カーネル空間から特定のプロセスに攻撃コードを注入するものが存在する。カーネルモードインジェクショントレーサー部は、カーネルに組み込まれたシステムファイルの実行を追跡し、プロセスに注入される攻撃コードを抽出する。また、PE ビルダー部と連携することで、実行形式ファイルを生成する。

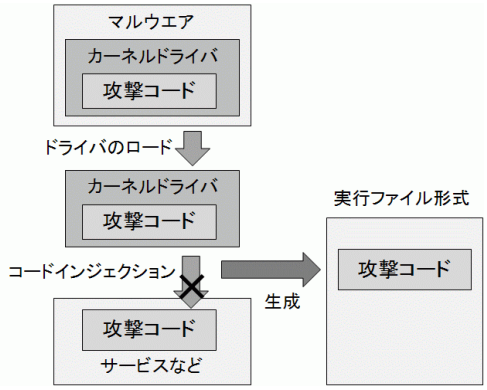

図 8 カーネルモードインジェクショントレーサー

3.7. レポートエンジン

レポートエンジンは、XML レポートジェネレータ、および、HTML フォームから構成される。レポートには、プログラムに施されてい

た難読化手法、デバッガ検出機能、API トレーサー等のツールが記録した情報などが出力される。

3.8. API 名解決ツール

事前に作成しておいた API プロトタイプ DB を利用し、API トレーサーと連携することで、呼び出された API の API 名、引数、および返り値の型についての情報を解決する。また、IDA 用のコメントファイルを生成し、これらの情報を IDA にコメントとしてインポートすることができる。

3.9. IAT リビルダー

実行時にメモリ上に生成された IAT を検出し、PE ビルダーと連携して適切な IAT を保持した実行形式ファイルを生成する。独自 IAT の検出は、インストラクショントレーサー部、および逆アセンブラ部と連携し、実行コード中における分岐命令から呼び出された API のエントリポイントまでの実行経路を追跡することで実現する。これにより、生成された実行形式ファイルを IDA 等の高機能アセンブラで解析する際に、正しい API 呼び出し情報を確認することが可能となる。

3.10. 実行パスアナライザー

インストラクショントレーサー部、および逆アセンブラ部と連携することで、実行されたアドレスを記録し、IDA 用のコメントファイルを生成する。これにより、当該コメントファイルを IDA にインポートすることで、プログラムの実行経路を IDA 上で確認することが可能となる。

3.11. デバッガ検出対策

プログラムに施されているデバッガ検出対策を回避する[3]。デバッガを利用してプログラムを実行した場合、それを示す情報が OS の管理するプロセス管理情報に記録される。これらの情報は、対象プロセスのプロセスメモリ内に含まれるため、コアエンジンからパッチングを行うことでデバッガの存在を隠ぺいする。

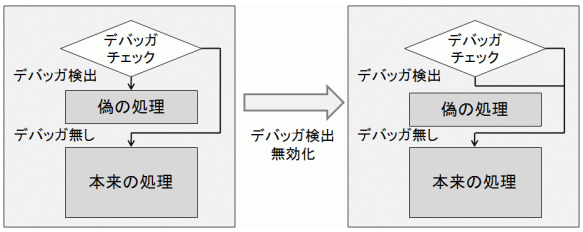

図 9 デバッガ検出対策

4. 本手法の有用性

IPA 標的型攻撃調査報告書にて解析が実施された検体「Mdropper」、「PCCLIE」、「PPDROP」、「Peacomm」 [4]、および、研究用データセット CCC DATAset 2008 のマルウェア検体（以降、CCC2008 検体）において本手法の有効性に関する調査を行った。本手法をツール化する事により、これら検体に実装されている耐解析機能をすべて解除できる見込みである。また、現在本手法のツール化を進めており、開発途中であるコアエンジンに CCC2008 検体を適用した結果、インストラクショントレーサー部の正常動作を確認した。

5. さいごに

今後は、本手法のツール実装を進める。また、本手法をさらに多くの標的型攻撃、および CCC2008 検体を含むマルウェア検体に適用し、その有用性に関する調査を行う。新たな耐解析機能を発見した場合は、その耐解析機能を無効化する仕組みをツールに追加実装していくことで、迅速かつ安定的に分析を行うためのツール整備を推進していく予定である。

参考文献

[1] IPA,「近年の標的型攻撃に関する調査研究 調査報告書」, 2008 年 3 月
[2] The IDA Pro Disassembler and Debugger, http://www.hex-rays.com/idapro/
[3] ITPro, マルウェアの解析対策を無効にする Anti-Anti-Debugging ツールを開発, http://itpro.nikkeibp.co.jp/article/Watcher/20071119/287574/
[4] シマンテックセキュリティレスポンス, http://www.symantec.com/ja/jp/security_response/index.jsp