Vol.2014-CSEC-66 No.17
Vol.2014-SPT-10 No.17
2014/7/3

マルウェア動的解析結果の可視化の一手法

星澤裕二[†1]　神薗雅紀[†1†2†3]

マルウェア動的解析を利用することにより，短時間で容易にマルウェアの挙動に関するさまざまな情報をログやレポートで確認することができる．しかし，多大な情報量により詳細な挙動を把握できる反面，専門的な知識や経験が少ない場合，解析結果を十分に理解できなかったり，重要な挙動を見落としてしまったりする可能性がある．その原因としては，情報量の多さに加え，解析結果が単なる API 呼び出し履歴の羅列であり，可読性が低いことが挙げられる．そこで，本論文では，マルウェア動的解析の結果を視覚的に把握しやすくする手法について提案する．提案手法では，マルウェアに関連するプロセスの親子関係とともに各プロセスの API 呼び出しを時系列に配列して表示する．

A Method to Visualize the Result of Malware Dynamic Analysis

YUJI HOSHIZAWA[†1]　MASAKI KAMIZONO[†1†2†3]

Using malware dynamic analysis, we can easily confirm various information about the malware behavior in a short time by logs or reports. However, although a large amount of information enables us to take hold of the detailed behavior of malware, there are possibilities that we cannot fully understand the analysis results or overlook the important behavior in the absence of specialized knowledge or experience. The reason is the analysis result is nothing but a list of calling record and not readable, in addition to the amount of information is too much. In this paper, we propose a method to visualize the result of malware dynamic analysis. The proposed method shows the parentage of process related to malware and API calling of each process chronologically.

1. はじめに

近年，標的型攻撃のような局所的に感染するマルウェアや日々大量に出現する新種や亜種に対処するため，マルウェア動的解析のニーズが高まっている．Anubis[1]や Threat Expert[2]のようにオンラインで利用できるサービスだけでなく，Cuckoo Sandbox[3]のように利用者が自ら構築して利用する動的解析システムも存在している．こうしたマルウェア動的解析を利用することにより，短時間で容易にマルウェアの挙動に関するさまざまな情報をログやレポートで確認することができる．

しかし，多大な情報量により詳細な挙動を把握できる反面，専門的な知識や経験が少ない場合，解析結果を十分に理解できなかったり，重要な挙動を見落としてしまったりする可能性がある．

その原因としては，情報量の多さに加え，解析結果が単なる API 呼び出し履歴の羅列であったり，API 呼び出しをファイルアクセスやレジストリーアクセスなどの単位でグルーピングしたりするだけで，可読性が低いことが挙げられる．

そこで，本研究では，マルウェア動的解析の結果を視覚的に把握しやすくする手法について提案する．提案手法では，マルウェアに関連するプロセスの親子関係とともに各プロセスの時間情報付き API 呼び出しを時系列に配列して表示する．

本稿では，時刻情報付き API 呼び出し履歴をイベントと呼び，イベントの時系列表示をタイムラインと呼ぶ．

2. 提案手法

本章では，提案手法の詳細について述べる．本提案は，プロセスツリー，タイムライン，ヒートマップの 3 つの可視化を組み合わせたものである．

本研究では，Cuckoo Sandbox の解析結果をもとに可視化の可能性について検討した．

2.1 プロセスツリー

プロセスツリーは，プロセスの一覧とプロセスの親子関係を確認できる．プロセスツリーのイメージを図 1 に示す．

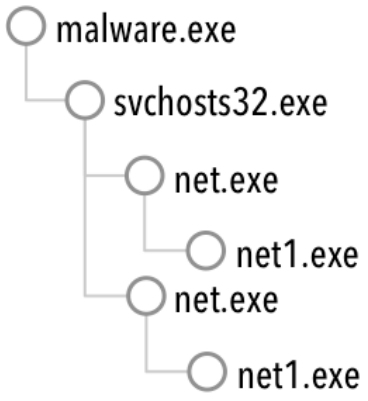

図 1　プロセスツリーのイメージ

†1 株式会社セキュアブレイン
Secure Brain Corporation
†2 独立行政法人 情報通信研究機構
National Institute of Information and Communications Technology
†3 横浜国立大学
Yokohama National University

実行されたプロセスを 1 行ずつ配置し，子プロセスは，親プロセスからインデントして表示する．例えば，図 1 中の net.exe というプロセスは svchosts32.exe によって実行された子プロセスであることを表している．

JSON 形式で出力された Cuckoo Sandbox の解析結果ファイルである report.json では，processtree オブジェクトにプロセスの親子関係がまとめられている．図 2 に processtree オブジェクトの例を示す．

```
"processtree": [
    {
        "parent_id": 1724,
        "pid": 1920,
        "children": [
            {
                "parent_id": 1920,
                "pid": 1452,
                "children": [
                    {
                        "parent_id": 1452,
                        "pid": 1644,
                        "children": [
                            {
・・・中略・・・
                            }
                        ],
                        "name": "svchosts32.exe"
                    }
                ],
                "name":
"0ab856b12055aae9c28cda03f5ad910a4e1d70f7.exe"
            }
        ],
        "name":
"0ab856b12055aae9c28cda03f5ad910a4e1d70f7.exe"
    }
],
```

図 2　processtree オブジェクトの例

Cuckoo Sandbox では，イベントを registry, filesystem, process, services, network, synchronization の 6 つのカテゴリーに分類している．このカテゴリーを利用して点の形や色を変え，イベントの種類を視覚的に表現することも可能である．

report.json では，calls オブジェクトにプロセスごとのイベントが発生順にまとめられている．図 3 に calls オブジェクトの例を示す．

```
"calls": [
    {
        "category": "process",
        "status": true,
        "return": "0x00000001",
        "timestamp": "2014-05-29 07:32:04,837",
        "thread_id": "1420",
        "repeated": 0,
        "api": "VirtualProtectEx",
        "arguments": [
            {
                "name": "Protection",
                "value": "0x00000040"
            },
            {
                "name": "ProcessHandle",
                "value": "0xffffffff"
            },
            {
                "name": "Address",
                "value": "0x0040a168"
            },
            {
                "name": "Size",
                "value": "0x000151f6"
            }
        ]
    },
```

図 3　calls オブジェクトの例

processtree オブジェクトは，各プロセスの関係を階層構造で表現しており，先頭のオブジェクトから順に処理することでプロセスツリーを容易に表示することが可能である．

2.2 タイムライン

タイムラインは，プロセスツリーの右側に配置する．プロセスの開始から終了までの時間を横棒で表し，指定された時間の範囲内のイベントを時間軸に沿って棒上に点として配置する．点を選択するとイベントの詳細を閲覧することができる．

指定された時間の範囲内に存在する API 呼び出しを画面上に配置するために，timestamp のデータを利用する．timestamp は解析した日時が起点となっているが，first_seen を利用することにより，起点日時を他のマルウェア解析結果と統一することが可能である．
図 4 の下部にタイムラインの例を示す．

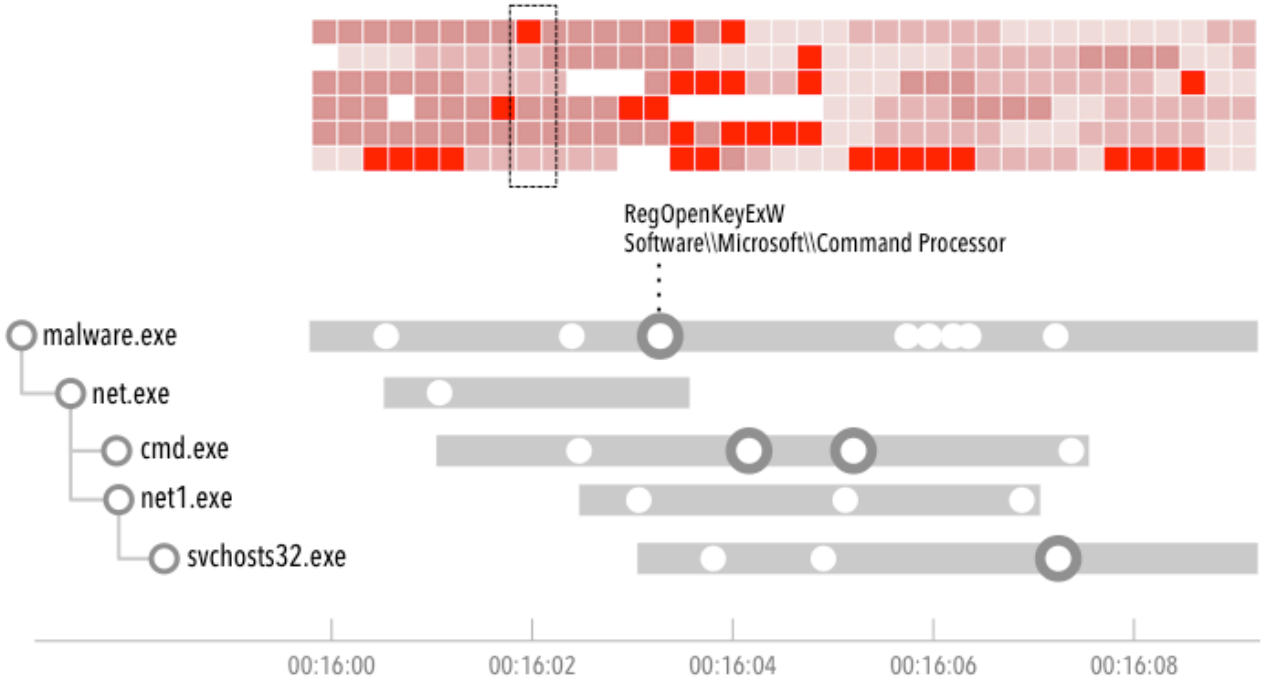

図 4　タイムラインとヒートマップの例

calls オブジェクトは，マルウェアの全ての API 呼び出しが列挙されているため，多量のデータとなり，タイムラインに表示した場合，煩雑になってしまうことがある．

calls オブジェクトではなく，calls オブジェクトのダイジェストである enhanced オブジェクトを利用することも可能である．calls オブジェクトと比較して情報量は少なくなるが，マルウェアの挙動を把握するためには，十分な情報が登録されている．利用対象者により，使用するデータを変更することも検討したい．図 5 に enhanced オブジェクトの例を示す．

```
"enhanced": [
    {
        "timestamp": "2014-05-29 07:32:05,218",
        "object": "file",
        "data": {
            "file":
"C:¥¥DOCUME~1¥¥ADMINI~1¥¥LOCALS~1¥¥Temp¥¥0ab856b12055aae9c28cda03f5ad910a4e1d70f7.exe"
        },
        "event": "execute",
        "eid": 1
    },
```

図 5　enhanced オブジェクトの例

2.3 ヒートマップ

大量の API 呼び出し履歴を整理して，マルウェアの一連の挙動の俯瞰を可能にするためにヒートマップを利用する．

前述のタイムラインは，マルウェアの挙動をミクロで捉えるために利用するのに対し，ヒートマップは，マルウェアの挙動をマクロで捉えることができる．図 4 にヒートマップの表示例を示す．

複数のマルウェアのヒートマップを比較することにより，類似性についての情報を得ることが可能となるが，本研究では，タイムラインに表示させるイベントを選択するために利用する．

カテゴリーごとに単位時間あたりのイベント数からヒートマップを作成する．色のグラデーションで表し，濃色になるほどイベントが多く，淡色になるほどイベントが少ない．

2.4 その他の可視化手法

プロセスツリー，タイムライン，ヒートマップを利用してマルウェアに関連するプロセスの親子関係と各プロセスの API 呼び出しを時系列に配列して表示する以外にも，利用者の分析をサポートするために次のような可視化も検討する．

(1) レーダーチャート

前述のように Cuckoo Sandbox の API 呼び出し履歴にはカテゴリー情報が付加されている．カテゴリーごとの API 呼び出し数からレーダーチャートを作成する．レーダーチャートにより，カテゴリーごとの API 呼び出し数の大小を把握しつつ，マルウェアの特徴を捉えることができる．図 6 にレーダーチャートの例を示す．

Vol.2014-CSEC-66 No.17
Vol.2014-SPT-10 No.17
2014/7/3

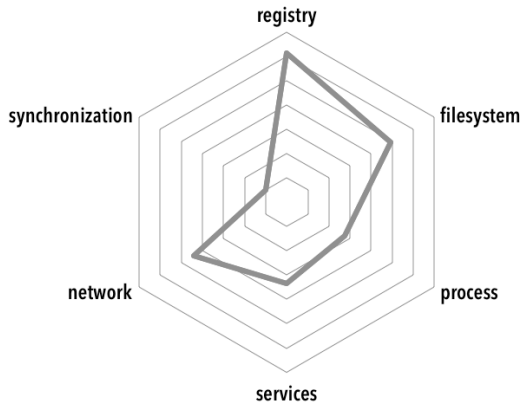

図 6　レーダーチャートの例

(2) **ネットワーク通信**

network オブジェクトにプロトコルごとの宛先情報がまとめられている．当該情報を利用して世界地図上に宛先をプロットすることができる．

```
"network": {
    "udp": [
        {
            "dport": 138,
            "src": "192.168.56.101",
            "dst": "192.168.56.255",
            "sport": 138
        },
・・・中略・・・
    ],
    "irc": [],
    "http": [],
    "smtp": [],
    "tcp": [
        {
            "dport": 41119,
            "src": "192.168.56.101",
            "dst": "192.168.56.1",
            "sport": 8000
        },
```

図 7　network オブジェクトの例

図 8 に宛先情報の可視化の例を示す．世界地図上に宛先情報をプロットするだけでなく，タイムラインで選択したイベントに関連する宛先を強調表示する．

図 8　宛先情報と VirusTotal 検知数の表示例

(3) **VirusTotal**

Cuckoo Sandbox の解析結果には，VirusTotal によるアンチウイルスソフトの検知数が含まれている．

検知数と検査対象アンチウイルスソフト数は，virustotal オブジェクトの positives と total にそれぞれセットされている．

```
"virustotal": {
    "scan_id":
"7c90e90d60371782238d4ed0b3ee90741724c26a4ecd466aba15a1e
a8ddfb0b9-1341833267",
    "sha1": "0ab856b12055aae9c28cda03f5ad910a4e1d70f7",
・・・中略・・・
    "positives": 37,
    "total": 41,
    "md5": "defc586fbd422d466a6bab7dfec48517",
```

図 9　virustotal オブジェクトの例

図 8 にアンチウイルスソフト検知数の表示例を示す．表示例では，検知数と検査対象アンチウイルスソフト数に加え，レベルメーターを模した検知率表示も検討した．

3. まとめ

本研究では，動的解析結果を用い，マルウェアの分析をサポートする可視化手法を提案した．時刻情報付き API 呼び出し履歴を時系列表示することに加え，レーダーチャートや世界地図上にネットワーク通信の宛先情報をプロットする可視化手法についても検討した．

今後は，本研究で提案した可視化を実装し，有効性を確認する．

謝辞　本研究は，総務省情報通信分野における研究開発委託「国際連携によるサイバー攻撃の予知技術の研究開発」により行われた．

参考文献

1)　Anubis - Malware Analysis for Unknown Binaries
https://anubis.iseclab.org
2)　ThreatExpert
http://www.threatexpert.com
3)　Cuckoo Sandbox: Automated Malware Analysis
http://www.cuckoosandbox.org/