メモリフォレンジックを用いた感染端末悪性プロセス検出の実証

湯下 弘祐[1,a)]　小林 和真[1]　門林 雄基[1]

概要：メモリフォレンジックツールを用いることによりアンチウイルスソフトなどでは検出できないメモリ上でのコード注入などを検出することが可能であるが, その解析にはメモリや OS に関する専門的な知識が必要である. そのため, 専門家による犯罪捜査などを除けば, マルウェア感染の検出にメモリフォレンジックツールを使うことは一般的ではない. そこで, 本論文ではメモリフォレンジックのフレームワークである Volatility Framework を用いて非専門家でも悪性プロセスの検出ができるツールを実装し, 感染検出を試みた. 実験の結果, 感染端末 10 個中 5 個で悪性プロセスを抽出し, 9 個の端末で悪性の挙動を確認できた.

キーワード：メモリフォレンジック, 感染検出, Volatility Framework, 悪性プロセス

Using Memory Forensics to Detect Malicious Processes in Infected Devices

Kosuke Yushita[1,a)]　Kazumasa Kobayashi[1]　Youki Kadobayashi[1]

Abstract: Memory forensics tools make it possible to detect suspicious actions such as code injection on memory, which cannot be detected by antivirus-software. However, analyses using memory forensics tools are not generally used except in cases of criminal investigation because it requires a deep understanding of memory and OS. Therefore, I implemented a tool to detect malicious processes in infected devices by using Volatility Framework and tried to detect infections. As the result, I succeeded in detecting 5 malicious processes and 9 malicious actions in 10 infected devices.

Keywords: Memory Forensic, Infected Detection, Volatility Framework, Malicious Process

1. はじめに

昨今, 標的型攻撃をはじめとするサイバー攻撃は増加傾向にある. 例えば警察庁の 「平成 27 年におけるサイバー空間をめぐる脅威の情勢について」[1] によれば, 報告された平成 27 年度の標的型攻撃の件数は平成 26 年度の 1723 件に対し 3828 件と 2 倍以上であった.

このように攻撃が増加する一方で, 攻撃を受ける側はシステム担当者以外はサイバー攻撃に対する十分な知識がないことが多く, 個人で使う端末ではアンチウイルスソフトのアラート以外で自分の端末が感染していることに気づくことは稀である. そのため, インシデント発生の際には初動の遅れにより被害が拡大することが少なくない.

このような状況に対する対処法の一つとして, メモリフォレンジックツールが存在する. メモリフォレンジックツールはアンチウイルスソフトなどによる従来の手法では検出できないメモリ上でのコード注入など不審な挙動を検出することが可能であるが, 解析には専門的な知識が必要であり, 実情として犯罪捜査などの一部の現場でしか用いられていない. しかし, 解析者の思考をアルゴリズム化すれば自動化は可能であると考える.

そこで, 本論文では, その仮定立証の第一歩として, メモリフォレンジックツールのひとつである Volatility Frame-

[1] 奈良先端科学技術大学院大学
Graduate School of Nara Institute of Science and Technology
[a)] mail:yushita.kosuke.yf5@is.naist.jp

work によって感染端末の解析を行い, 悪性プロセスの検出の実証を行う.

2. メモリフォレンジックツール

2014 年に Symantec の幹部から「アンチウイルスソフトは死んだ」という発言があったとおり, 近年ではアンチウイルスソフトだけではサイバー攻撃には対応できなくなっている. しかし, 個人単位でのセキュリティ対策と言えばアンチウイルスソフトとセキュリティパッチの適用をする他ないのが大多数の現状であり, 端末の挙動がおかしくても個人で感染しているかを判断することは難しい.

そこで数年前から, 「ライブフォレンジックにおける有効性の検討及び具体的実施手法の提案」[2] などにもあるようにメモリフォレンジックを用いたマルウェア検出などの技術が一部で提唱されており, 犯罪捜査におけるデジタルフォレンジックの一部として広く使用されている. メモリフォレンジックの特徴として, メモリからしか入手できない情報を入手できること, 改ざんへの耐性が高いこと, ディスクイメージ全体を取る必要はないため見る容量は少なくて済むことなどが挙げられる. これらの点は, マルウェアの感染を検出する上で相性がよい. また “The Art of the memory forensics”[3] でも紹介されている Volatility Framework のような無料のフォレンジックツールも存在し, 誰でも手の届くものとなっている.

しかしこのようなツールを適切に利用し, 感染を検出するには, OS やメモリなどに対する高度な知識が必要とされるため, 入手は容易であるが使いこなすには敷居が高い. 過去に JPCERT/CC の記事 [4] で紹介されているように標的型攻撃を自動で検出するためのツールの配布が行われたりはしているものの, こういったツールはあくまで特定のマルウェアにのみ焦点をしぼったものであり, アンチウイルスソフトのように汎用的に自動で検出する用途で作られているものは存在しない. 商用ソフトではメモリ解析を利用するソフトも一部存在するが, それらは専門性が高く費用も高額である例が多い.

以上のことから, 誰でも簡単に感染検出に使えるようなメモリフォレンジックのツールがあれば, 一般の人々でもアンチウイルスソフトとセキュリティパッチ以外のセキュリティ対策を手に入れることができると考える.

3. 悪性プロセスの検出に用いたツール

3.1 Volatility Framework

Volatility Framework(以下, Volatility) は Volatility Foundation[5] により提供されているメモリフォレンジックのフレームワークである. ツールキットは python ファイルで作られており, 豊富なコマンドと引数を兼ね揃えている. プラグインで提供される幅広い解析機能はメモリ解析の基礎の域を超えており, メモリ内のレジストリハイブ

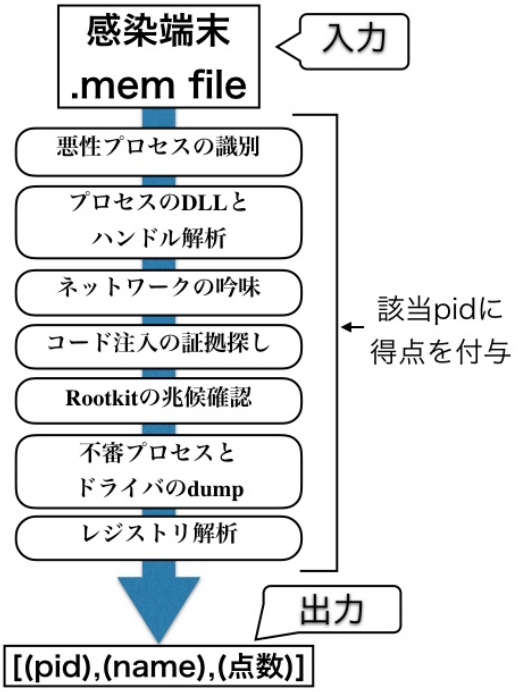

図 1 解析手順

抽出やプロセスインジェクションの抽出, Yara シグネチャによるスキャンなども可能である [6].

3.2 Volatility の解析手法

Volatility には多くの機能があるが, 機能が多すぎて予備知識がないと使いこなすことは難しい. sans.org のチートシート [7] によれば, 大まかに以下のような要素の解析が行われる.

(1) 悪性プロセスの識別
　psscan や pstree で隠しプロセスを調査
(2) プロセスの DLL とハンドル解析
　getsids, handles などを用いてハンドル情報などを調査
(3) ネットワークの吟味
　netscan で通信先を調査
(4) コード注入の証拠探し
　malfind や ldrmodules でコード注入を検出
(5) Rootkit の兆候確認
　psxview や ssdt で隠しプロセスやシステムサービスを調査
(6) 不審プロセスとドライバの dump
　dlldump などで特定の pid の DLL を抽出
(7) レジストリ解析
　hivelist や printkey を用いてレジストリを調査

3.3 実装した python プログラム

Volatility が持つ感染を検出するための機能を組み合わせて, マルウェアが感染した RAM イメージから作成した mem ファイルの検査を多角的に行う python プログラ

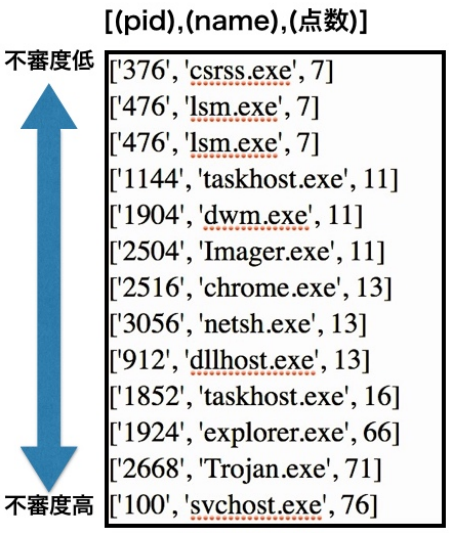

図 2 出力例

ム作成した. 使用する Volatility のコマンドは sans.org や downloads.volatilityfoundation.org で提供しているチートシート [7][8] を参考に選んだ. マルウェアに感染した場合, 最終的には悪性プログラムを排除する必要があるため, プロセス ID(以下, pid) を出力するコマンドを中心に実装した.

このプログラムはコマンドの引数として mem ファイルを入力すると, 図 1 に示す解析手順を全ての pid に対して行う. 不審な挙動が多い pid ほど点数が高くなるように設定し, 最終的には不審プロセスの候補リストとして図 2 に示すようなリストを出力する.

図 2 で示すリストは抽出した不審プロセスのリストであり, pid, プロセス名, 点数からなる. ここで言う点数とは, Volatility の各コマンドで得られた情報に基づくプロセスの不審さを表す点数である. リストの下の方ほど不審度が高く, 悪性プロセスである可能性が高い.

3.4 使用したコマンド一覧

解析に使用したのは以下のコマンドである. 出力に pid を含むコマンドは, プロセスに悪性挙動を認めると pid に対して点数を与えるように設定した. 出力に pid を含まないコマンドは, 端末が感染しているかどうかの判断を補完する目的で用いた.

3.4.1 出力に pid を含むコマンド

pid ごとの不審さを判断するために使用

(1) psscan
プロセスリストの作成に使用

(2) psxview
隠しプロセスなどの検出に使用

(3) netscan
通信先の特定に利用. malwaredomainlist.com[9] と ipvoid.com[10] にクエリを出し, 悪性ドメインか否かを判断

(4) malfind -p PID –dump-dir=.
注入が行われたプロセスをダンプ. ダンプされたものは手動で VirusTotal へ送信

(5) malfind -D /tmp
/tmp に残っているファイルから悪性のものを抽出

(6) printkey -K Microsoft \\ Windows \\CurrentVersion\\ Run

(7) printkey -K Software\\Microsoft\\Windows\\CurrentVersion\\Run
HKLM と全ユーザに向けの runkey を持つものを抽出

(8) getsids |grep -e Domain -e Enterprise
プロセスが domain か Enterprise を持たないかの確認

(9) privs --silent --regex=debug | grep Present,Enable | grep -v Default
デフォルト以外でデバッグ権限を付与されているものを抽出

(10) dlldump -p PID -D ./ --fix --memory
注入が行われた dll をダンプし, そこから更に UNKNOWN を探す

(11) threads -F OrphanThreads |grep StartAddress
カーネルルートキットの隠しスレッドを抽出

(12) handles -t Files |grep \\Device\\RawIp\\0 raw
ソケットのプロセスを発見

(13) ldrmodules |grep False
リンクされていない DLL を抽出

3.4.2 出力に pid を含まないコマンド

pid 以外の要素で, 端末感染の判断に使用

(1) callbacks |grep UNKNOWN
不明な callbacks を抽出

(2) ssdt|grep -v nots|grep -v win32k
SSDT の照合

4. 悪性プロセス検出の評価実験

4.1 実験概要

各感染端末から FTK imager[11] で抽出した Windows7 の mem ファイルに対して, 3 章で作成したツールを使って解析を行う. 各プロセスに対して, ツールで検出された不審挙動に応じて点数を与え, プロセスの不審度のランク付けをする. ランキング上位ほど不審なプロセスとみなす.

評価の基準として, 各端末に感染させた本物の悪性プロセスがランキング上位にくるほど, 正しく検出されているものと判断する.

4.1.1 調査端末を Windows7 にする理由

2016 年 7 月 3 日に掲載された OS のシェアに対する記事 [12] によると, 2016 年 6 月時点での日本国内における OS のシェアは Windows7 が圧倒的に多く, 49.05%である. 2016 年 7 月末まで Windows10 への無料アップデート期

間があったにせよ, なお多くのユーザがしばらくは Windows7 を使い続けると思われる. また Volatility の機能のうち, Network 検査機能など重要ないくつかの機能は, まだ Windows8 以降には対応していないものがある. 以上のような理由から今回は Windows7 を調査対象の端末とした.

4.1.2 解析環境

解析マシンのスペックは下記のとおりである.

- OS : ubuntu16.04LTS
- メモリ : 5GB
- HDD : 50GB
- ネットワーク : NAT, ホストオンリーアダプター
- プログラム : Volatility2.5
- 言語 : python2.7.12

各端末からキャプチャした mem ファイルを解析用の ubuntu16.04LTS に送り, 今回作成したツールで解析を行った.

4.2 mem ファイルの作成

4.2.1 非感染端末の作り方

VirtualBox 上で Windows7SP1x64 を作成し, 作成後 InternetExplorer で GoogleChrome を取得した後, 速やかにメモリ取得ツールである FTK imager でメモリをキャプチャして非感染状態の mem ファイルを作成した. キャプチャ後の検査では不審な動きは見当たらなかった.
端末のスペックは下記のとおりである.

- OS : Windows7SP1
- メモリ : 1GB
- HDD 容量 : 25GB
- ネットワーク : NAT
- キャプチャまでに使用したプログラム :
  InternetExplorer, FTK imager, google chrome
- キャプチャまでにインストールしたソフト :
  FTK imager, google chrome

4.2.2 感染端末の作成方法

前節で作成した非感染 Windows7 を元に, クローンを 10 個作成した. 各クローンに 1 種類ずつ悪性検体を感染させ 10 個の感染端末を作成した. 悪性検体は malwr.com[13] から調達し,Windows 上で通信の確立やダウンロードの開始, malwr.com に記載してあるプロセスの起動などの悪性挙動が行われていることを確認した後に FTK imager でメモリイメージをキャプチャし, mem ファイルを作成した. 悪性検体の名前は, 検出時に識別しやすいようにプログラム名を全て trojan.exe に変更した.
悪性検体を選ぶ際には以下のことを基調として選んだ.
(1) name に trojan や rootkit などの文字がある
(2) Windows7 で動作する検体である
(3) アンチウイルスソフトでの検出率が高い (例外:mem9)
(4) 外部との通信が見られる (例外:mem3, 10)

表 1 コマンドごとの得点表

command	point
psxview	False ごとに+1 点
netscan	悪性ドメインで+30 点
malfind -p PID --dump-dir=.	該当ごとに+30 点
malfind -D /tmp	該当ごとに+20 点
printkey -K （中略）\Run	該当ごとに+10 点
getsids grep -e Domain -e Enterprise	該当ごとに+10 点
privs --silent --regex=debug （中略）	該当ごとに+15 点
dlldump -p PID -D ./ --fix --memory	UNKNOWN ごとに+20 点
threads -F OrphanThreads （中略）	該当ごとに+15 点
handles -t Files | grep（中略）	該当ごとに+5 点
ldrmodules | grep False	該当ごとに+5 点

表 2 各 mem の不審 pid 上位 3 個

	top1	top2	top3
非感染	explorer(86)	chrome(66)	wmpnetwk(56)
mem1	svchost(76)	**trojan(71)**	explorer(66)
mem2	svchost(95)	chrome(65)	explorer(65)
mem3	regsvr32(365)	regsvr32(265)	**trojan(65)**
mem4	svchost(116)	**trojan(66)**	explorer(66)
mem5	SrpnFiles(10185)	svchost(76)	taskhost(15)
mem6	svchost(75)	chrome(65)	explorer(65)
mem7	**trojan(166)**	svchost(117)	svchost(117)
mem8	svchost(256)	svchost(186)	svchost(106)
mem9	msiexec(160)	msiexec(120)	msiexec(100)
mem10	chrome(316)	svchost(76)	explorer(66)

(凡例) : .exe を除いたプログラム名（得点）

5. 実験結果

5.1 検出の可否

5.1.1 感染端末

python プログラムで悪性挙動を検出する度に, 表 1 に示す得点表で点数を付与し, 各 mem ファイルごとの評価を行った. 点数の重み付けは本実験の前の下調べの際の傾向に基づき, コード注入や悪性ドメインとの通信など明らかに悪意または不審なものに関するものは点数を高くし, 悪性ではないプロセスからでも頻繁に見られるものは点数を低く設定した.

各 mem ファイルの pid 上位 3 個は表 2 のとおりである. 表 2 のプログラム名 trojan は, malwr.com からダウンロードしてきた悪性検体に対してこちらで付与した名前であり, それぞれは別々の検体である. （以後, 感染させた悪性検体は便宜上 trojan.exe と呼ぶ ）

表 2 を見ると感染させた trojan.exe が不審プロセスの上位 3 個に入っているのは mem10 個中 4 個である. mem8 では trojan.exe 本体こそ 86 点で上位 5 番目であったが, mem8 の上位 3 個は pstree で確認したところ全て trojan.exe の子プロセス, 孫プロセスであるため不審プロセスとして検出していること自体は間違いではない. よって, 10 個中 5 個は正解として差し支えないと考える.

表 3 端末ごとの悪性挙動一覧

検出項目	psscan で pid を検出	netscan で悪性 domain を検出	VirusTotal で悪性 dump を検出	trojan.exe が top3 内	AV 検出率	不審に感じるその他の特徴	処理に要した時間
非感染端末	-	×	×	×	-		16m51s
mem1	○	×	×	○	45/48	trojan の子プロセス発生	16m24s
mem2	×	×	×	-	47/55	デフォルト以外のデバック権限検出	22m19s
mem3	○	○	○	○	26/56		17m38s
mem4	○	×	×	○	39/54		13m59s
mem5	×	×	×	-	26/53	malfind 大量検出	16m20s
mem6	×	○	×	-	0/56		16m39s
mem7	○	○	×	○	45/50		25m4s
mem8	○	×	○	×	40/56	trojan の子プロセス発生	9m41s
mem9	×	×	○	-	41/50	デフォルト以外のデバック権限検出	6m31s
mem10	×	○	×	-	47/55		16m53s

また psscan で trojan.exe が検出できなかった mem2, 5, 6, 9, 10 についても, 表 3 に示すとおりの内容で悪性の挙動を確認できた. しかし, mem2 に関しては「操作中にわかりやすく感染した」ことの他は, 悪性であるという直接の証拠はなく,privs で検出したものの, 不審度の低いプロセスであったことから検出には失敗したとみなした.

各 mem の trojan.exe を検出できたコマンドは表 4 のとおりである. psscan で検出ができなかったものは pid がないため, malfind, printkey などのコマンドでも検出ができなかった. handles や threads のコマンドも実施したが, いずれの mem ファイルも malfind, printkey, psxview, ldrmodules, getsids, privs 以外では不審な挙動は検出されなかった.

表 5 は同様のことを trojan.exe 以外の最も得点が高かった pid について示したものであるが, 同じ傾向であった.

5.1.2 非感染端末

非感染端末である Windows7 の explorer や chrome で高い点数になったのは, explorer.exe や chrome.exe に対する malfind コマンドで 1,2 回ずつ検出されたためである. しかし, この dump を VirusTotal で確認したところ, 不審な検体は発見されなかったことや検出された通信先と timeliner 上のコマンドに全て心当たりがあったことから感染していない端末であると考える.

5.2 各 mem ごとの調査結果

検体の紹介のため, 各検体に関する, 表 3, 表 4, 表 5 で示した以外の特徴を示す.

5.2.1 mem1 の特徴

アンチウイルス検出率：45/48

name: Trojan_490d8(中略)_9831.exe_

md5: 32fe14296f34ba25148ebbb512a40416

(1) trojan.exe を親プロセスする子プロセス (netsh.exe) が存在するが, psxview, printkey による検出のみで点数は 40 点であった.

(2) malfind で dump したものを VirusTotal に入れたが, 悪性検体は検出されなかった.

5.2.2 mem2 の特徴

アンチウイルス検出率：47/55

name: 0472.Trojan-Downloader.Win32.Agent.exe

md5: 6de4e7f74dd664105ac5fe4454001e94

(1) trojan.exe をダブルクリックしたら, わかりやすく感染しファイルを次々にダウンロードした.

(2) Windows security alert という偽の警告を発して, クリックを促していた.

(3) malfind で dump したものを VirusTotal に入れたが, 悪性検体は検出されなかった.

(4) 30 得点の WerFault.exe がデフォルト以外のデバッグ権限を保有していた.

5.2.3 mem3 の特徴

アンチウイルス検出率：26/56

name: FakeFiredfoxTrojan.bin

md5: 49a748e9f2d98ba92b5af8f680bef7f2

(1) trojan は 2192 の他にも 2428 の pid でも活動していた.

(2) trojan.exe は firefox の偽 exe.

(3) 悪性通信をしているのは regsvr のみ.

(4) malfind で dump したものを VirusTotal に入れたところ, trojan 系の検体が検出された.

5.2.4 mem4 の特徴

アンチウイルス検出率：39/54

name: Filecoder.DG.trojan - Copy.exe

md5: 7fb4a1161c4c2f38e9ceecd7b0c77030

(1) malfind で dump したものを VirusTotal に入れたが, 悪性検体は検出されなかった.

表 4 trojan.exe を検出したコマンド

検出コマンド	psscan	malfind	printkey	psxview の False	ldrmodules	getsids	privs	その他
mem1	○	○	○	×	×	×	×	×
mem2	×	-	-	-	-	-	-	-
mem3	○	×	×	×	×	×	×	×
mem4	○	○	○	○	×	○	×	×
mem5	×	-	-	-	-	-	-	-
mem6	×	-	-	-	-	-	-	-
mem7	○	○	○	○	○	○	×	×
mem8	○	○	○	○	○	○	×	×
mem9	×	-	-	-	-	-	-	-
mem10	×	-	-	-	-	-	-	-

表 5 最も得点の高い pid を検出したコマンド

	プロセス名 (点数)	psscan	malfind	printkey	psxview の False	ldrmodules	getsids	privs	その他
mem1	svchost(90)	○	○	×	○	○	×	×	×
mem2	svchost(95)	○	○	×	×	○	×	×	×
mem3	regsvr(300)	○	○	○	×	○	○	×	×
mem4	svchost(116)	○	○	×	○	○	×	×	×
mem5	SrpnFiles(10185)	○	○	○	×	○	○	×	×
mem6	svchost(75)	○	○	×	×	○	×	×	×
mem7	svchost(117)	○	○	×	○	×	×	×	×
mem8	svchost(256)	○	○	○	○	○	○	×	×
mem9	msiexec(160)	○	○	○	○	○	○	○	×
mem10	chrome(316)	○	○	○	○	×	○	×	×

5.2.5 mem5 の特徴

アンチウイルス検出率：26/53

name: Trojan.exe

mad5: 1b186bb4726fad837b1d835d491e0050

(1) SrpnFiles.exe は malfind で約 300 回検出された.

(2) malfind で dump したものを 50 個ほど VirusTotal に入れたが, 悪性検体は検出されなかった. dump ファイルの総数は約 300 個で, 今回観測した範囲では通常 5-20 個程度であり群を抜いて多かった.

5.2.6 mem6 の特徴

アンチウイルス検出率：0/56

name: rootkitremover.exe

md5: b3f6cd53990f49e7e1afcd6b21b9822f

(1) chrome.exe(65 点) から netscan で悪性ドメインとの通信を検出.

(2) malfind で dump したものを VirusTotal に入れたが, 悪性検体は検出されなかった.

(3) 悪性ドメインは CBL[14] の判定によれば spam 送信トロイなどに感染した際に見られる ip アドレスであった.

5.2.7 mem7 の特徴

アンチウイルス検出率：45/50

name: 0098.Trojan-Downloader.Win32.Agent.exe

md5: 4270cd741b06caedbc77a8cf74bcd62c

(1) malfind で dump したものを VirusTotal に入れたが, 悪性検体は検出されなかった.

(2) 悪性ドメインは MyWOT[15] の判定では trojan や spyware を配布するサイトであった.

5.2.8 mem8 の特徴

アンチウイルス検出率：40/56

name: Trojan.Kryptik.exe

md5: 933f66c004c1c03d6b607f69499dd4f6

(1) malfind で dump したものを VirusTotal に入れたところ, trojan 系の検体が検出された.

(2) 得点の高い svchost.exe はいずれも trojan.exe の子, 孫, ひ孫プロセスであった.

5.2.9 mem9 の特徴

アンチウイルス検出率：41/50

name: 0880.Trojan-Banker.Win32.Banker.exe

md5: 133c9455e63fd22dc694ef026ce41afd

(1) 他の端末よりも msiexec.exe の数が多かった.(検出された mseiexc.exe は 7 個で, 今回観測した範囲では通常 0 個)

(2) malfind で dump したものを VirusTotal に入れたところ, trojan 系の検体が検出された.

(3) 最高点の msiexec.exe の他, 複数のプロセスがデフォルト以外のデバッグ権限を保有していた.

5.2.10 mem10 の特徴

アンチウイルス検出率：47/55

name: rootkit.exe

md5: 5782e28edb01bde30d2b9e754dfca38a

(1) chrome.exe(316) の悪性ドメインは CBL[14] の判定によれば spam 送信トロイなどに感染した際に見られる ip アドレスであった.
(2) malfind で dump したものを VirusTotal に入れたが, 悪性検体は検出されなかった.

5.3 その他の発見

以下では, その他に本調査により発見したことについて述べる.

出力に pid を含まないコマンドに関しては, 今回収集した範囲では ssdt, callbacks で検出されたものはなく, あまり活用できなかった. ssdt では win32k や ntoskrnl.exe 以外で動くものを抽出できるが, 今回は該当するものがなく, callbacks も Detail が UNKNOWN のものはなかった.

また解析所要時間については, 解析の所要時間は速いものでは 6 分台, 遅いもので 25 分程度かかり有意な差が見られた. 全てのプロセスやリモートアドレスに対して確認を行うため, 感染したマルウェアの種類によって大きく差が出る結果になったと考える.

6. 考察と結論

6.1 考察

6.1.1 ツールの有効性

今回は Volatiliy を用いた python プログラムによる悪性プロセス検出の実証を行った. 感染した Windows7 の 10 端末で実験を行ったところ, psscan で trojan.exe を検出できたものが 5 端末あり, そのうち作成したツールにより trojan.exe を不審プロセスの上位 3 個以内に検出できたものは 4 端末であった. mem8 の上位 3 個全てが trojan.exe の子プロセスであることを考慮すれば, 10 個中 5 個は検出できている. よって, Volatility を用いて不審さに対し点数を付与して悪性プロセスを検出するという試み自体はある程度は的を得ていると思われるので, 概念としてこの種のツールは有効であると考える.

しかし psscan で検出できなかった場合は Volatility での pid を絡めた調査が行えないため, 悪性ドメインとの通信や別の不審プロセスの dump で得たものから判別する他ない. そのため, 効果が限定的になることも事実であり, その点において工夫が必要である.

また現状では端末の感染や挙動を知ることが出来, その候補もわかるが最終的な「答え」となる悪性プロセスがどれであるかを知るために深い知識技能が必要である点は改善を要する.

6.1.2 psscan で検出できなかった理由とその対処

今回感染させた trojan.exe は, psscan では確認できなかったものでも, 検体の存在自体は timeliner や mftparser などのコマンドに grep trojan.exe を絡めることで確認することができた. この手法は, 感染している検体名に心あたりがあるときは有効であるが, そうでないときは虱潰しに調べるしかなく, その他の検出方を編み出す必要がある.

6.1.3 感染検出の可否

今回の結果から, pid がわかっており, なおかつコマンドでカバーしている範囲の注入や改ざんが行われた場合は, 自動化したツールによって端末自体の感染を検出することが可能であると考える. また得点の重み付けが出力される不審プロセスのランキングに大きく影響を与えるため, 今回使用しなかったコマンドも含めて得点の重み付けの最適化を行うと, より良い検出率につながると考える.

6.1.4 偽陽性の抑止

今回, 非感染端末として用いた Windows7 の端末では, malfind や printkey によりいくつかのプロセスが検出されたため結果として累計の点数が感染端末並みに高くなってしまった. このようなケースによる偽陽性を防止するためには, ホワイトリストを用いたり, 何かのコマンドで検出されても他の何か特定のコマンドで検出されない限りは加点しないなどの工夫が必要である.

6.1.5 検出できなかったコマンド

今回検査を行ったコマンドの中で検出できなかった handles などのコマンドは, raw ソケット通信を見たりするものでマルウェアとしてはさして特別なものではないと考える. ただし, 今回の実験においては当初予想していたよりも検出されなかったため, コマンドの選択に関してはもう少しデータを集める必要がある.

6.2 結論

上記の考察から, Volatility Framework を用いた感染端末の検出は可能であり, 自動化による悪性プロセス検出をするツールの有効性はある程度確認できた. しかし, 現状ではまだ悪性挙動をするプロセスのリストが得られただけの状態であり, そこから真に止めるべき悪性プロセスを突き止めるには, まだ知識技能が要求される段階である. 今後, 精度を高めていくための工夫として, 使用するコマンドと得点の重み付けの最適化や, 偽陽性防止のためにホワイトリストや検出コマンドの組み合わせによる加点などの工夫が必要である.

また psscan で取得できなかったプロセスに関しては, timeliner や mftparser などを用いて過去に実行されたプログラムを発掘するか, 常駐する検体が対象であれば定期的に RAM を抽出してプロセスを監視する必要がある.

6.3 今後の課題と展望

6.3.1 実装上の改善点

(1) コマンドの取捨選択
メモリフォレンジック自体の問題にも関連するが, 昨今のメモリ量増大により今後は解析時間が長くなることが予想される. 例えば今回使用した Windows7 のメ

モリは 1GB であるが, 最近では 16GB や 8GB のものが主流になってきている. そのため, コマンドを必要十分な範囲で厳選し, 且つ見るべきプロセスも初期の調査で不審な要素が全くなければその後の捜査では省くなどの工夫が必要になってくると思われる.

(2) アンチウイルスソフト等との関係
セキュリティの非専門家でもできる対策として, アンチウイルスソフトとセキュリティパッチの適用が十分に普及していると仮定した場合, 網羅性の観点からメモリフォレンジックツールはそれらがカバーできない部分をまず重点的に見るべきである. アンチウイルスソフトで検出できなかった今回の mem6 の検出はそのような例でありメモリフォレンジックツールの利点を活かせた形だと考える.

(3) psscan で見つからない場合
trojan.exe が起動したにも関わらず, psscan で検出できない pid があった. pid を観測できなかった実際の理由は不明であるが, メモリの容量の関係上, 稼働させた全てのプロセスが pid として残るわけではないため, 消えたものと推測する. しかし, 定期的にメモリを確認すれば, rootkit や trojan などの常駐型検体は定期的に悪性のプロセスが走るため, 検出できると考える. その他にも, netscan や mftparser, timeliner などでのコマンドで psscan で漏れているプロセスを拾ってくることも可能であるため, psscan で取得した pid 以外のプロセスも収集する工夫が必要である.

6.3.2 APT などの未知の脅威の検出

今回メモリフォレンジックツールを用いたことの理由として, メモリフォレンジックであれば DLL へのコード注入などを検出できるため, ログを見ても判別がつかないような攻撃も発見できるかもしれないことが挙げられる. ツール自体も Volatility Framework などであれば無料で入手することができるため, このようなアンチウイルス以外のツールを使ってマルウェアに感染しているかを確認できる手段が存在することは重要である.

6.3.3 解析所要時間の短縮

解析時間を短縮するためにはいくつかの手法が考えられるが, python アルゴリズムを洗練化することと豊富な解析リソースを確保することは万人にとって有効な手法である. 環境が許せば, CPU をクラスタ化したり, I/O の速度を速めるためにストレージを交換してみるなどの工夫が求められる.

7. おわりに

現状として, 個人単位でのセキュリティはアンチウイルスソフトとセキュリティパッチの他は, あまり普及していないのが現状である. 少し調べれば, 便利なツールが世の中にたくさんあることに気づくことはできるが, それらが普及していない理由の一つとしては入手が簡単でも使うための敷居が高いということが挙げられる.

本研究の最終的な目標は, そういった課題を解決するための一つの方向性として, 自動化されたフォレンジックツールによって感染を検出できるようにし, アンチウイルスソフト以外の対処法を誰でも使えるようにすることである. 今回は pid を絡めて悪性プロセスの検出を試みたが, 課題は残るもののその有効性に関してはある程度実証できたものと考える. 今後は解析の精度を上げるとともに, 未知の検体の検出, 解析時間の短縮化に取り組む.

参考文献

[1] 警察庁, 「平成 27 年におけるサイバー空間をめぐる脅威の情勢について」, https://www.npa.go.jp/kanbou/cybersecurity/H27_jousei.pdf, 2016 年 8 月 6 日アクセス.

[2] 今野直樹, 田中英彦. 「L-017 ライブフォレンジックにおける有効性の検討及び具体的実施手法の提案 (L 分野: ネットワーク・セキュリティ, 一般論文).」, 情報科学技術フォーラム講演論文集 14.4 (2015): 205-212.

[3] Ligh, Michael Hale and Case, Andrew and Levy, Jamie and Walters, Aaron, “The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory”, Wiley Publishing(2014)

[4] JPCERT/CC, 「標的型攻撃に使われるマルウエアを検知する Volatility Plugin(2015-10-28)」, https://www.jpcert.or.jp/magazine/acreport-aptscan.html, 2016 年 8 月 6 日アクセス.

[5] Volatility Foundation, http://www.volatilityfoundation.org, 2016 年 8 月 6 日アクセス

[6] Jason T. Luttigen and Matthew Pepe and Kevin mandia, 「インシデントレスポンス 第 3 版」, 日経 BP 社 (2016):432.

[7] SANS, “SANS COMPUTER FORENSICS and INCIDENT RESPONSE Memory Forensics Cheat Sheet v1.2”, https://digital-forensics.sans.org/media/memory-forensics-cheat-sheet.pdf, 2016 年 8 月 6 日アクセス

[8] The Volatility Foundation,“CheatSheet 2.4 Edition”, http://downloads.volatilityfoundation.org/releases/2.4/CheatSheet_v2.4.pdf, 2016 年 8 月 6 日アクセス

[9] MALWARE DOMAIN LIST, https://www.malware-domainlist.com, 2016 年 8 月 6 日アクセス

[10] IPVoid, http://www.ipvoid.com, 2016 年 8 月 6 日アクセス

[11] ACCESSDATA, “FTK imager”, http://accessdata.com/product-download, 2016 年 8 月 9 日アクセス

[12] マイナビニュース, 「Windows7 と Windows10 が増加, 6 月 OS シェアランキング」, http://news.mynavi.jp/news/2016/07/03/035/, 2016 年 8 月 6 日アクセス.

[13] malwr, https://malwr.com, 2016 年 8 月 6 日アクセス

[14] SPAMHAUS, “COMPUTING BLOCK LIST”, http://www.abuseat.org, 2016 年 8 月 6 日アクセス

[15] MyWOT, https://www.mywot.com, 2016 年 8 月 6 日アクセス