Vol.2019-IOT-44 No.7
2019/3/7

メモリフォレンジックを用いたランサムウェア解析の提案

Proposal of Ransomware analysis using Memory Forensics

深井亨† 柿崎淑郎† 広瀬幸† 猪俣敦夫†

概要：ランサムウェアによるファイルの暗号化による被害が世界各地で増大しており，ランサムウェアの特徴を早期に解明することは喫緊の課題である．ランサムウェアの特徴を把握することにより，感染前に対策の効率が上がると考えられる．既存の研究では VirusTotal レポートを用いてランサムウェアと正規のソフトウェアの差異を比較した解析であり，ランサムウェアの挙動やプロセスの特徴は明らかになっていない．そこで，本論文ではメモリフォレンジックを用いたランサムウェアの解析として Volatility Framework 等のツールを使用した手法を提案し，その有効性を評価するために仮想環境上でランサムウェアを実行し，メモリイメージの取得をして解析を行った．その結果を報告する．本提案手法を活用することにより，今後のランサムウェアによるファイル暗号化の被害を減少させるための手段として用いられることが期待される．

キーワード：ランサムウェア，マルウェア対策，デジタルフォレンジック，メモリフォレンジック

1. はじめに

ランサムウェアによってファイルが暗号化されてしまうといった被害が世界各地で増加している．インターネットが一般にも普及した現在，企業だけではなく一般人も被害の対象となりえる．トレンドマイクロが発表した[1]ランサムウェア WannaCry の 2017 年間検出台数は，国内でおよそ 1 万 8,500 台，全世界では 32 万 1,800 台に上る．病院，鉄道などの業種における特有環境での被害事例も国内外で多く確認された．また，2017 年に確認された新種のランサムウェアは 327 種類と，2015 年の 29 種類と比較しても 10 倍以上となる．全世界で確認したランサムウェア攻撃件数は 2016 年の 10 億 7,800 万件から 2017 年には 6 億 3,100 万件と大幅に減少はしている．しかしながら，億単位で，かつ世界各地でサイバー攻撃の一つとしてランサムウェアは定着していると言える．

ここ数年で，ランサムウェアに対する対策等が執られて来ている．中でもトレンドマイクロは，AI を活用した機械学習型スキャンを導入し，ランサムウェアの亜種など未知の脅威に対し，迅速な防御を実現しようとしている．また，インターネットで検索すれば，無料の対策ソフト等も少なからず存在している．Microsoft 社が 2015 年にリリースした Windows10 は，ランサムウェア等の脅威からファイルを保護し，攻撃を受けた場合には，ファイルを復元する機能を有する Windows Defender が備わっている．

このことからもランサムウェアは，世間的に認知されていることが窺がえる．しかしながら，先程も述べた通り，未だ世界各地にて，ランサムウェアの亜種が増え続けている．いくらランサムウェアへの対策をしたところでも，その対策を掻い潜るような形で感染してしまう．このような背景から，ランサムウェアの被害を減少させるために，ランサムウェアの特徴を知る必要があると考える．何故なら，ランサムウェアの特徴を把握することで，感染前に対策の効率が上がると考えられるためである．

本論文では，これまでに多くの亜種が登場しているランサムウェアを使用し，端末に感染させる．感染した際に，どのようなプロセスが起動したのか，CPU がどのように変化したのかを捉える．それを基にメモリイメージを取得した上で，メモリフォレンジックのツールを用いて解析を行い，ランサムウェアの特徴の情報を取得する手法を提案する．本論文では，解析環境を用いるため，ランサムウェアによる感染拡大の可能性を下げることができる．実験の際には，環境設定を間違ってしまうと感染してしまうので，注意が必要である．

2. メモリフォレンジック

メモリフォレンジックとは，端末の揮発性のメモリを解析する技術である．HDD などの補助記憶装置に残されない情報であったとしても，メモリ上であれば残っている可能性が有る．また，メモリイメージの取得ができればオフラインでの解析が可能となるので，攻撃者によって改竄される恐れもない．しかしながら，欠点もいくつか挙げられる．揮発性の情報のため，時間経過や電源を落としてしまうと必要な情報が取得できない可能性が有る．そして，メモリフォレンジックで一番の欠点となるのが，解析をするメモリイメージの取得のタイミングの難しさである．取得したタイミングによっては，解析対象のプロセスが一つも取得できていないことも有り得る．また，実験を行うメモリ容量によってもプロセスが取得できないことが有った．本論文の実験をする上でも，そのような問題点が見受けられた．

3. 関連研究

ランサムウェアによって暗号化されたファイルについて山本らが FTK Imager および Volatility Framework(以降 Volatility)という 2 つのツールを用いて復号できる可能性が

あるのかを調査報告している[2]．山本らは FTK Imager でメモリイメージの取得を行った上で，取得したメモリイメージを Volatility で解析を行い，ファイルの暗号化に使われた鍵データを取得する手法を提案している．しかし，取得できた鍵データは, WannaCry に備えられている一部の暗号化されたファイルのみを復号するための秘密鍵であった．この秘密鍵は，攻撃者が予め WannaCry に埋め込んでいる鍵データであり，金銭を支払えば全てのファイルが復号できるという攻撃者から被害者への示唆だと考えられる．また，鍵データを得るために素数の取得も試みていたが，取得できなかった．その要因として，鍵生成後に CryptDestroyKey 関数が実行され, それによって鍵データが破棄され，メモリを 0 で上書きをしてしまったために鍵データが残っていなかったと山本らは推測をしている. また，動的解析とは異なり，処理の途中でのメモリ上の情報を取得することができないため，一部情報が欠けてしまったのではないかとし，メモリ上の動きを常駐して監視するツールが必要だとしている．また，暗号化処理が実行されるタイミングでのメモリイメージを取得できるツールの開発が必要だと明記している．この文献で使用された検体は WannaCry とその亜種による実験であったため, 他のランサムウェアに感染した場合にも同じ手法が適用できるのかが未知ではあるが，ランサムウェアの種類によっては暗号化処理のタイミングやアルゴリズムも異なることは予測される．また，WannaCry は他のランサムウェアとは異なり，身代金を要求する GUI に Decryptor という機能が実装されているため, 比較的良心的な構造をしている. しかしながら，他のランサムウェアもそうとは行かない．このことからもメモリイメージから鍵データの取得を行い，復号をするということは難しいと考える．そこで，本論文では同じツールは使用するが，復号することは目的とせず，ランサムウェアへの対策として解析を行い，挙動や特徴の情報を取得することを目的とする．

ランサムウェア検知のために，重田らが VirusTotal を用いて特徴解析を行った上で，その結果を調査報告している[3]．この文献では，ランサムウェアの特徴を利用することで，検知手法の改良が可能ではないかと考え，検知手法の精度向上のため，ランサムウェアの特徴の調査を行っている．調査方法としては，マルウェア検査をオンライン上で行うことができる VirusTotal を用いている．VirusTotal 上に過去にアップロードされたファイルの検査結果のレポートに含まれる項目別の比較を行なっている．比較した結果からランサムウェアと正規のソフトウェアの差異を明らかにし，特徴を模索している．しかし，この文献では，ソフトウェアの差異から特徴を模索しているため，ランサムウェアの挙動やプロセスの特徴が明らかにされていない．そのため本論文では，挙動やプロセスに着目し，解析を行う．

また，竹林らが TrueCrypt というツールを用いた暗号化ファイルの復号技術を開発し，その検証を行った[4]．FTK Imager Lite でメモリイメージの取得を行った上で取得したメモリイメージを，aeskeyfind[5]を用いて解析した．その結果，暗号化されたファイルを復号するのに成功している．しかし，この文献は TrueCrypt で暗号化されたファイルのみの復号手法であり，その他の手法で暗号化されたファイルの復号について明記はされていない．また，TrueCrypt の最新版では脆弱性にパッチが当てられてしまっているため, aeskeyfind による鍵取得ができなくなってしまっており，そのため, 実験を行うためには旧版を用意する必要が有る．さらに暗号化されたファイルの復号をする上で，TrueCrypt の GUI 上から鍵データを入力し, 暗号化されたファイルを復号するための機能はないため，GitHub 上からソースコードを取得し，一部のソースコードの修正等を行い，機能を実装する必要もあるために手間が掛かってしまう．

4. 提案手法

4.1 概要

本論文では，メモリフォレンジックで使用されるツールを用いて異なるランサムウェアの解析を行い，共通点となる挙動や特徴の情報を収集し，ランサムウェアへの対策を提案する．

従来のランサムウェアへの対策方法として以下のことが挙げられる．

1. OS およびソフトウェアを常に最新の状態に保つ
2. セキュリティソフトを導入し，定義ファイルは最新状態を保つ
3. メールや SNS のファイルの URL に注意をする
4. 重要なデータはバックアップを取る

これらは何れも一般的なユーザでも行える対策である．特にバックアップを取るという行為は，非常に有効な手段ではあるが，ランサムウェアに関しては欠点もいくつか挙げられる．例えば，バックアップファイルそのものを暗号化されてしまった場合，バックアップの意味をなさない．復号できなかった場合，企業であれば業務に影響が出てしまう．ファイルが暗号化されたまま放置をする訳にはいかないため，従来企業は，ランサムウェアに感染してしまった場合，攻撃者に身代金を支払ってしまった．通常，被害者から身代金を受け取ったことを確認した攻撃者は，ファイルを復号するための鍵データを渡すのがランサムウェアの流れとなっている．しかし，身代金の支払いをし，復号のための鍵データを攻撃者から受け取り，暗号化されたファイルの復号ができたという事例がない．このため，ランサムウェアに感染してしまった場合，攻撃者の思う壺となってしまい, 被害者は金銭的被害を回避することはできない．そして，この身代金を手にした攻撃者が新しい攻撃手法を

開拓するための資金にしてしまう恐れも考えられるため，被害者は金銭的被害を受けるだけではなく，攻撃者に資金援助をしている可能性も有り，攻撃者を助長してしまっていることも考えられる．このことから，ランサムウェアに感染しないために，ランサムウェアならではの挙動や特徴を押さえ，感染被害を減少させる必要がある．

本論文では，メモリフォレンジックを用いて，異なるランサムウェアを起動し，端末を感染させ，メモリイメージの取得を行う．取得したメモリイメージの解析を行い，不審な挙動や特徴の共通点を見つけ，今後のランサムウェア対策へと活かすための方法を提案する．本提案でランサムウェアならではの共通点を見つけることができれば，端末への被害を最小限にすることができると考える．今回の実験で使用するツールは全て無償で入手できるため，金銭的に低コストで済ますことができるというメリットがある．

4.2 提案手法の流れ

今回の実験では，Locky，CERBER，WannaCry の 3 種類のランサムウェア毎に端末に感染させ，ファイルの暗号化処理に使用されたであろうプロセスに着目し，そのプロセスの起動を合図としてメモリイメージの取得を行う．取得したメモリイメージを，メモリフォレンジックツールを用いて解析を行う．Locky，CERBER，WannaCry 毎のプロセス ID の特定を行い，ダンプファイルの取得を行う．また，それぞれのランサムウェアのプロセスに関連している DLL(Dynamic Link Library)のリストの取得を行う．その後，取得したダンプファイルを，Linux コマンドを使用してバイナリファイルに変換し，特徴を模索する．後述する Volatility[6]で上記の作業を行うことが可能である．

4.3 従来の対策方法

ランサムウェアに感染した場合，或いは感染しないための方法として，先程も述べた 4 点が挙げられる．その中でも最も有効な方法として考えられるのが企業，一般人問わずに重要なデータはバックアップを取るということである．しかし，バックアップファイルそのものが暗号化されてしまった場合，復元不可能な状態になってしまう．しかしながら，それ以前の問題も有る．バックアップファイルを取るためにそれ相応の容量が必要となってくることだ．重要なデータは日を重ねる毎に増していくため，その度にバックアップファイルを作成する必要が有る．そのため，十分なデータ容量を格納するための保存先が必要となってしまうためコスト的問題が発生してしまう．

OS およびソフトウェアを常に最新の状態に保つことや，セキュリティソフトを導入し，定義ファイルを常に最新状態に保つことも企業および一般家庭でも行える有効な方法の一つではある．これらも根本的にデータ容量を確保する必要があり，コストが発生してしまうことや，最新状態にすることを怠ってしまったりすることで感染してしまうことも考えられる．

最も初歩的とされる対策方法として，メールや SNS のファイルの URL に注意をすることが挙げられる．ランサムウェアの感染経由は，大抵がメールからとも言われている．攻撃者が内部の人間，もしくは関係者であるかのように宛名を装い，メール文に添付されたファイルを開く旨のメッセージを記述することで被害者はそれに騙され，添付ファイルを開き，感染してしまう．そういった人の心理を利用したソーシャルエンジニアリングによる攻撃が多いのが現実である．そのため，対策を取ったところでも根本的にインターネットを利用する人間の心理を突かれてしまうため，利用者の意識を変えるか，そもそもランサムウェアに感染しないように対策をする必要が有る．

本論文の提案手法では，ランサムウェアの挙動や特徴の共通点を見つけ出し，今後のランサムウェアへの対策方法への糸口を模索する．見つけることができれば，コストの低減や，ソフトウェアを最新状態にすることを怠ったとしても被害を最小限に抑えるといったメリットが有る．

4.4 関連研究との比較

山本ら[2]，竹林ら[4]の手法と比較をすると，WannaCry のみならず，他のランサムウェアを用いて実験を行うため，ランサムウェアならではの挙動や特徴の情報を習得しやすいと考える．また，本論文の実験目的は，暗号化されたファイルの復号ではなく，ランサムウェアの解析を行うため，メモリフォレンジックを用いた場合の有効性について，2 つの文献よりも有効性を明確にすることができるのではないかと考える．

また，重田ら[3]の手法と比較し，本論文では，ランサムウェアの挙動やプロセスに着目するため，感染を防ぐ，または検知をするための特徴を得られるのではないかと考える．

5. 実験

5.1 概要

前章で述べた本論文の提案手法が有効であるかを検証するため，実験を行った．実験では，Locky，CERBER，WannaCry のそれぞれのランサムウェアに感染した端末のメモリイメージの取得を行った上で，取得したメモリイメージを，メモリフォレンジックツールを用いて解析を行い，挙動や特徴の共通点を模索した．

5.2 実験環境

実験で使用した環境を表 1 に示す．仮想環境は Oracle 社の Virtual Box を使用した．スナップショット機能も備わっているため，メモリイメージの取得に失敗したとしても，スナップショットを記録したところから何度もやり直すこ

とが可能である．バージョン 5.2.18 を使用した．

表 1　実験環境

	OS	メモリ
解析対象	Windows7 32bit	4GB
解析環境	Kali-Linux	4GB

解析対象とは，今回の実験で使用したランサムウェア，Locky，CERBER，WannaCry に感染させる端末である．感染した端末からメモリイメージの取得を行う．特に 2017 年に WannaCry によって多大な被害にあったことや，セキュリティパッチが当てられていなかったことから，感染する危険性が高いため，この OS を使用した．メモリ容量は 4GB とした．また，今回の実験ではワーム型のランサムウェアを感染させるため，他の端末が被害を負わないために，IP アドレスを静的に割振り，仮想マシン上のローカルネットワークに設定した．

解析環境とは，解析対象で取得したメモリイメージを解析するための端末である．ペネトレーションテストを主な目的として使用される OS のため，様々なセキュリティツールが予め導入されている．後述するツールもフォレンジックツールの一つとして，予め導入されているため，一から環境を整える手間を省くことができる．

解析対象および解析環境はいずれもメモリイメージを，余裕を持って格納できるストレージとして 50GB とした．また，いずれも仮想マシンの Virtual Box 上のゲスト OS としたため，1 台の PC で実験を行うことができる．

5.3 使用したツール

今回の実験で使用したツールとその目的を表 2 に示す．

表 2　使用したツール

ツール	使用目的
WinPrefetchView[7]	直近に起動したプロセスを監視
Process Explorer[8]	どのタイミングでプロセスが起動しているか監視
FTK Imager Lite[9]	メモリイメージの取得
Volatility Framework (以降 Volatility)[6]	メモリイメージの解析

WinPrefetchView，Process Explorer および FTK Imager Lite は，何れも解析対象にインストールする．Volatility は Volatility Foundation が提供する無償な CUI ベースのツールであり，Python2 系で動作する．Kali-Linux では予め Python2 系がインストールされているため環境設定の手間は省くことができる．

これらのツールは全て無償で使用できるため，金銭的コストを抑えることができ，入手も容易である．

5.4 実験手順

実験手順を以下に示す．

(1) 解析対象にてランサムウェアを起動し，メモリイメージの取得を行う
(2) 解析環境にて(1)の解析を行う

より詳細に明記すると，まず解析対象にてランサムウェアを 1 度起動させ，身代金要求画面に切り替わったところで暗号化処理の終了とし，WinPrefetchView を用いて直近に起動されたプロセスを見る．そして，再度ランサムウェアを起動し，Process Explorer で解析目的のプロセスがいつ起動しているのかを監視し，起動された直後に FTK Imager Lite を使用してメモリイメージの取得を行う．取得したメモリイメージを USB に保存し，解析環境にメモリイメージを移し，Volatility を用いて解析を行う．以上の流れを繰り返し行う．

5.5 使用した検体

解析対象にて使用したランサムウェアの検体のハッシュ値(SHA-256)を表 3 に，検体の詳細を表 4 に示す．

表 3　検体のハッシュ値

検体番号	ハッシュ値(SHA-256)
検体 1	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
検体 2	c35f705df9e475305c0984b05991d444450809c35dd1d96106bb8e7128b9082f
検体 3	b8c1977671dc2911c8d6fd90c74076cb831c3ca2372f791e4ef6c2c4d2916948

表 4　検体情報の詳細

検体番号	ファイルサイズ	タイムスタンプ
検体 1	3432KB	2017/05/14 23:29
検体 2	646KB	2017/10/19 14:17
検体 3	232KB	2016/07/07 00:05

5.6 メモリイメージの取得について

解析を行うため，FTK Imager Lite を用いてメモリイメージの取得を行う．取得するタイミングは，ランサムウェアを起動し，Process Explorer で解析に必要なプロセスが起動したのを確認した直後にメモリイメージの取得を行う．取得を行う上で，山本ら[2]の場合，WannaCry が暗号化の処理の終了の合図として身代金要求画面の表示を基準とし，その直後にメモリイメージの取得を行っていた．メモリ上の情報が時間経過によって消滅することを防ぐためであった．しかし，暗号化の処理で使用されたであろうプロセスは，身代金要求画面が表示される頃には既にメモリイメー

ジから取得できない可能性が高い．Locky，CERBER でも同様に身代金要求画面が表示されたのを暗号化処理の終了の合図として，メモリイメージの取得を行った上で，Volatility で解析したところ，プロセスリストには殆どのプロセスの痕跡は残ってはいなった．このことから WannaCry のときに残されていたプロセスの痕跡の殆どは，身代金要求画面を表示するために使用されたであろうプロセスであり，そのため，身代金要求画面表示の段階では暗号化されたファイルの復号のための鍵データは残されていなかったのだと考えられる．

5.7 解析対象となるプロセス

今回の実験では，メモリ上に展開されていた全てのプロセスに対して解析を行うのではなく，ランサムウェアに関連している可能性の高いプロセスを特定し，そのプロセスのみ解析を行う．関連しているプロセスを特定するために WinPrefetchView というツールを使用した．このツールは，実行したプログラムを Prefetch ファイルとして作成し，プログラムがいつ実行されたのか，プログラムがどの DLL ファイルをロードしたのか等が，記録として残り，ユーザに分かりやすく表示することができる．そのため，1 度ランサムウェアを起動し，起動後に生成された Prefetch ファイルを確認する．ランサムウェア起動後に生成された Prefetch ファイルであれば，ランサムウェアと関係のあるプロセスであると考えられるためである．確認後，再度解析対象にてランサムウェアを起動し，特定のプロセスが起動するタイミングを Process Explorer にて監視する．起動された直後に FTK Imager Lite を使用して，メモリイメージを取得し，解析を行う．ランサムウェア起動後に生成されたプロセスの Prefetch ファイルおよび FTK Imager Lite を使用し，不審なプロセスの取得の有無を表 5 にて示す．プロセス名を PN，プロセス ID を PID と表記する．また，表 5 で示したプロセスを今回の解析対象のプロセスとする．

表 5　ランサムウェア起動後に生成された不審なプロセス

検体番号	PN	PID	取得の有無
検体 1	taskhsvc	2280	有
	@WanaDecryptor@	2644	有
		2764	有
		3784	有
		3988	無
	icacls	-	有
	ed01ebfbc9eb5b	800	有
	cscript	-	無
	taskdl	468	無
		1736	無
	attrib	-	無
検体 2	dllhost	912	有
	dda37961870ce0	1792	有
検体 3	dllhost	1948	無
		2012	無
	vssadmin	1948	無
	WMIC	1144	無
	ntkrnlpa	-	有
	PING	2536	無
	taskkill	-	無
	b8c1977671dc29	3788	有

FTK Imager Lite で取得できなかったプロセスは，今回の実験においては Process Explorer から直接メモリダンプの取得を行った．Process Explorer は，表示されているプロセスからそのプロセスのメモリダンプを直接取得することが可能である．しかし，一部のプロセスはアクセス拒否によってメモリダンプの取得が不可能であったりするため，解析をするためのプロセス全てを取得することはできなかった．また，Process Explorer からメモリダンプを直接取得するため，メモリイメージとは異なり，Volatility で解析をすることはできない．しかしながら，後述する Volatility の memdump コマンドを使用して取得できるメモリダンプと Process Explorer で取得したメモリダンプは同じメモリダンプだと考えられるため代用することができる．また，Volatility の dlllist コマンドで取得できる特定のプロセスがロードした DLL リストも，WinPrefetchView から見ることが可能なため，代用することができる．

5.8 メモリイメージの解析

解析対象にて取得したメモリイメージは，解析環境で解析を行う．その際に Volatility というツールを使用し，解析を行う．使用したコマンドを表 6 に示す．

表 6　使用したコマンド

コマンド	機能
psscan	プロセス情報のリストを表示
dlllist	プロセスがロードした DLL 情報のリストを表示
memdump	プロセスのメモリダンプの取得

psscan コマンドを使用することで，メモリイメージを取得したタイミング時に，メモリ上で展開されていたプロセスのリストを表示する．表 5 で示したプロセスが取得できているのかを確認する意味もある．同時にプロセス ID もこのときに取得する．取得結果の一部を図 1 に示す．

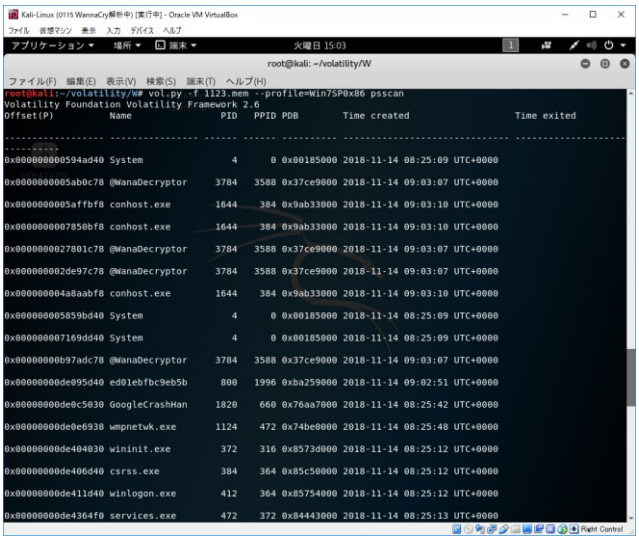

図 1　psscan コマンドの出力結果

また，pstree コマンドを使用すれば，psscan と同じくプロセスのリストを表示するが，プロセスの親子関係が見やすくなる．

dlllist コマンドで解析対象のプロセスがロードしていた DLL 情報のリストを表示する．dlllist コマンドの出力結果を図 2 に示す．

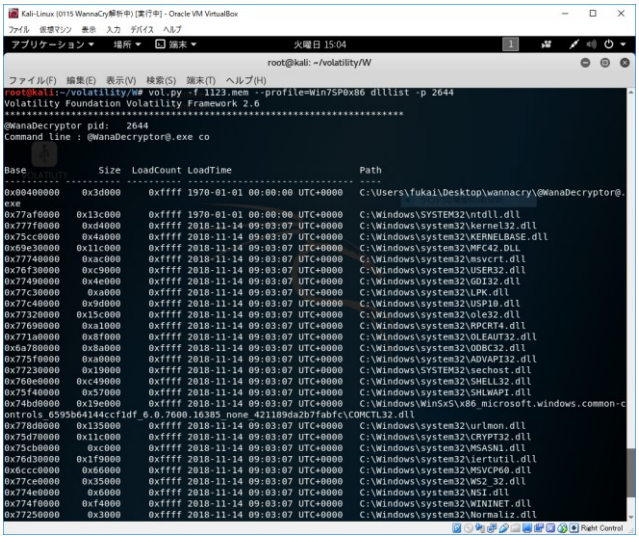

図 2　dlllist コマンドの出力結果

memdump コマンドは，プロセス内の全ての物理メモリ上のページを個別のダンプファイルとして展開することができる．memdump コマンドで取得した個別のメモリダンプファイルを，Linux コマンドを使用して ASCII 形式の文字列に変換した．使用した Linux コマンドは，表 7 に示す．

表 7　Linux コマンド

コマンド	機能
strings	バイナリファイルから ASCII 形式の文字列として読める箇所を表示
grep	特定の文字を含む行を抽出する

strings コマンドを使用することで，バイナリファイルの内容を把握することができる．また，strings コマンドでは，grep コマンドも併用できる．grep コマンドは特定の ASCII 形式の文字列を抽出することができるため，今回の実験では，RSA，AES，Crypt で grep した．主に暗号化で使われるものと，暗号化する際に使われる関数は，CryptoAPI だと予測していたためである．

5.9 実験結果

実験の結果，共通点となるものを発見することができた．得られた挙動や特徴の共通点について以下に示す．

まず，ランサムウェアを実行後，Process Explorer で CPU の動きを監視していると，WannaCry を除いた Locky，CERBER の 2 つの検体は，CPU の使用率が 1 度だけではなく，複数回に渡って 100%近く占有した．WannaCry のみ 1 度だけであった．

次に，WinPrefetchView を用いて，ランサムウェア実行後に記録されたファイルを見たところ，記録されたプログラムは，ランサムウェア毎に異なっていた．しかし，どのランサムウェアもハッシュ値名のプロセスは必ず記録されていた．

最後に，WinPrefetchView で不審だと思われるプロセスを推定し，メモリダンプをバイナリファイルに変換し，解析したところ，全てのプロセスにおいて CryptoAPI の痕跡があった(図 3)．また，CERBER の ntkrnlpa のプロセスを除いた全てのプロセスで RSA1 または，RSA2 もしくは両方が残っていた．更に，DLL を調査したところ，どのプロセスも Windows 上の DLL を使用していたが，WannaCry のみ独自の DLL らしきものを使用していることが分かった．

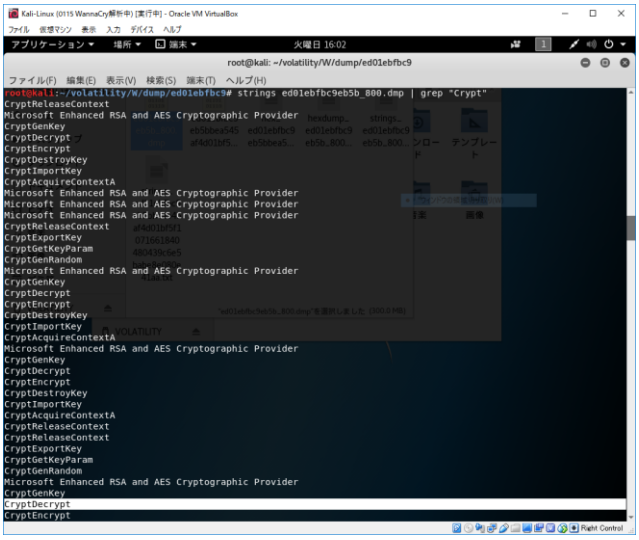

図 3　CryptoAPI の痕跡

6. 考察

本論文では，ランサムウェアの挙動や特徴の共通点となるものを模索するために解析を行った．まず，CPU 使用率が 100%近くを占めた要因として，暗号化の処理が行われていたと考えられる．WannaCry は，起動直後に暗号化の動きがあり，数秒で身代金要求画面に切り替わった．このことから，CPU 使用率が 100%近くを占めたのが 1 度だけだったと考えられ，身代金要求画面に切り替わるまで比較的時間のあった Locky，CERBER は複数回に渡り，CPU 使用率が 100%近く占めたと予測される．また，CPU 使用率が高い間に鍵データの生成を行い，CryptDestroyKey 関数によって鍵データの破棄が行われていたと考える．そのため，暗号化されたファイルの復号をするためには，CryptDestroyKey 関数によって鍵データが削除される前に鍵データを取得する必要がある．しかし，これらのことからメモリ上から鍵データの取得となると現実的ではない．しかし，CryptoAPI が呼び出されるタイミングを検知さえすれば，鍵データを入手できる可能性が有る．鍵データの入手方法として，API フック等が考えられる．

7. おわりに

本論文では，メモリフォレンジックを用いてランサムウェアの挙動や特徴の共通点の模索を行った．今回は，FTK Imager Lite を使用して，特定のプロセスがメモリ上に展開されたであろうタイミングにメモリイメージの取得を行った．プロセスの実行されるタイミングは疎らなため，1 度に解析に必要なプロセスの取得ができないことが欠点である．

しかしながら，メモリフォレンジックを用いたことで，ランサムウェアの挙動や特徴の共通点について大まかに把握することができた．これらを踏まえると CPU の使用率が高いかつ，CryptoAPI が呼び出された場合に，警告を出すといったツールを開発することが有効なのではないかと考えられる．

また，プロセス自体は取得できていたが，Volatility で解析しようとしたところ，プロセス ID が見つからないエラーにより，解析できなかったプロセスも有り，原因究明が必要である．

今後は，Volatility で起きたエラーの原因を究明すること，また，今回の実験環境よりもより現実に近い，模擬的な通信を行えるようなネットワーク環境と疑似的な C&C を用意し，より詳細な挙動や特徴の共通点を見出すことができるかを検証していく．

参考文献

[1] “トレンドマイクロ”, ”2017 年年間セキュリティラウンドアップ”, https://www.trendmicro.com/ja_jp/about/press-release/2018/pr-20180227-01.html.

[2] 山本裕也，広瀬幸，柿崎淑郎，猪俣敦夫，“メモリフォレンジックを用いたランサムウェアによる暗号化ファイルの復号可能性の調査”， 2018 年 暗号と情報セキュリティシンポジウム(SCIS2018).

[3] 重田貴成，伊沢亮一，森井昌克，井上大介，中尾康二，“ランサムウェア検知のための特徴解析”，コンピュータセキュリティシンポジウム 2017 論文集, 2017, No.2, 2017.

[4] 竹林康太，上原哲太郎，佐々木良一，“ライブフォレンジックを用いた暗号化ファイル復号技術の開発”，研究報告マルチメディア通信と分散処理， 2015-DPS-162, No.38, pp.1-6, 2015.

[5] “aeskeyfind”, https://github.com/eugenekolo/sec-tools/tree/master/crypto/aeskeyfind.

[6] “The VolatilityFoundation”, “Volatility Framework”, https://www.volatilityfoundation.org/.

[7] “NirSoft”, “WinPrefetchView”, https://www.nirsoft.net/utils/win_prefetch_view.html.

[8] “Microsoft”, “Process Explorer”, https://docs.microsoft.com/ja-jp/sysinternals/downloads/process-explorer.

[9] “ACCESS DATA”, “FTK Imager Lite”, https://accessdata.com/.

†東京電機大学 〒120-8551 東京都足立区千住旭町 5. Tokyo Denki University, 5, Senju-Asahi-cho, Adachi-ku , Tokyo 120- 8551, Japan.