ランサムウェアの親プロセスと子プロセスの API コール分析

Analysis of the API calls of the parent/child processes of ransomwares

松田 祥希 † 高橋 健一 † †† 東野 正幸 † †† 川村 尚生 † ††

Yoshiki Matsuda † Kenichi Takahashi † †† Masayuki Higashino † †† Takao Kawamura † ††

† 鳥取大学大学院持続性社会創生科学研究科　†† 鳥取大学クロス情報科学研究センター

1　はじめに

近年，様々なサイバー攻撃が発生している．特に企業の DX 化などによってインターネット，コンピュータの利用範囲が増え，サイバー攻撃の被害が大きくなっている．2022 年の 2 月にはトヨタ自動車の主要な取引先である「小島プレス工業」がランサムウェアに感染し，トヨタ自動車の国内全ての 14 工場の 28 ラインの稼働が停止する事態となった[1]．2023 年の 7 月には名古屋港運協会がランサムウェアによる攻撃を受け，NUTS（名古屋港統一ターミナルシステム）に障害が発生し，全ターミナルの作業が停止した[2]．また，2022 年 2 月から Emotet の感染が再拡大している．特に Emotet 自身では，アンチウイルス製品による検知回避の強化を行っているとともに，OneNote を利用した感染方法にシフトされていることが確認されている．

このようなサイバー攻撃による被害の軽減のため，マルウェアファミリの推定やマルウェアの検知に関する研究 [1][2][3] が行われている．しかし，これらの研究では親プロセスと子プロセスを一体として扱っており，親プロセスと子プロセスの違いや関係について着目されていない．一方で，処理を子プロセスに分割することで，マルウェアの分析・検知が難しくなることが報告されている [4]．そこで，本研究ではランサムウェアを対象に，親プロセスと子プロセスの分析を行う．

2　関連研究

佐藤ら [1][2] は API の呼び出しとそれに伴うシステム負荷に着目し，マルウェアが自身の存在を隠蔽する際の動作等を抽出し，API 呼び出しとシステム負荷から作成した特徴量を利用した分類器でマルウェアの検知を試みている．飛山ら [3] は異なる種類の Deep Neural Network を多段的に用いることで，プロセスから得られた API コール列の特徴を効率的に抽出し，Windows 端末上で実行されている悪性プロセスの推定を行っている．しかし，これらの研究では親プロセスと子プロセスの分析はされていない．

Ramilli ら [5] はマルウェアの挙動を複数のプロセスに分割することによってマルウェアの検知を逃れることができることを BullMoose というマルウェアを用いて確認した．Ji ら [4] は Zeus bot[6] による攻撃活動や C&C サーバとの通信などの挙動を複数のプロセスに分割することでマルウェアの検知が難しくなることを確認した．Ma ら [7] は複数の shadow プロセスによって実行されるマルウェアを生成するツールを作成し，マルウェアの検知を逃れることができることを確認した．これらの研究では，単一のプロセスの分析のみではマルウェアの検知を行うことは難しく，マルウェアの親プロセスとその子プロセスによってなされる悪性挙動に着目する必要があることを示している．

Gaspari ら [8] はランサムウェアを分析し，ランサムウェアが複数のプロセスに分割する仕組みを示すとともに，分割されたプロセスを学習することによって，検知できることを確認した．中村ら [9] は API コール列の類似性を分析するために，API コール列に自然言語処理の手法を適用し，マルウェアと正規ソフトウェアで親プロセスと子プロセスの API コール列の類似度測定を行った．その結果，マルウェアと正規ソフトウェアの間では親プロセスと子プロセスの類似性に違いがあることが分かった．更に，得られた類似度を用いて SVM を利用したマルウェアの判別を行い，親プロセスと子プロセスの類似度にマルウェアとクリーンウェアで違いがあることを確かめた．

図 1: MegaCortex のプロセスツリー

3　ランサムウェアとクリーンウェア

3.1　ランサムウェア

ランサムウェアとはコンピュータに保存されているファイルの暗号化を行い，コンピュータを使用不可能にするマルウェアグループの総称である．

ランサムウェア MegaCortex[3]では，正規プログラムを子プロセスとして利用する．また，10 個のファイルのみを暗号化する子プロセスを作成するといった，マルウェア検知の回避を意図したと考えられる挙動が見られる．図 1 に MegaCortex のプロセスツリーの一例を示す．　図 1 で赤で塗りつぶされているプロセスは MegaCortex の本体で，青で塗りつぶされているプロ

[1] https://www.pentasecurity.co.jp/pentapro/entry/toyota-ransomware-attack

[2] https://meikoukyo.com/wp-content/uploads/2023/07/0bb9d9907568e832da8f400e529efc99.pdf

[3] https://www.mbsd.jp/blog/20191113.html

表 1: 子プロセスの作成数

子プロセス作成数	0 個	1 個	2 個	3 個	4 個	5 個	6 個	7 個	8 個	9 個	10 個以上
ランサムウェア検体数	110	345	871	80	37	127	53	24	3	20	13
クリーンウェア検体数	1566	118	18	24	7	6	0	1	1	2	6

セスは正規プログラムを利用して作成された子プロセスである．このように MegaCortex ではファイルの暗号化を少しずつ行う，サービスの停止やプロセスの終了などに正規プログラムを利用するなど，マルウェア検知を逃れることを意図して設計されていることがわかる．

3.2 クリーンウェア

マルウェア以外のソフトウェアをクリーンウェアと呼ぶ．クリーンウェアの例としてはウェブブラウザ，ワープロソフト，画像編集ソフト等があり，特定のタスクを効率的に処理するために設計されている．クリーンウェアでは実行中に発生したエラー等の問題による影響範囲の限定，マルチスレッドによるコンピュータリソースの効率化のために子プロセスを作成する．

3.3 プロセスの親子関係情報

どのプロセスがどのプロセスによって作成されたかという情報をプロセスの親子関係情報と呼ぶ．プロセスを作成したプロセスを親プロセス，作成されたプロセスを子プロセスと呼ぶ．一つの親プロセスが複数の子プロセスを作成することもある．

本研究では最初に実行されたプロセスを親プロセス，作成されたそれ以外のプロセスを子プロセスとする．図 1 の例では，最上段の winnit.exe が親プロセスであり，それ以下の net.exe, taskkill.exe, winnit.exe, vssadmin.exe, cipher.exe はすべて子プロセスとなる．

3.4 研究目的

クリーンウェアでは並列実行やエラーによる問題の影響範囲の限定のために子プロセスを作成すると考えられる．一方，マルウェアでは並列実行やマルウェア検知の回避のために子プロセスを作成する．特に，マルウェア検知を回避するために子プロセスを作成する検体では，一連の処理を複数の子プロセスに分割すると考えられる．このため，親プロセスと子プロセスを別々に分析し，その動作を明らかにする必要がある．

そこで，本研究ではランサムウェアを対象とし，親プロセスによる子プロセスの作成数や API コール等の分析を行う．

3.5 データセット

ランサムウェアとクリーンウェアの分析のために動的解析のデータセットを用いる．本研究では Cuckoo Sandbox を利用した Windows 7 環境で作成された FFRI Dataset 2017[4] と Mal_Ben_Dataset[5] を用いる．FFRI Dataset 2017 からは Kaspersky がランサムウェアと識別した 1683 検体のランサムウェアのデータが，Mal_Ben_Dataset からはフォーマット不備のデータを除いた 1749 検体のクリーンウェアのデータが得られた．

[4] https://www.iwsec.org/mws/2017/20170606/FFRI_Dataset_2017.pdf

[5] https://www.reddit.com/r/datasets/comments/exhy38/malware_and_benign_windows_pe_cuckoo_reports/

4 ランサムウェアとクリーンウェアの比較

親プロセスによる子プロセスの作成数や API の呼び出し種類数の分析を行い，クリーンウェアとの比較を通じてランサムウェアの特徴を分析する．

4.1 子プロセスの作成数

表 1 にランサムウェアとクリーンウェアが作成した子プロセス数と検体数をまとめる．

表 1 から，ランサムウェアは 1683 検体中 1573 検体に当たる 93.5%が子プロセスを作成していることが確認できる．一方，クリーンウェアでは 1749 検体中 183 検体に当たる 10.5%が子プロセスを作成していることが確認できる．ここから，ランサムウェアとクリーンウェアでは子プロセス作成数に違いがあることがわかる．特に，ランサムウェアでは 90%以上の検体が子プロセスを作成しており，ほとんどのランサムウェアは子プロセスを利用した処理を行っている．また，子プロセスを 2 つ作成する検体が 50%存在しており，子プロセスを 5 つ作成する検体も 7.5%存在している．一方，クリーンウェアでは子プロセスを作成しない検体が 90%近くを占めており，子プロセスを作成した検体も 64%が子プロセスを 1 つしか作成していない．

4.2 API の呼び出し種類数の比較

ランサムウェアは負荷分散や悪性挙動を子プロセスに分割し検知を逃れる等，様々な目的のために子プロセスを作成すると考えられる．子プロセスの作成目的によって親プロセスと子プロセスには違いが生じる．そこで，親プロセスと子プロセスで利用されている API の呼び出し種類数の分析を実施した．図 2 にそれぞれの API を呼び出したランサムウェアとクリーンウェアの検体の割合を示す．横軸は API コール名，縦軸は API を呼び出した検体の割合を示す．

図 2 から，ランサムウェアとクリーンウェアで利用する API の傾向に違いがあることが確認できる．例えば，RegQueryValueExA はランサムウェアの親プロセスでは 80%が利用していたが，クリーンウェアでは 23%しか利用していなかった．また，NtDuplicateObject では，ランサムウェアの親プロセスでは 83.7%が利用していたが，クリーンウェアでは 38.5%しか利用していなかった．また，ランサムウェアのみが利用している API（ControlService, recv, sendto, send, RtlDecompressBuffer 等）も存在していた．このようにラン

表 2: 90%以上のプロセスで利用されていた API 名

ランサムウェア	親プロセス	NtCreateFile, NtAllocateVirtualMemory, NtOpenKey, NtQueryValueKey, GetSystemTimeAsFileTime, LdrLoadDll, LdrGetDllHandle, LdrGetProcedureAddress, NtCloce
	子プロセス	NtCreateFile, NtAllocateVirtualMemory, NtOpenKey, NtQueryValueKey, GetSystemTimeAsFileTime, LdrLoadDll, LdrGetDllHandle, LdrGetProcedureAddress, NtCloce
クリーンウェア	親プロセス	NtOpenKey, NtQueryValueKey, LdrGetDllHandle, NtCloce
	子プロセス	NtCreateFile, NtQueryValueKey, GetSystemTimeAsFileTime, LdrGetDllHandle, LdrGetProcedureAddress, NtCloce

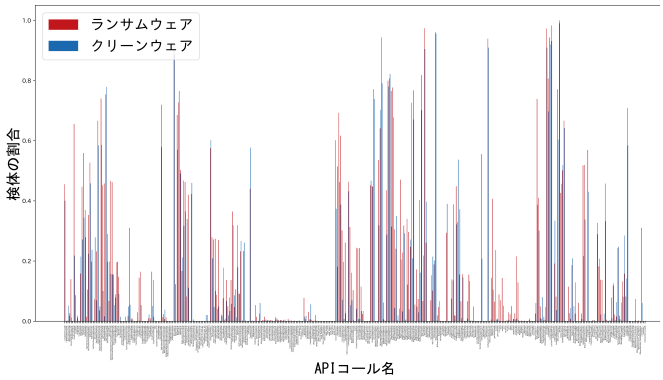


図 2: ランサムウェアとクリーンウェアの API 利用率の比較

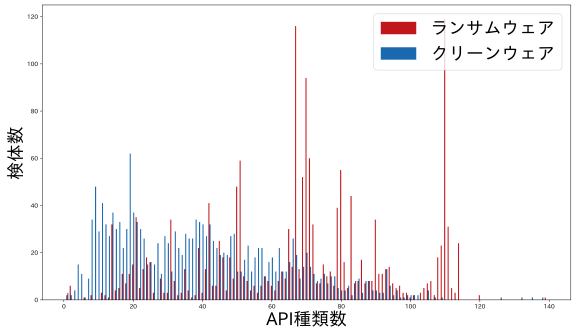


図 3: 親プロセスが利用した API 種類数の比較

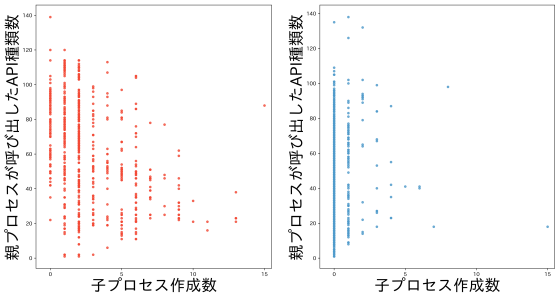
サムウェアとクリーンウェアで利用する API の傾向に違いがあり，この特徴がランサムウェアの特徴として利用できると考えられる．そこで，ランサムウェアとクリーンウェアの親プロセスと子プロセスの 90%以上が利用していた API 名を抽出した（表 2）．

ランサムウェアでは親プロセスと子プロセスのそれぞれで 9 種類の API が抽出され，その API はすべて同じものとなった．クリーンウェアでは親プロセスから 4 種類，子プロセスから 6 種類の API が抽出された．クリーンウェアの親プロセス，子プロセスから抽出された API は全てランサムウェアでも抽出されていた．ランサムウェアでよく利用されていた API はクリーンウェアでもよく利用されているが，その割合は減っていることがわかる．

4.3 API の種類数の分布

ランサムウェアとクリーンウェアの親プロセスが利用した API の種類数を図 3 に示す．横軸は API 種類数，縦軸は検体数を表している．

図 3 のグラフの形を比較すると，ランサムウェアとクリーンウェアで異なる形の山となっており，ランサムウェアの方が利用している API の種類が多いことがわかる．また，クリーンウェアでは広くて低い山が，ランサムウェアでは狭く高い山が複数存在することが確認できる．このランサムウェアの山はファミリや亜種によるものではないかと考えられる．ランサムウェアでは同じファミリに属する検体が同じような挙動を



ランサムウェア　　クリーンウェア

図 4: 子プロセスの作成数と API 種類数の比較

行うことが多く，これにより特定の値の山を作っていると考えられる．

4.4 親プロセスで利用される API の呼び出し種類数の比較

子プロセスを作成する目的によって親プロセスの挙動が変化すると考えられる．挙動の変化が API コールの変化につながる．そこで，ランサムウェアとクリーンウェアの子プロセス作成数と親プロセスによる API の呼び出し種類数の比較を行った．子プロセスの作成数と親プロセスによる API の呼び出し種類数の関係を図 4 に示す．横軸は子プロセスの作成数，縦軸は親プロセスが呼び出した API の種類数を示している．

子プロセスを作成しないランサムウェアは，98.2%

の親プロセスが 40 種類以上の API を利用していた．一方で，子プロセスを作成するランサムウェアでは，81.7%に減少しており、18.7%の親プロセスが 40 種類以下の API しか利用していなかった．子プロセスを作成しない検体では 40 種類以上の API を利用している検体の割合が多いが，子プロセスを作成する検体では 40 種類以下の API しか利用しない親プロセスの割合が増加する．これは，子プロセスを作成する検体では，子プロセスに悪性挙動を行わせているからではないかと考えられる．また，子プロセスを作成する検体でも，親プロセスが利用する API の種類数にばらつきが見られる．このばらつきは，親プロセスが悪性挙動を行うかどうかによって生まれていると考えられる．ここから，ランサムウェアを子プロセスを一切利用しない検体，子プロセスのみに悪性挙動を行わせる検体，悪性挙動を親プロセスと子プロセスの双方で行う検体の 3 種類に分類することができると考えられる．クリーンウェアでは子プロセスを作成しない上に利用する API の種類も少ない検体が存在していた．これは，クリーンウェアでは GUI 等を利用してユーザからの操作を待つため，今回のデータ取得環境では初期化等の動作しか実行されなかったためだと考えられる．また，Cuckoo Sandbox はマルウェアがよく利用する 360 種類程度の API を取得するため，それ以外の API が分析対象となっていないためだと考えられる．

5　おわりに

ランサムウェアとクリーンウェアで子プロセスの作成数や利用している API の呼び出し種類数に違いがあることがわかる．また，ランサムウェアには親プロセスのみで処理を行う検体と，子プロセスを利用して処理を行う検体が存在した．更に，子プロセスを利用して処理を行っている検体に関して，API をあまり利用しない親プロセスと，多くの API を利用する親プロセスがあることが確認できた．子プロセスの作成数と API コールの関係からランサムウェアを 3 種類に分類する特徴を見つけた．この分類ごとに適した手法で親プロセスと子プロセスの挙動を組み合わせることで，ランサムウェアの動作をより鮮明にすることができると考えられる．

謝辞

本研究は，JSPS 科研費 JP23K11101 の助成を受けて行った．

参考文献

[1] 佐藤順子，三須剛史，花田真樹，鈴木英男，布広永示：API 呼び出しとシステム負荷を用いたマルウェアの特徴抽出に関する一検討，コンピュータセキュリティシンポジウム 2016，pp. 305-309（2016）．

[2] 佐藤順子，花田真樹，面和成，村上洋一，鈴木英男，布広永示関口竜也：API 呼び出しとそれに伴う経過時間とシステム負荷を用いた重み付けスコアに基づくマルウェア検知手法，コンピュータセキュリティシンポジウム 2018，pp. 1266-1270（2018）．

[3] 飛山駿，山口由紀子，嶋田創，秋山満昭，八木毅：Deep Neural Network 多段化によるプロセスの挙動に着目したマルウェア推定手法，コンピュータセキュリティシンポジウム 2016，pp. 310-317（2016）．

[4] Ji, Y., He, Y., Zhu, D., Li, Q., Guo, D.：A Mulitiprocess Mechanism of Evading Behavior-Based Bot Detection Approaches，The 10th International Conference on Information Security Practice and Experience, pp. 75-89（2014）．

[5] Ramilli, M., Bishop, M., Sun, S.：Multiprocess Malware，The 6th International Conference on Malicious and Unwanted Software，pp. 8-13（2011）．

[6] Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., Youssef, A., Debbabi, M., Wang, L.：On the Analysis of the Zeus Botnet Crimeware Toolkit，The 8th International Conference on Privacy, Security and Trust，pp. 31-38（2010）．

[7] Ma, W., Duan, P., Liu, S., Gu, G., Liu, J. C.：Shadow attacks: automatically evading system-call-behavior based malware detection，Journal in Computer Virology, Vol. 8，pp. 1-13（2012）．

[8] De Gaspari, F., Hitaj, D., Pagnotta, G., De Carli, L., Mancini, L.V.：Evading behavioral classifiers: a comprehensive analysis on evading ransomware detection techniques，Neural Computing and Applications, Vol. 34，pp. 12077-12096（2022）．

[9] 中村英敏，松田祥希，高橋健一，川村尚生：マルウェア検知に向けた親プロセスと子プロセスにおける API コールの類似性に着目した分析，情報処理学会論文誌, Vol.64, No.9，pp. 1306-1316 (2023).