LLMによるサンドボックスの回避耐性強化支援

伊藤 曜[1,a)]　グエンティヴァンアン[2]　インミンパパ[2]　田辺 瑠偉[2,3]　吉岡 克成[2,4]

概要：マルウェアは増加の一途を辿っており，短時間で大量の検体を自動実行できるサンドボックス解析技術が必要不可欠となっている．特に，感染前の状態に迅速に復元できる仮想化技術を用いてサンドボックスが構築される場合が多い．しかし一方で，サンドボックスを検知して解析を回避する耐解析マルウェアが増加しており，これに対応するためのサンドボックスの高度なカスタマイズが求められている．本研究では，大規模言語モデル（LLM）を活用して耐解析マルウェアに対して耐性を持つサンドボックスを実現する方法を提案する．提案手法において，LLM はサンドボックス回避技術を実装した概念実証コードを入力として，回避の仕組みを分析し，回避に対抗するためにサンドボックスに必要なカスタマイズ方法を導出し，その実現のためのスクリプトや手順書を生成する．評価実験では，オープンソースの CAPE サンドボックスを用いて構築したマルウェア解析システムに対して，システムを構成するゲスト OS，VM ハイパーバイザーおよび CAPE サンドボックス本体のそれぞれに対して可能なカスタマイズ方法を提示するシステムを実装した．そして，あらかじめ用意した擬似的な耐解析マルウェア検体と解析回避 PoC コードの合計 43 検体を用いて，提案手法が提示したカスタマイズを適用したサンドボックスの回避耐性を評価したところ 31 件（約 72%）で解析に成功し，VirusTotal 上で動作する CAPE サンドボックスと比較して大幅な回避耐性の強化が確認された．また，手順書やスクリプトを自動生成することで，効率的かつ迅速なサンドボックス強化のプロセスを実現できた．

Enhancing Sandbox Evasion Resistance Using LLM

Hikaru Ito[1,a)]　Nguyen Thi Van Anh[2]　Yin Minn Pa Pa[2]　Rui Tanabe[2,3]
Katsunari Yoshioka[2,4]

Abstract: Malware continues to increase, making sandbox analysis technologies that can automatically execute large volumes of samples in a short time indispensable. In particular, sandboxes are often built using virtualization technologies that allow for rapid restoration to a pre-infection state. However, the number of evasive malware capable of detecting sandboxes and evading analysis is also increasing, and there is a growing need for advanced customization of sandboxes to counter this. In the proposed approach, the LLM accepts proof-of-concept code implementing sandbox evasion techniques as input, analyzes the underlying evasion mechanisms, and produces the required customization methods along with the scripts and manuals necessary to customize the sandbox. For evaluation, we implemented a system that suggests possible customization methods for each component of a malware analysis system built on the open-source CAPE sandbox, namely the guest OS, VM hypervisor, and the CAPE sandbox itself. Using a total of 43 prepared samples—comprising pseudo anti-analysis malware and evasion PoC code—we evaluated the evasion resistance of the sandbox after applying the proposed customizations. As a result, analysis was successfully performed on 31 cases (approximately 72%), demonstrating a significant improvement in evasion resistance compared to the CAPE sandbox operating on VirusTotal. Furthermore, by automatically generating manuals and scripts, the proposed method enabled an efficient and rapid process for sandbox reinforcement.

[1] 横浜国立大学大学院環境情報学府
Graduate School of Environment and Information Sciences, Yokohama National University
[2] 横浜国立大学先端科学高等研究院
Institute of Advanced Sciences, Yokohama National University
[3] 順天堂大学
Juntendo University
[4] 横浜国立大学大学院環境情報研究院
Faculty of Environment and Information Sciences, Yoko-

1. はじめに

増加の一途を辿るマルウェアに対抗するために，サンドボックスと呼ばれる解析環境内でマルウェア検体を自動実行してその挙動を把握するサンドボックス解析技術は重要な役割を果たす．特に，短時間で大量の検体解析が求められる状況において，サンドボックスを感染前の状態に迅速に復元できる仮想化技術が広く利用されている [1, 2]．しかし一方で，サンドボックス固有の特徴を検知することで，マルウェア本来の挙動を隠して解析を回避する耐解析マルウェアが問題となっている．先行研究 [3] では，2010 年から 2019 年に収集したマルウェア 45,375 検体のうち，約 80%が解析の回避を試みたと報告している．

耐解析マルウェアによる解析回避を防ぐためには，サンドボックスをユーザマシンに似せるなどして回避耐性を強化する必要がある．しかし，サンドボックスを回避する技術は多岐に渡るため，サンドボックスのカスタマイズは実運用上の課題となっている [1]．近年では，大規模言語モデル（LLM：Large Language Model）が耐解析マルウェアの多様化やサンドボックス回避技術の開発に悪用される可能性が指摘されている [4–6]．AI による技術革新は，マルウェアを解析する防御者側のワークフローの効率化やサンドボックスのカスタマイズにも活用できるため，我々は先行研究においてその可能性を検証した [7]．

本研究では，LLM を活用してサンドボックスの回避耐性を強化する方法を生成し，提案手法を用いて強化されたサンドボックスの回避耐性を評価する．具体的には，LLM に (a) 耐解析機能を有する検体のソースコードと，(b) サンドボックスに関する知識を与えることで，サンドボックスの回避耐性強化に必要なスクリプトや手順書を生成し，実際にサンドボックスのカスタマイズを行った．

評価実験では，オープンソースのマルウェア解析システムである CAPE サンドボックス [8] に対して回避耐性の強化を行った．既存研究 [9] とサンドボックス検知ツール [10] を用いて用意した解析回避検体 122 件のうち，初期状態の CAPE サンドボックスの解析回避に成功した合計 43 検体を OpenAI 社が提供するモデル gpt-4.1 に入力した．さらに，(b)LLM に与えるサンドボックスに関する知識が回避耐性強化の成功率にどのように影響するか調査した．実験 1（簡易情報の提示）では，LLM にサンドボックスを構成するゲスト OS やハイパーバイザなど、基本的な情報のみを提供した．この結果，LLM は回避耐性強化のための手順書を 40 件，自動化スクリプトを 3 件生成し，このうち 20 個が実際のサンドボックス強化に繋がった．

実験 2（具体的な実装例の提示）では，LLM に実験 1 で LLM に与えた簡易情報に加えて，仮想マシンのカスタマイズに関する専門的な指針（例：PowerShell スクリプトの使用）を追加で提供した．この結果，LLM は回避耐性強化のための手順書を 12 件，自動化スクリプトを 24 件生成した．初期状態の CAPE を提案手法でカスタマイズしたところ，このうち 26 個が実際に強化に繋がった．これは，より具体的な情報を提供することで，LLM が手順書よりも直接的な実装コードを優先して生成する傾向があることを示している．

実験 3（追加の具体的な実装例の提示）では，LLM に実験 1 の簡易情報と実験 2 の具体的な実装例に加えて，CAPE サンドボックスの API フックに関する具体的な実装例を提供した．この結果，LLM は回避耐性強化のための手順書を 12 件，自動化スクリプトを 31 件生成した．初期状態の CAPE を提案手法でカスタマイズしたところ，このうち 31 個が実際に強化に繋がった．このことから，LLM に具体的な技術的詳細を与えることで，実効性の高いコードや手順をより多く生成できることがわかった．

最後に，これまでの実験で用いた検体群を VirusTotal 上で実運用されている CAPE サンドボックスで解析し，提案手法でカスタマイズしたサンドボックスとの比較を行った．実験には，最も回避耐性を有していた実験 3 でカスタマイズしたサンドボックスを用いた．この結果，VirusTotal 上で実運用されている CAPE サンドボックスで解析に成功したのは 40%未満であり，提案手法が大きく上回る結果となった．このように，提案手法を用いて効率的かつ迅速なサンドボックス強化のプロセスを実現できた．

2. 提案手法

本章では，LLM を活用してサンドボックスの回避耐性強化方法を生成する手順と，生成された強化方法に基づいたサンドボックスのカスタマイズの流れを説明する．

2.1 LLM を活用した回避耐性強化支援

サンドボックスは，プログラムやファイルを実行・検証するための実行環境であり，仮想化技術を用いて外部から隔離された仮想空間上に作られる場合が多い．また，一般的にサンドボックスは，マルウェア解析を行うホスト側とマルウェアを実行するクライアント側で構成される．以下に，それぞれのカスタマイズ方法をまとめる．

(I) **ホスト側（マルウェア解析ツール）：**特定の DLL 関数に API フックを挿入することで，API 呼び出しやシステムコールを補足・修正する新規ロジックの導入を行う．これにより，通常では不可能な値の取得や取得された値の書き換えを可能にする．

(II) **クライアント側（仮想化環境）：**ゲスト OS 内部，および，ゲスト OS を管理するハイパーバイザを操作することで，設定の変更を行う．例えば，論理プロセッサの数を調整する場合，ハイパーバイザの GUI 設定画

hama National University
a) ito-hikaru-gd@ynu.jp

面から変更を行う方法がある．

続いて，LLM を活用してサンドボックス解析システムのホスト側とクライアント側の両方を強化する方法を生成する．図 1 に提案手法の全体像を示す．初めに，LLM に回避技術を実装した検体のソースコードを入力し，コードを解析させることで検体に含まれる回避技術を明らかにする．ここでは，(1) 回避のために取得するシステム情報（例：論理プロセッサの数），(2) システム情報を取得するための具体的な方法（例：SYSTEM_INFO.dwNumberOfProcessors の呼び出し），(3) 解析環境の判定に設定される閾値（例：4 コア未満）に焦点を当て，抽出した具体的な回避技術の情報を説明文として出力させる．次に，出力した回避技術に合わせてサンドボックスの耐性を強化するためのサンドボックスカスタマイズ手法を LLM に生成させる．入力には，検体のソースコードおよび出力した回避技術の説明文に加えて，次の 3 つの情報のいずれか，あるいはその組み合わせを提供することが想定される．

A. **サンドボックス解析システムの簡易情報**：システムを構成するゲスト OS，仮想化ハイパーバイザ，マルウェア解析ツールを記載したテキストファイル，および，公式ドキュメントに基づいて構築した際のそれぞれのシステムコンポーネントの設定値をリスト化した CSV ファイルを入力として与える．

B. **サンドボックス強化の具体的な実装例**：システムを構成するンポーネント毎のカスタマイズ方法を例示し，それぞれのコンポーネントに特化した強化方法と出力の形式を入力として与える．また，以下に示すカテゴリの実装の指針を入力として与える．なお，人手での操作を代替した自動実行への試みとして，PowerShell スクリプトを生成する手法を取り入れている．また，出力の形式を実装の対象・主体を明確に区別して組織化することにより，カスタマイズ手法の実装における変更手順が一意に分割され，回避技術とそれに対するアプローチ先の対応関係の検証が容易となる．
   (1) ゲスト OS 自動設定変更スクリプト作成
   (2) ゲスト OS 手動設定変更手順書作成
   (3) ハイパーバイザ手動設定変更手順書作成
   (4) 新規 API フックコード作成

C. **追加の具体的な実装例**：マルウェアの挙動を監視する解析ツールのモニタリングロジック関連のソースコード，および，ゲスト OS 向けの API フックライブラリのソースコードを入力として与える．なお，API フックコードを実装する際にコンパイルエラーが発生した場合，そのエラーメッセージを LLM に入力して該当箇所のバグ修正や実装改善を行なったフックコードを再生成させる．このように，ソースコードの生成とデバッグの過程を繰り返すことで，実用可能な API フックコードを出力し，カスタマイズの実装精度の向上を補助することができる．

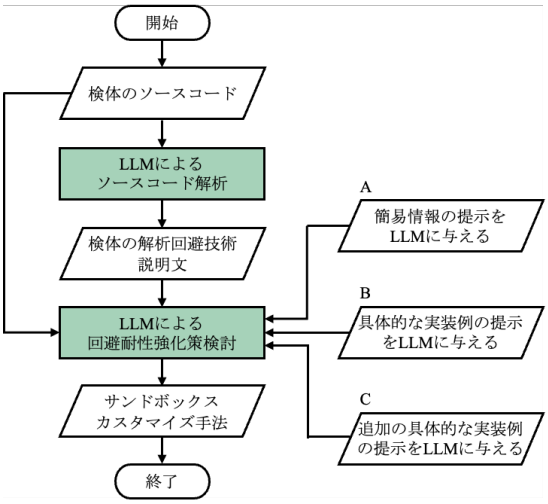

図 1　LLM を活用したサンドボックスの回避耐性強化の流れ．

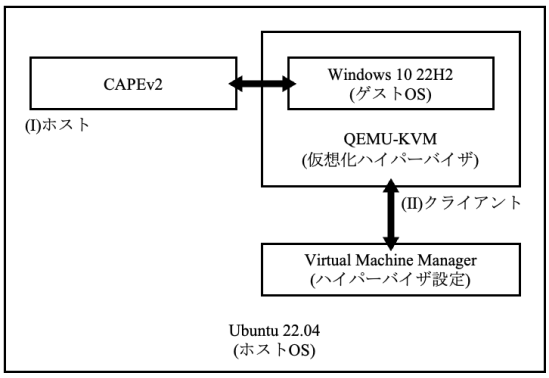

図 2　CAPE サンドボックス解析システムの構成

2.2　サンドボックスのカスタマイズ

提案手法は様々な種類のサンドボックスに適用可能であるが，3 章の評価実験では，オープンソースの CAPE サンドボックス解析システムを強化対象のサンドボックスとして用いた．CAPE サンドボックス解析システムの構成を図 2 に示す．以降では，この解析システムを用いてサンドボックスのカスタマイズの流れを説明する．

LLM を活用して生成される強化方法に応じてサンドボックスのカスタマイズ手順は異なる．はじめに，サンドボックス内の OS 上で実行する PowerShell スクリプトが生成された場合，そのテキストをゲスト OS 内部にコピーして PowerShell スクリプトを実行する．同様に，手動操作する手順書が生成された場合，その手順に従って OS 内部の操作を行う．仮想化ハイパーバイザ設定画面の GUI を手動操作する手順書が生成された場合，その手順に従って仮想マシンの設定を変更する．CAPE サンドボックスの設定ファイルに適用する API フックコードが生成された場合，そのソースコードをコンパイルして API フックコードを CAPE の設定ファイルに適用する．なお，コンパイル時に

エラーが出力される場合は，LLM により API フックコードの修正を行う．

提案手法による強化には，実装者による手動カスタマイズを要する手順を含むことがある．評価実験では，人手の介在が想定される操作については，実装者の能力差の影響を最小限に抑えるため，すべて作業をサイバーセキュリティを専攻する大学院生 1 名が行った．操作の手順書に曖昧な表現や，実施不能な操作指示が含まれると判断した場合は，推測で作業を行わず当該手法の実装を中断した．

2.3 提案手法の評価

提案手法の有効性を評価するために設定した，2 つの評価指標を説明する．また，提案手法を用いて回避耐性を強化したサンドボックスの有効性を評価する方法を説明する．

(1) **解析成功率：**疑似マルウェア検体群および Pafish 上の解析回避 PoC コードに対して，提案手法を適用したサンドボックスが解析を成功した割合を評価する．解析の成功・失敗の判断は，各検体に定義されている解析成否の判定基準（3.1 データセット）をもとに客観的に行う．実験前後での成功率の比較により，LLM が生成したカスタマイズ手法の有効性を検証する．

(2) **カスタマイズ自動化率：**PowerShell スクリプトおよび API フックコードを利用したカスタマイズ手法は，実装者による手動操作を要する手順を含むが，システムに実装されるカスタマイズは全て LLM の出力結果に依存し，人手による介入がなくその過程を自動化することができる．ここでは，カスタマイズの方法がスクリプト実行などで自動化されている割合と，それに基づく解析の成否を評価する．解析に成功した全てのカスタマイズ手法と比較して，自動化の達成度合いや成功率を定量的に提示し，実装者による判断を完全に排除した自動カスタマイズの有効性を検証する．

(3) **外部解析サービスとの比較：**既存の外部解析サービスである VirusTotal に用いられている CAPE サンドボックスにおける解析結果と，提案手法に従って初期状態からカスタマイズした CAPE サンドボックスによる解析結果を比較することで，回避技術を実装した疑似マルウェア検体の解析結果を評価する．

3. 評価実験

提案手法の有効性の評価を行った．また，提案手法を用いて強化されたサンドボックスの回避耐性の評価を行った．現在提供されている LLM サービスの種類は多岐に渡るが，OpenAI 社の「gpt-4.1」を API 経由で利用した．

3.1 データセット

初めに，論文 [9] で提案されている手法を用いて 61 個の回避技術を生成し，回避技術を反映した C++コードを LLM で生成・コンパイルして擬似マルウェア検体セットを作成した．各検体は単一の回避技術を実装しており，実行時にサンドボックス内で動作していると判断した場合には動作を停止し，それ以外では，スクリーンロックを行う悪性ペイロードを実行する．そのため，解析環境でのペイロード実行の有無をもって解析の成否を判定できる．公式ドキュメントに従って構築した直後の初期設定状態の CAPE サンドボックスでこれらの検体群を解析したところ，37 個が解析環境を検知して回避に成功した（解析に失敗した）．

続いて，サンドボックス環境検知ツールである Pafish [10] で公開されている PoC コードの一部を利用して 61 個の検体を作成した．これらの検体は，実行するとサンドボックスの検知結果がコンソール上に表示され，その結果を持って解析の成否を判定することができる．同様に，初期設定状態の CAPE サンドボックスでこれらの検体群を解析したところ，6 個が解析環境を検知して回避に成功した．

以降では，解析環境を検知して回避に成功した計 43 個の検体を評価実験の対象とする．なお，これらのデータセットに含まれる検体の解析回避技術は以下の 6 種類である．

- 解析実行時のユーザによる操作を検出
- OS 上のユーザによる操作痕跡・履歴を確認
- 長時間スリープによる解析時間の制限を回避
- 特定の解析ツール関連ファイルの有無を確認
- システムのハードウェア構成や環境情報を調査
- スリープ処理の実行状況や処理時間を解析

3.2 回避耐性強化方法の評価結果

LLM に与える情報やプロンプト設計を変化させることにより，サンドボックスの強化方法の生成に与える影響を調査した．以下に，実施した 3 つの実験の結果を示す．

実験 1（簡易情報を入力）：LLM に簡易的な動的解析システム構成情報のみを知識として与えた．この情報には，検体のソースコードおよびソースコード解析で抽出した回避技術の説明文に加え，サンドボックスのゲスト OS やハイパーバイザに関する情報が含まれる．これにより，LLM が自身の既存知識に依存して自由な記述を行う状況を促した．出力の形式は特に指定せず，自動化された手法（スクリプト等）と手動による手順書の両方の生成を期待した．
実験の結果：LLM は全ての検体に対して何らかのカスタマイズ手法を生成した（表 1）．ここで，LLM が自動化スクリプトとして提案した強化策のうち，実際に検体の解析に成功したのは，疑似マルウェア検体で 3 個，PoC コードで 0 個であった．また，LLM が手動でのカスタマイズ手順書として提案した強化策は，疑似マルウェア検体で 34 個，PoC コードで 6 個であった．これらの手順書に従い

手動実装を試みたところ，実際に解析に成功したのは，それぞれ 15 個と 2 個であった．この結果から，評価対象のマルウェア検体に対して LLM による生成手法が解析成功に貢献した割合は，47%（20/43 個）であった．LLM は，ユーザの操作痕跡やシステムのハードウェア構成等を確認するといった回避技術に対してのみ，解析成功に有効なカスタマイズ手法を生成する傾向が見られた．以上から，解析システムに関する十分な情報を与えず，LLM の既存知識に依存する場合でも，サンドボックスの解析成功率をある程度向上させることが可能であることがわかった．しかし，評価対象の検体全てに対して何らかの提案は行ったものの，実際に解析に成功した実装を提案できたのは半数以下であった．これは，実現性が不確実な手法が数多く生成されることを示している．この結果は，LLM の既存知識だけでは，多様な回避技術に網羅的に対応することが困難であることを示唆している．

実験 2（実験 1 の簡易情報に加えて具体的な実装例を入力）：LLM に，システム構成要素ごとのカスタマイズ実装例を加えて入力として与えた．具体的には，サンドボックスを構成するゲスト OS やハイパーバイザなど，各コンポーネントにおける最適解を一つずつ出力するよう形式を指定した．これにより，不確実な手法の生成を抑制し，より具体的な実装の提案を促した．
実験の結果：LLM は全ての検体に対して何らかのカスタマイズ手法を生成した．ここで，LLM が自動化スクリプトとして提案した強化策のうち，実際に検体の解析に成功したのは，疑似マルウェア検体で 18 個，PoC コードで 1 個であった．また，LLM が手動でのカスタマイズ手順書として提案した強化策に従い手動実装を試みたところ，実際に解析に成功したのは，それぞれ 5 個と 2 個であった．この結果から，評価対象のマルウェア検体に対して 65%（28/43 個）で解析に成功し，さらに，自動化された手法が成功に貢献した割合は，疑似マルウェア検体で 78%（18/23 個），PoC コードで 33%（1/3 個）と大幅に上昇した．なお，新規 API フックの実装はすべて失敗した．また，疑似マルウェア検体について，実験 1（簡易情報の提示）での実験結果と同様に，有効なカスタマイズ手法は，ユーザーの操作痕跡やハードウェア構成を確認する回避技術に対するものに限定された．以上から，解析システムを構成する各コンポーネントに特化したカスタマイズの具体的な実装例の指示は，解析成功率をさらに向上させ，カスタマイズ自動化率を大幅に上昇させることが明らかになった．これにより，LLM が持つ既存の知識に加えて，特定の技術的詳細と実装例を与えることが、より実用的な強化策を生成する上で極めて有効であることが示された．一方で，システムのクライアント側におけるカスタマイズは，ユーザー操作の痕跡やハードウェア構成の変更といった手法には有効であるものの，その適用範囲は限定的であることも判明した．

実験 3（実験 1 と 2 の情報に加えて追加の具体的な実装例を入力）：LLM に与えるプロンプト情報をさらに拡張し、挙動監視ソースコードと API フックライブラリを加えて入力として与えた．これにより，LLM は API フックの具体的な実装コードを生成できるようになった．また，生成されたコードの正確性を高めるため，実装の際にコンパイルエラーが発生した場合にはそのエラーメッセージを LLM に入力し，バグ修正や実装改善を最大 3 回まで試行する設定を設けた．実装には，最終的にコンパイルエラーが発生しないコードを用いた．
実験の結果：生成された 14 個の新規 API フックのうち，9 個がコンパイルエラーなく実装可能であった．そのうち，5 個が実際に解析成功に繋がった．LLM は全ての検体に対して何らかのカスタマイズ手法を生成した．ここで，LLM が自動化スクリプトとして提案した強化策のうち，実際に検体の解析に成功したのは，疑似マルウェア検体で 23 個，PoC コードで 1 個であった．また，LLM が手動でのカスタマイズ手順書として提案した強化策に従い手動実装を試みたところ，実際に解析に成功したのは，それぞれ 5 個と 1 個であった．この結果から，評価対象のマルウェア検体に対して 72%（31/43 個）で解析に成功した．成功した件数に占める自動化された手法の割合は、疑似マルウェア検体で 82%、PoC コードで 33%と，これまでの実験で最も高い水準を示した．特筆すべきは，新規 API フックの実装により，長時間のスリープによって解析時間の制限を回避する技術を持つ疑似マルウェア検体全てに対して，解析成功に有効なカスタマイズ手法を生成することができた．以上から，API フックコードの生成に焦点を当てたシステムは，実用可能なコードを生成できることが明らかになった．これにより，解析成功率とカスタマイズ自動化率の両方をさらに向上させることができた．また，このアプローチは，システムのクライアント側に限定されがちだったこれまでのカスタマイズ手法の適用範囲を拡張することができた．ホスト側におけるカスタマイズによって，解析実行時のユーザー操作の偽装や長時間スリープによる解析時間の制限無効化といった，より高度な回避技術にも有効に対処できることが示された．

3.3 公開サンドボックスサービスとの解析成功件数の比較

Google が運営し，外部に公開しているマルウェア検知ツールである VirusTotal に，データセットに含まれる全 122 個の検体を第三者に共有せずに検査が可能なプライベートスキャン機能を用いて解析した．VirusTotal では様々なセキュリティベンダーのツールを用いた解析を行うことができ，本項では，VirusTotal 上の CAPE サンドボックスにおける解析結果に注目することで，提案手法に対す

表 1 LLM に与える入力の違いによるサンドボックス回避耐性強化作の生成結果．ここで，カッコ内の数字は強化策を用いてカスタマイズしたサンドボックス上で解析回避検体を実行した時に解析に成功（検体が解析回避に失敗）した検体数を表す．

解析回避技術の種類	データセットの内訳	強化策生成数 (実験 1)		強化策生成数 (実験 2)		強化策生成数 (実験 3)	
		自動手法	手順書	自動手法	手順書	自動手法	手順書
(疑似マルウェア検体セット)							
解析実行時のユーザによる操作を検出	4(1)	0	1(0)	0	0	1(1)	0
OS 上のユーザによる操作痕跡・履歴を確認	25(17)	3(3)	14(10)	18(14)	2(2)	17(14)	2(2)
長時間スリープによる解析時間の制限を回避	3(3)	0	3(0)	0	0	3(3)	0
システムのハードウェア構成や環境情報を調査	27(15)	0	15(5)	5(4)	7(3)	11(5)	7(3)
スリープ処理の実行状況や処理時間を解析	2(1)	0	1(0)	0	0	1(0)	0
合計	61(37)	3(3)	34(15)	23(18)	9(5)	33(23)	9(5)
(解析回避 PoC コードセット)							
解析実行時のユーザによる操作を検出	6(3)	0	3(0)	3(1)	3(0)	3(1)	3(0)
OS 上のユーザによる操作痕跡・履歴を確認	1(0)	-	-	-	-	-	-
特定の解析ツール関連ファイルの有無を確認	31(0)	-	-	-	-	-	-
システムのハードウェア構成や環境情報を調査	22(3)	0	3(2)	0	3(2)	3(0)	3(2)
スリープ処理の実行状況や処理時間を解析	1(0)	-	-	-	-	-	-
合計	61(6)	0(0)	6(2)	3(1)	6(2)	6(1)	6(2)

表 2 実験 3 におけるコードデバッグの有効性．

検体に含まれる解析回避技術カテゴリ	カスタマイズ対象データセット	コードデバッグなし		コードデバッグあり		**解析に成功した検体数**
		コンパイル成功	成功	コンパイル成功	成功	
解析実行時のユーザによる操作を検出	1	0	-	1	1	1
OS 上のユーザによる操作痕跡・履歴を確認	17	3	1	4	2	2
長時間スリープによる解析時間の制限を回避	3	3	3	3	3	3
システムのハードウェア構成や環境情報を調査	15	2	0	7	1	1
スリープ処理の実行状況や処理時間を解析	1	1	0	1	0	0
合計 (LLM Quasi-Malware)	37	9	4	16	7	7
解析実行時のユーザによる操作を検出	3	1	0	3	0	0
システムのハードウェア構成や環境情報を調査	3	1	0	2	0	0
合計 (Pafish)	6	2	0	5	0	0

成功=解析に成功した手法

る評価を行った（表 3）．

データセット中の全 122 検体について，VirusTotal 上の CAPE で解析が成功したものは，77 個 (63%) であり，初期インストール状態の CAPE では，79 個 (65%) であった．それに対して，評価実験の実施において 3 つの入力情報を段階的に与えてカスタマイズでは，110 個 (90%) であった．

以上の結果より，本手法によって強化したサンドボックスの解析能力は，VirusTotal 上のサンドボックスおよび初期状態のサンドボックスを大幅に上回ることがわかる．これは，本提案手法がマルウェアの回避技術へのサンドボックスの耐性を強化させることが可能であることを示すとともに，VirusTotal 上のサンドボックスにカスタマイズの余地が大いにあることを示している．

3.4 考察

提案手法は，サンドボックスの回避耐性向上を LLM により支援するフレームワークであり，評価実験によりその有効性を明らかにした．しかし，設計上のシナリオや実験環境，LLM 活用の性質に起因する制約や適用範囲が存在する．主な適用範囲・限界を以下にまとめる．

解析環境の依存性：本手法の評価における検証は，Windows マルウェアの解析を CAPEv2 サンドボックスという特定のシステム構成下で実施しており，実装されたカスタマイズ手法はそれぞれのシステムコンポーネント別のアプローチをとる．そのため，Linux マルウェアや別種解析環境，あるいはサードパーティ製品を新たにシステムに組み込む場合については，その効果を検証できていない．

検体ソースコードの制約：本手法は，検体のソースコードを LLM に入力として与えることで回避技術を把握・抽出するアプローチをとる．そのため，解析回避 PoC のソースコードが存在する場合には，その回避技術に対してサンドボックスの耐性を強化するというユースケースに適しているが，実行ファイルのみが入手できる場合や，コードが暗号化されている場合については対応できていない．

4. 関連研究

4.1 サンドボックス検知耐性強化に関する研究

従来より，サンドボックス環境がマルウェアに検知され

表 3　公開サンドボックスサービスとの解析精度の比較

	解析回避検体データセット	VirusTotal CAPE	初期インストール状態 CAPE	3 段階カスタマイズ実装済み CAPE
LLM を用いて生成した擬似マルウェア検体群	61	23	24	52
Pafish	61	54	55	58
合計	122	77	79	110

にくくなるようにすることや，解析回避技術への耐性強化を目的とした多様な研究が進められている．Ether [11] はハードウェア仮想化を活用し，マルウェア実行を高い秘匿性をもつ環境へ分離することで，サンドボックスの検出を困難にしている．BareCloud [12] は複数の実行パスを追跡する設計とすることで，動的解析の対応範囲を広げ，さまざまな悪性動作の検知を図っている．Catcher [13] は，CPU キャッシュ挙動を監視する新たな手法を導入し，CPU レベルでのマルウェアによる回避行動の可視化を実現している．これらの手法に加え，実際の運用・研究の現場では Cuckoo や CAPE [8, 14] といったオープンソースサンドボックスが広く利用されている．これらのプラットフォームは分析機能を拡張できる自動化基盤を備えており，学術と産業の双方で解析活動を支えている．

4.2　解析回避技術への対応と検証基盤の発展

耐解析マルウェアに対応するため，回避挙動の体系化および検証ツールの開発も活発に行われている．MITRE ATT&CK フレームワーク [15] は防御回避手法の網羅的な分類を示し，サンドボックスの設計や検知ロジック策定の指針を提供している．Al-Khaser [16] や Pafish [10] といった解析環境テストツールは，代表的な解析回避技術を実装しており，サンドボックスの耐性評価やベンチマークに活用されている．また，SandPrint [17] のように，マルウェアが仮想環境固有の特徴のフィンガープリントを行い，検知を回避する手法も示されている．Check Point [18] 等のセキュリティーベンダー による実環境の事例共有も広く行われ，実践的な回避例および対策案が提案されている．

他にも，仮想マシン内への現実を模擬したシステムの痕跡の埋め込むことでサンドボックス検出を難しくする研究 [19, 20] も報告されている．しかしながら，これら多様な対応技術の適用は自動化が難しい場合が多く，新たな回避手法に適応するには分析者によるスクリプト作成やベンダへの改善要請が必要となる．これにより，迅速な対策が困難であり，人的リソースに大きく依存する点は依然として課題となっている [1].

4.3　LLM を活用した解析回避研究の進展

近年では，LLM を活用した解析回避型マルウェア生成およびその検証基盤の研究が進んでいる．松澤ら [9] は，概念実証（PoC）レベルの耐解析マルウェアデータセットを自動生成する手法を提案し，東ら [21] は Al-Khaser や Pafish といった既存フレームワークにおける回避手法コードの体系的分類を行った．さらに，PaPa ら [4] は，LLM を悪用することで，高度で多態的なマルウェアの作成障壁が低下するリスクを警鐘している．また，Check Point [5] や CyberArk [6] は AI により生成されたマルウェアが静的・動的解析を回避する例を実証しており，AI を活用したマルウェア攻撃の実効性が報告されている．これらの一連の研究は，AI 技術が攻撃に悪用されるリスクとともに，回避挙動検証の自動化・効率化を図る技術としての LLM 活用の可能性にも焦点をあてている．

5.　まとめと今後の課題

大規模言語モデル（LLM）を活用し，耐解析マルウェアに対応可能なサンドボックスの自動カスタマイズ手法を提案した．評価実験の結果，CAPE サンドボックスに対し，ゲスト OS，VM ハイパーバイザー，サンドボックス本体へのカスタマイズ手法を生成できることを確認した．実際に提案手法を用いて導出したカスタマイズ手法を CAPE サンドボックス環境に施した結果，LLM により生成した擬似マルウェア検体や Pafish の解析回避 PoC コードに対しても高い解析精度を得ることができ，従来のサンドボックスや外部サービスに比べて大きな性能の向上を示した．今後は LLM に入力する情報の最適化・プロンプト自動化の検討，ならびに他のサンドボックスでの評価を行う．

謝辞　本研究の一部は国立研究開発法人新エネルギー・産業技術総合開発機構（ＮＥＤＯ）の委託事業「経済安全保障重要技術育成プログラム／先進的サイバー防御機能・分析能力強化」（JPNP24003）によるものである．

参考文献

[1] M. Y. Wong, M. Landen, M. Antonakakis, D. M. Blough, E. M. Redmiles, and M. Ahamad, “An inside look into the practice of malware analysis,” *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2021.

[2] M. Botacin, “What do malware analysts want from academia? a survey on the state-of-the-practice to guide research developments,” *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2024.

[3] N. Galloro, M. Polino, M. Carminati, A. Continella, and S. Zanero, “A systematic and longitudinal study of evasive behaviors in windows malware,” *Computers & Security*, vol. 113, p. 102574, 2022.

[4] Y. M. Pa Pa, S. Tanizaki, T. Kou, M. van Eeten,

K. Yoshioka, and T. Matsumoto, "An attacker' s dream? exploring the capabilities of chatgpt for developing malware," in *Proceedings of the 16th Cyber Security Experimentation and Test Workshop*, ser. CSET '23, 2023, p. 10–18.

[5] Check Point Research, "Cybercriminals Bypass ChatGPT Restrictions to Generate Malicious Content," February 2023, accessed: 2025-07-27. [Online]. Available: https://blog.checkpoint.com/2023/02/07/cybercriminals-bypass-chatgpt-restrictions-to-generate-malicious-content/

[6] CyberArk Labs, "Chatting Our Way Into Creating a Polymorphic Malware," January 2023, accessed: 2025-07-27. [Online]. Available: https://www.cyberark.com/resources/threat-research-blog/chatting-our-way-into-creating-a-polymorphic-malware

[7] R. Tanabe, Y. M. Pa Pa, R. Tanabe, K. Yoshioka, K. Uchida, and S. Shirakawa, "A framework for iterative sandbox evasion code generation using attacker and defender llms," in *Proceedings of Security Summer Summit 2025*, ser. SSS '25, 2025.

[8] CAPE Project, "CAPE Sandbox Documentation," https://capev2.readthedocs.io/en/latest/, 2025, accessed: 2025-08-01.

[9] H. Matsuzawa, S. Kubo, Y. M. Pa Pa, R. Tanabe, and K. Yoshioka, "Investigating the impact of evasive malware created with llm on sandbox analysis," in *Proceedings of Computer Security Symposium 2024*, ser. CSS '24, 2024, p. 1148–1155.

[10] A. Ortega, "PaFish: Paranoid Fish - Sandbox Evasion Testing Tool," https://github.com/a0rtega/pafish, 2024, accessed: 2025-07-28.

[11] A. Dinaburg, P. Royal, M. Sharif, and W. Lee, "Ether: Malware analysis via hardware virtualization extensions," in *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*. ACM, 2008, pp. 51–62.

[12] D. Kirat, G. Vigna, and C. Kruegel, "Barecloud: Bare-metal analysis-based evasive malware detection," in *23rd USENIX Security Symposium*. USENIX, 2014, pp. 287–301.

[13] F. Zhang, K. Leach, A. Stavrou, H. Wang, and K. Sun, "Using hardware features for increased debugging transparency," *2015 IEEE Symposium on Security and Privacy*, pp. 55–69, 2015.

[14] Cuckoo Project, "Cuckoo Sandbox Documentation," https://cuckoo.readthedocs.io/en/latest/, 2025, accessed: 2025-08-01.

[15] MITRE Corporation, "MITRE ATT&CK® Framework: Adversarial Tactics, Techniques, and Common Knowledge," https://attack.mitre.org/, 2024, accessed: 2025-07-28.

[16] L. Kelemen, "Al-Khaser: Windows Subsystem Evasion Techniques Tester," https://github.com/LordNoteworthy/al-khaser, 2024, accessed: 2025-07-28.

[17] A. Yokoyama, K. Ishii, R. Tanabe, Y. Papa, K. Yoshioka, T. Matsumoto, T. Kasama, D. Inoue, M. Brengel, M. Backes, and C. Rossow, "Sandprint: Fingerprinting malware sandboxes to provide intelligence for sandbox evasion," in *International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2016.

[18] C. P. S. Technologies, "Sandbox Evasions Resource Portal," https://evasions.checkpoint.com, 2024, accessed: 2025-07-28.

[19] N. Miramirkhani, E. Kang, and N. Nikiforakis, "Spotless sandboxes: Evading malware analysis systems using wear-and-tear artifacts," in *Proceedings of the 38th IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, pp. 1001–1018.

[20] fr0gger, "RocProtect-V1: Malware Sandbox Evasion Detection Framework," https://github.com/fr0gger/RocProtect-V1, 2022, accessed: 2025-07-28.

[21] Y. Higashi, H. Matsuzawa, R. Tanabe, Y. M. Pa Pa, and K. Yoshioka, "Poster: Conventional llm use struggles to generate sandbox evasion code from unseen categories," in *The 10th IEEE European Symposium on Security and Privacy*, ser. Euro S&P, 2025.