一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

サンドボックス解析を回避する実証コード群のLLMを用いた生成

松澤　輝[†]　東　志拓[†]　グエンチ ヴァン アン[††]　田辺　瑠偉[††]　インミン パパ[††]
吉岡　克成[††,†††]

† 横浜国立大学　〒240-8501 神奈川県横浜市保土ケ谷区常盤台 79-1
†† 横浜国立大学先端科学高等研究院　〒240-8501 神奈川県横浜市保土ケ谷区常盤台 79-5
††† 横浜国立大学大学院環境情報研究院　〒240-8501 神奈川県横浜市保土ケ谷区常盤台 79-7
E-mail: †{matsuzawa-hikaru-nx,higashi-yukihiro-rd}@ynu.jp,
††{nguyen-anh-xd,tanabe-rui-xj,yinminn-papa-jp,yoshioka}@ynu.ac.jp

あらまし　サンドボックス解析は，マルウェアの挙動を把握する上で不可欠な技術である一方，解析環境を検知して解析を逃れる「回避型マルウェア」が深刻な問題となっている．既存研究では回避耐性の向上手法が提案されているが，評価用検体セットの不足により十分な評価が難しく，結果としてサンドボックスの回避耐性は依然として課題として残っている．そこで本研究ではサンドボックスの回避耐性評価を目的に，多様な実証コードからなる大規模データセットを構築する．具体的には，言語や実装方針が異なる既存評価ツールの PoC を LLM を用いて統一的な形式で再実装するとともに，先行研究で定義された回避技術のカテゴリ定義に基づき，LLM を用いて回避手法を探索・実装することで，既存ツールで十分にカバーされない回避技術を補完する．さらに，生成した PoC を基に LLM で実装を書き換えることで，単一の回避技術に対する実装バリエーションを拡張する．一方で，LLM が生成したコードには誤りが含まれ得るため，ビルド可否，正常実行，一定の回避性能を検証する 3 段階の手順を設け，基準を満たす PoC のみをデータセットに採用する．実験では 4,508 件の PoC を生成し，検証の結果 1,520 件をデータセットに採用した．作成したデータセットを商用サンドボックスに適用した結果，593 件（39.0%）が実行環境を正しくサンドボックスと判定した．一方，既存ツールに含まれる 246 件の PoC を同一条件で実行した場合，回避に成功したのは 56 件（22.8%）にとどまった．以上より，本データセットは既存ツールより規模・性能の両面で優れており，サンドボックスの回避耐性評価に有用である．
キーワード　サンドボックス解析，回避型マルウェア，大規模言語モデル

LLM-Based Generation of Proof-of-Concept Code for Evading Sandbox Analysis

Hikaru MATSUZAWA[†], Yukihiro HIGASHI[†], Anh NGUYEN THI VAN[††], Rui TANABE[††], Yin MINN PA PA[††], and Katsunari YOSHIOKA[††,†††]

† Yokohama National University
†† Institute of Advanced Sciences, Yokohama National University
††† Faculty of Environment and Information Sciences, Yokohama National University
E-mail: †{matsuzawa-hikaru-nx,higashi-yukihiro-rd}@ynu.jp,
††{nguyen-anh-xd,tanabe-rui-xj,yinminn-papa-jp,yoshioka}@ynu.ac.jp

Abstract　Sandbox analysis is essential for understanding malware behavior, yet evasive malware can detect analysis environments and avoid observation. Although many defenses have been proposed, evaluating sandbox evasion resistance remains difficult due to the lack of comprehensive test samples. We present a large-scale dataset of proof-of-concept (PoC) programs for assessing sandbox evasion resistance. Using large language models (LLMs), we unify and reimplement PoCs from existing tools, extend coverage by generating additional techniques guided by prior technique taxonomies, and produce multiple implementation variants per technique. To mitigate errors in LLM-generated code, we retain only PoCs that pass a three-stage validation pipeline: buildability, correct execution, and verification of evasion-related behavior. We generated 4,508 PoCs with an OpenAI model and curated 1,520 validated PoCs. When tested on a commercial sandbox, 593 PoCs (39.0%) detected the environment as a sandbox, compared with 56 (22.8%) of 246 PoCs from existing tools under the same conditions. Our dataset is larger and more effective than tool-based PoC sets, enabling more rigorous evaluation of sandbox evasion resistance.
Key words　Malware Dynamic Analysis, Sandbox Evasion, Large Language Model

1. はじめに

サイバー攻撃の増加に伴いマルウェア検体数は増大を続けており，限られた時間で大量の検体を効率的に解析することが求められている．この課題に対し，サンドボックスと呼ばれる解析環境上で検体を実行し，短時間で挙動を把握するサンドボックス解析技術は，実運用において重要な役割を担っている．

一方，サンドボックスの多くはその性質上，実際のユーザが使用する環境（以降，実ユーザ環境）とは異なる特徴を有することが指摘されている．例えば，仮想化技術に起因するデバイス名やドライバ，MAC アドレス（OUI），BIOS 文字列，プロセス構成の差異，ならびにユーザ操作や利用履歴の欠如などが報告されている [1], [8], [12]．このような差異を利用してサンドボックス上での解析を回避する機能を備えた「回避型マルウェア」は深刻な問題となっており [2]，サンドボックスの回避耐性を定量的に評価し，改善に結び付けることが重要となっている．

この目的のため，代表的な回避技術を実証コード（Proof-of-Concept，以降 PoC）として実装し，サンドボックスの耐性評価に用いるオープンソースツールが公開されている [3], [10]．しかしながら，回避型マルウェアが用いる回避技術は時代とともに多様化しており，過去に観測された回避技術のみを前提とした評価では，最新の回避技術やその周辺技術を十分にカバーできない可能性がある．さらに，同一の回避技術であっても，情報取得の経路や API 選択など実装上の差異によって検知の成否が変化し得るため，回避耐性評価には，回避技術の種類だけでなく実装バリエーションを含む検体セットが必要となる．一方で，このような大規模かつ多様な PoC 群を人手で継続的に整備することはコスト面で困難である．

そこで本研究では，回避技術を網羅し，かつ実装差の影響も評価可能な PoC 群からなる大規模データセットを構築する．具体的には，大規模言語モデル（LLM）を用い，先行研究で定義された回避技術のカテゴリ定義を入力として与えることで，当該カテゴリ内での回避アイデアを探索し PoC として実装する．これにより，過去回避技術に限定せず，想定され得る周辺技術を含む回避技術を幅広く実装する．さらに，生成した PoC を基に LLM で実装を書き換えることで，同一カテゴリに対する複数の実装バリエーションを展開し，実装差がサンドボックス回避に与える影響を評価可能とする．加えて，生成された PoC について（i）実行ファイルへの変換可否，（ii）正常実行可否，（iii）サンドボックスおよび実ユーザ環境での実行結果に基づく一定の有効性，を段階的に確認することで，基準を満たすもののみをデータセットとして採用する．

本論文の主な貢献は以下の 2 点である．(i) 多様な回避技術とその実装バリエーションを含む，大規模な回避 PoC データセットを構築する．(ii) ビルド可否・正常実行・サンドボックス／実ユーザ環境での判定検証を構築プロセスに組み込み，採用 PoC の有効性を担保する．また，ユースケースとして実際に商用サンドボックスに適用し，回避耐性評価への有用性を示す．

表 1　サンドボックス回避手法の例

手法	概要
メモリサイズ評価	実行環境のメモリ容量を取得して閾値と比較し，低容量であればサンドボックスと推定する．
マウス操作監視	マウス移動・クリックを監視し，操作が観測されれば実ユーザ環境と推定する．
スリープによる遅延	解析時間制限を見越して長時間スリープし，制限内に悪性挙動を発現させない．

2. 関連研究

回避型マルウェアは，実行環境がサンドボックスである可能性が高いと判断した場合に，悪性挙動を抑制・遅延させることで動的解析における観測可能性を低下させる．このとき，回避技術は環境の特徴量を収集して実行環境を推定する技術，すなわち環境推定能力として捉えられる．代表例を表 1 に示す．

先行研究において，回避技術の体系化が行われている．Galloro らは回避技術を技術ベースで 16 カテゴリに分類している [4]．また，MITRE ATT&CK では，Virtualization/Sandbox Evasion（T1497）[9] を以下に示す 3 つのサブカテゴリに分類しており，これらの分類は回避技術を整理・比較する上で有用である．

- **System Checks（T1497.001）**：VM やサンドボックスに特有のアーティファクト（ハードウェア/OS 構成，ドライバ，プロセス等）を検査し，解析環境かどうかを推定する．
- **User Activity Based Checks（T1497.002）**：マウスやキーボード入力，アイドル時間など，実ユーザ活動の有無を観測して自動解析環境を推定する．
- **Time Based Evasion（T1497.003）**：稼働時間や時刻，sleep の実行時間などの時間特性を利用し，実行を遅延させる/観測時間をやり過ごすことで解析を回避する．

これらの回避技術は，大規模分析により一定割合のマルウェアに含まれることが報告されている．Maffia らは 18 万超の Windows マルウェア検体の動的解析に基づき，少なくとも 1 つの回避技術を用いる検体の割合が 2016 年の約 30% から 2020 年には 40% 超へ増加したと報告した [7]．Galloro らの縦断的分析においても，回避技術を含む検体の出現が長期的に増加傾向であることが示されている [4]．また，採用される手法の選好も変化しており，VM 検知の比率が低下する一方で anti-debugging が増加するなど，手法トレンドの変化が報告されている [7]．このため，回避耐性の評価は特定の既知手法に固定せず，継続的な更新を前提とすることが望ましい．

加えて，実際にサンドボックス解析の回避可否を検証した研究が行われている．横山らは，オンラインサンドボックス解析サービスから得られた特徴量に基づき，サンドボックス環境と実ユーザ環境を判別できることを示した [12]．また，Miramirkhani らは，実利用に伴う利用痕跡（wear-and-tear）の欠如そのものが強い識別手掛かりとなり得る点，ならびに公開サンドボックスが脆弱になり得る点を指摘している [8]．

これらに対する防御側の方向性としては，サンドボックスを

実ユーザ環境に近づけるアプローチ，例えば利用痕跡やユーザ操作を付与する手法が提案されている [5],[6],[11]．ただし，対策の設計・改善には，「どの回避技術に対して，どの程度の耐性があるか」を評価できる枠組みが不可欠である．

そこで，回避技術を PoC として集約してサンドボックスの評価に利用する運用が広く行われてきた．代表例として Pafish [10] と Al-khaser [3] がある．いずれもサンドボックスが回避され得る点の特定に有用であり，多くの研究で利用されてきた．しかし，回避技術のトレンド変化や実装の多様化を踏まえると，既存 PoC データセットには課題が残る．第一に，過去に観測された手法のみを対象としているため，その周辺手法や将来的に攻撃者が採用し得る手法などを十分にカバーできていない．第二に，サンドボックスおよび実ユーザ環境の変化に伴い，現行環境では有効性が低下した PoC が含まれる可能性がある．第三に，各回避技術について単一の実装パターンしか提供されておらず，実装の異なる亜種に対する評価が難しい．特に第三の点について，先行研究では，同一の回避技術であっても実装の違いにより回避の成否や，回避挙動の観測可否が変化し得ることが報告されている [13]．しかし，このような実装差の影響を体系的に評価できる PoC データセットは整備されていない．

本研究の位置付け：本研究では，サンドボックス解析を回避する PoC 群からなる大規模データセットを構築することで，サンドボックスの回避耐性評価において，回避技術を実装した検体セットが不足しているという課題の解決を目指す．そのため，本データセットは，回避技術のカバレッジ確保と，同一手法に対する複数の実装方法の実現を要件とする．

3. データセット構築方法

本章では，サンドボックス解析を回避する PoC データセットの構築方法を述べる．本研究では，データセット構築を 3 つの STEP に分けて行う．その流れを図 1 に示す．以降では，LLM を用いて生成したコードを PoC 候補と呼び，後述の条件をすべて満たしたものを PoC と呼ぶ．

STEP1 では，LLM を用いて既知の回避技術を実装し，ベースとなる PoC 候補を生成する．STEP2 では，それらの PoC 候補の判定ロジックを保ったまま LLM で実装を書き換え，実装バリエーションを拡張する．また，STEP1 および STEP2 の生成後には，ソースコードから実行形式ファイルへのビルド可否と，実行可能性を確認する．最後に STEP3 では，PoC 候補を複数のサンドボックス環境および実ユーザ環境で実行して回避機能を検証し，データセットへの採用可否を判定する．

3.1 PoC の基本設計

本研究で作成する PoC の実行時の流れを図 2 に示す．はじめに，事前に設定した回避技術に基づき，実行環境の特徴（システム情報等）を取得してサンドボックスか実ユーザ環境かを判定する．実行環境の判定には閾値や条件を用いるがその基準は柔軟に変更可能である．そして，サンドボックス環境と判断した場合には判定結果を出力して終了し，実ユーザ環境と判定した場合には悪性挙動を示してから判定結果を出力して終了する．

ここで，サンドボックスの多くは実行した検体の挙動を観測して解析レポートを作成することが想定される．そこで，サンドボックス判定時は `Sandbox`，実ユーザ環境判定時は `RealUser` という文字列を含むレジストリキーを作成することで，解析レポートに判定結果が記載されるようにする．一方，実ユーザ環境で PoC を実行した場合，サンドボックスのような挙動観測や解析レポート出力を利用できない．そのため，判定結果を機械的に回収するための終了コードを定める．具体的には，サンドボックスと判定した場合は 100，実ユーザ環境と判定した場合は 200 を返して終了する．（それ以外の実行時エラーは別コード）．加えて，取得値・閾値・判定理由・失敗理由等を標準出力し，結果の解釈とデバッグ可能性を確保する．

3.2 STEP1：ベース PoC の生成

STEP1 では，(A) 既存ツールに含まれる PoC の再実装を行う．ただし，既存ツールは過去に観測された回避技術を中心に整理されたものであり，その周辺技術や派生的な実装まで十分にカバーできていない．そこで，(B) 既存の回避カテゴリに基づく回避技術アイデアの探索・実装を行う．以降では，これらを合わせてベース PoC 候補と呼ぶこととする．また，検証実験では，回避カテゴリの定義として MITRE ATT&CK の Virtualization/Sandbox Evasion（T1497）の 3 カテゴリ [9]，および Galloro らの 16 カテゴリ [4] を併用する．前者のカテゴリは回避の方向性を大局的に捉えやすく，後者のカテゴリは技術要素をより具体的に掘り下げやすい．そのため，両者を併用することで，アイデア探索を広くかつ細かく行うことを狙う．

(A) **既存 PoC の再実装**：既存ツールに含まれる PoC は，実装言語・出力形式・実行フローがツールごとに統一されていない．そこで，既存 PoC を入力として LLM に与え，3.1 節の基本設計に基づきベース PoC 候補として再実装する．

(B) **回避カテゴリに基づく回避技術の探索・実装**：LLM に回避技術のカテゴリ定義を与え，当該カテゴリに合致する回避技術を探索・実装することで，未整理な回避技術も含めた網羅的な PoC 候補の構築を図る．具体的には，特定カテゴリの定義に沿って「サンドボックスと実ユーザ環境の間で差が生じ得る観測量（システム情報等）」を探索し，それらの観測量を用いてサンドボックス環境判定を行うコードとして実装する．

最後に，両方のアプローチで生成した PoC 候補を実行形式のファイルに変換し，ビルド可否および正常に実行可能であるかを確認する．ビルドに失敗した場合には，エラー内容を LLM へ与えてデバッグを最大 3 回まで繰り返す．

3.3 STEP2：実装書き換えによる多様化

STEP2 では，STEP1 で生成したベース PoC 候補を入力として，論文 [13] で提案されている手法に基づき，判定に用いる情報の取得や操作の実装を別手段に置き換えるよう LLM に指示し，コードを書き換える．例えばメモリ容量の取得は，Windows

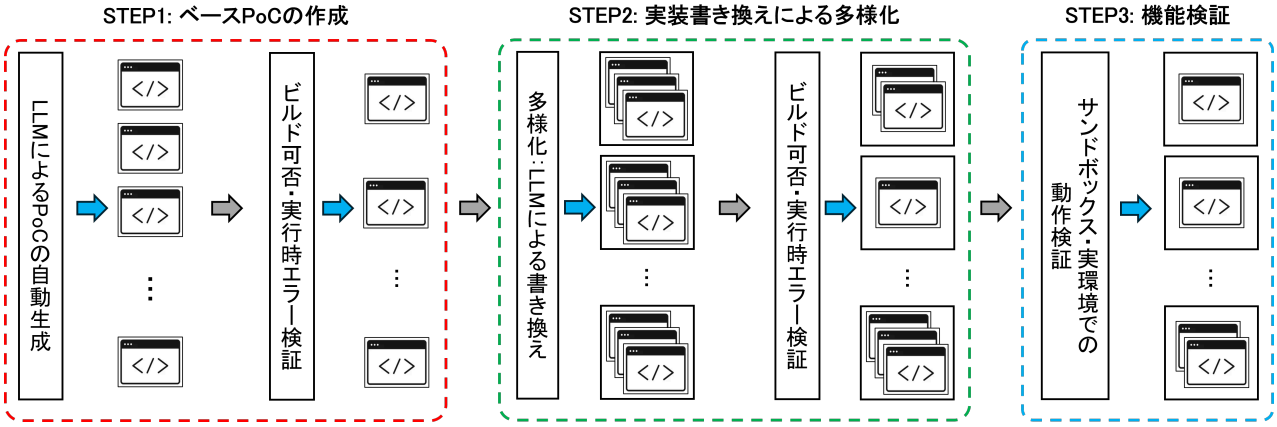

図 1　サンドボックス解析を回避する PoC データセット構築の流れ

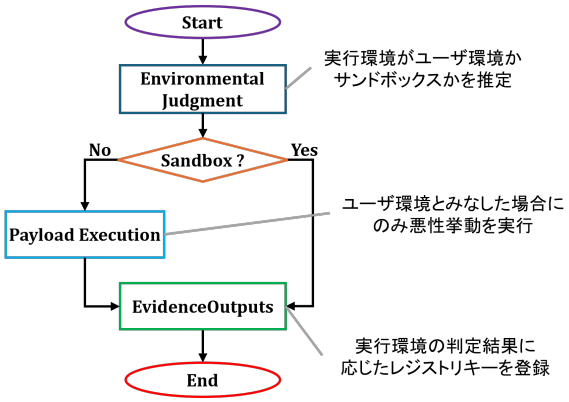

図 2　PoC 実行時の流れ

API 呼び出し，WMI 参照，PowerShell や CIM 経由など複数の経路で実現できる．本研究では，判定ロジックそのものは維持したまま，環境推定に用いる情報の取得や操作部分のみを別の API や取得経路へ置換し，複数の実装バリエーションを生成することを多様化と呼ぶ．なお，多様化によって生成した PoC 候補についても同様に，ビルド可否と実行可能性を確認する．

3.4　STEP3：回避性能の検証

STEP3 では，STEP1 および STEP2 で動作確認を終えた PoC 候補について，回避耐性評価に資する挙動が再現できるかを確認し，データセットへの採否を決定する．本研究における採用の基本方針（条件）は，実ユーザ環境では正しく `User` と判定され，かつ少なくとも一つ以上のサンドボックス環境で `Sandbox` と判定されるか，もしくは回避技術の性質に照らして意味のある差が観測されることである．これにより，回避耐性評価に有用な PoC 候補をデータセットに採用する．

ただし，回避技術には，明示的に環境判定へ到達するものと，判定に到達しないこと自体を狙うものが混在する．このため，一律に `Sandbox` 判定への到達のみを要件とすると，長時間遅延など本質的に未到達を引き起こす候補を不適切に除外し得る．そこで，MITRE ATT&CK の T1497 に基づいて PoC 候補を 3 つのカテゴリに分類し，カテゴリごとにデータセットへの採否条件を設ける．表 2 に具体的な採否条件を示す．

4.　データセット構築結果

本章では，PoC の生成・多様化・機能検証を行った結果をまとめる．

回避技術の PoC 作成は単なるコード生成にとどまらず，Windows 内部仕様の理解に加え，回避技術の探索・設計に関する推論を要する．本タスクに最適なモデルが現時点で確立しているとは言い難いため，本研究では実装能力と推論能力の両面で実績のある OpenAI モデルを採用した．PoC の生成およびデバッグには gpt-5.1 を，多様化には gpt-5.1-codex-mini を API 経由で利用した．なお，本実験で対象とする PoC は，Windows 10/11 上でユーザ権限により実行可能な回避技術に限定し，実装言語は C++ とする．また，実行結果の確認には，第 3. 章で定義したレジストリキーおよび終了コードを用いた．

4.1　STEP1：ベース PoC 候補の生成結果

はじめに，前述の (A) 既存 PoC の再実装，および (B) 回避カテゴリに基づく回避技術の探索・実装，の 2 つのアプローチでベース PoC 候補を生成した．既存 PoC は Pafish [10] と Al-khaser [3] から収集した 249 件を対象とした．回避カテゴリに基づく生成では，MITRE ATT&CK の 3 カテゴリについて各 50 件（計 150 件），Galloro らの 16 カテゴリについて各 20 件（計 320 件）を試行し，全体で 719 件の PoC 候補を生成した．また，生成後にはビルド可否と，実行可能性を確認した．表 3 に STEP1 の生成数とビルド可否・正常実行の確認を実施後の結果を示す．この結果から，合計 719 件の PoC 候補のうち，629 件がビルド可能かつ正常に実行可能であることを確認した．

4.2　STEP2：多様化の結果

続いて，STEP1 で確認した 629 件のベース PoC 候補を入力として，各ベース PoC 候補に対して最大 10 回の書き換えを試行し，同一の判定ロジックを維持したまま，取得経路や API 選択のみが異なる実装パターンの PoC 候補を生成した．また，これらについても，ビルド可否および実行可能性の確認を行った．表 4 に STEP2 の結果を示す．629 件のベース PoC 候補に対して合計 6,290 回の書き換えを試行した結果，5,338 件で書き換えに成功し，そのうち 3,879 件がビルド可能かつ正常に実行可

表 2　データセットに含める PoC の採否条件

MITRE ATT&CK のカテゴリ	データセットへの採用条件
System Checks T1497.001	システム情報取得と閾値比較により明示的に判定へ到達する PoC 候補を対象とし，実行時間上限を 5 分として，実ユーザ環境で常に `User` 判定となり，かつ少なくとも 1 つ以上のサンドボックスで `Sandbox` 判定となるものを採用する．
Time Based Checks T1497.003	長時間の遅延や操作を伴う PoC 候補を対象とし，実行時間上限を 30 分として，実ユーザ環境で 30 分以内に `User` 判定へ到達し，かつ少なくとも 1 つ以上のサンドボックスで 30 分以内に `Sandbox` 判定へ到達する，または判定自体に到達しないものを採用する．
User Activity Based Checks T1497.002	ユーザ操作の有無に依存して判定が変化する PoC 候補を対象とし，当該操作によって `User` と `Sandbox` の判定が分岐すること，かつ少なくとも 1 つ以上のサンドボックスで `Sandbox` 判定となるものを採用する．

表 3　STEP1：ベース PoC 候補の生成と確認結果

	(i) 既存 PoC	(ii) Galloro 分類 [4]	(iii) MITRE 分類 [9]	合計
生成数	249	16 × 20	3 × 50	719
ビルド可能	246	269	146	661
正常に実行可能	235	262	132	629

表 4　STEP2：多様化と確認結果

	件数
書き換え試行数	629 × 10 = 6290
書き換え成功	5,338
ビルド可能	4,574
正常に実行可能	**3,879**
PoC 候補の合計	**4,508(629+3,879)**

能であることを確認した．

4.3　STEP3：回避性能の検証結果

最後に，ベース PoC 候補 629 件と多様化後の PoC 候補 3,879 件を合わせた，合計 4,508 件の PoC 候補を対象に，MITRE ATT&CK T1497 の 3 類型に基づいて事前にカテゴリ分類した上で，実ユーザ環境およびサンドボックス群で実行し，サンドボックス解析を回避する PoC データセットを構築した．

PoC 候補の機能検証に用いた環境群を表 5 に示す．検証環境は，(i) 実ユーザ環境，(ii) デフォルト状態のサンドボックス，(iii) マルウェア解析サービス VirusTotal に含まれるサンドボックスの 3 種類とし，それぞれ 2 つの環境，合計 6 つの環境を用意した．実ユーザ環境には，研究チームが保有し日常的に利用している Windows PC を用いた．デフォルト状態のサンドボックスには，VirtualBox 上の Windows 環境および Windows に標準搭載されている Windows Sandbox を用いた．ここでデフォルト状態のサンドボックスとは，環境構築直後であり，サンドボックス特有の特徴を隠蔽しておらず，ユーザ利用に伴う履歴や状態がほとんど存在しない環境を指す．一方，VirusTotal のサンドボックスは公開解析サービスとして運用される性質上，一定の対策や解析機構が導入されている点で，デフォルト状態のサンドボックスとは性質が異なる．

なお，検証条件を統一するため，実ユーザ環境およびデフォルト状態のサンドボックスでは PoC 候補群を無操作状態で実行した．また，各 PoC 候補が返す判定結果は `User/Sandbox` の 2 値に加え，実行時間上限内に判定に到達しないケースを `Timeout`，実行時エラーを `Error` として記録した．`Timeout` には，解析時間の制約により挙動が観測されない場合と，実装不備やハング等により判定に到達できない場合の両方が含まれ得るため，本研究ではその性質に応じて採否条件を分離した．

検証結果：PoC 候補を MITRE ATT&CK T1497 で定義される 3 カテゴリに事前分類した結果，System Checks が 3,872 件，Time Based Checks が 378 件，User Activity Based Checks が 258 件であった．表 6 に結果を示す．この事前分類に基づき，表 2 のカテゴリ別採否条件を適用したところ，採用数は System Checks が 1,151 件，Time Based Checks が 189 件，User Activity Based Checks が 180 件となり，合計 1,520 件のサンドボックス解析を回避する PoC からなるデータセットを構築した．

表 5　STEP3：PoC 候補の機能検証に用いた環境

表記	概要
実ユーザ環境 A	日常的に利用しているマシン（Windows 10）
実ユーザ環境 B	日常的に利用しているマシン（Windows 11）
サンドボックス A	デフォルト状態の VirtualBox
サンドボックス B	デフォルト状態の Windows Sandbox
サンドボックス C	VirusTotal 上のサンドボックス
サンドボックス D	VirusTotal 上のサンドボックス

表 6　STEP3：PoC 候補の機能検証結果

カテゴリ（MITRE T1497）	PoC 候補数	採用数
System Checks（T1497.001）	3872	1,151
Time Based Checks（T1497.003）	378	189
User Activity Based Checks（T1497.002）	258	180
合計	4,508	**1,520**

5.　商用サンドボックスの検知実験

本章では，構築したデータセット（最終採用 1,520 件の PoC）を商用サンドボックス（以降では *Commercial-SB* と呼ぶこととする）へ適用した時の判定結果を分析する．また，既存ツールに含まれる PoC との判定結果を比較する．

なお，各 PoC は実行環境の判定結果に応じて異なるレジストリキーを作成する．そのため，Commercial-SB の解析レポートから当該レジストリ操作を抽出し，`Sandbox` 判定，`User` 判

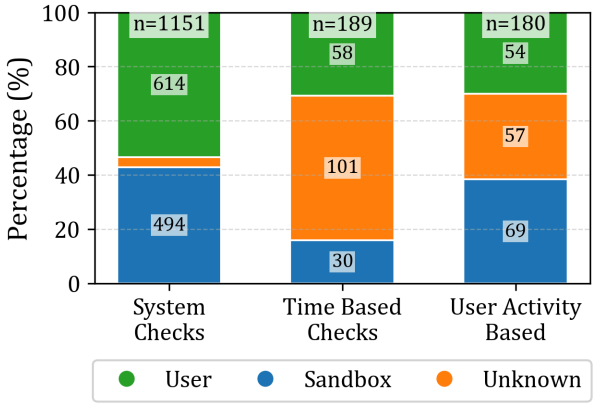

図 3　Commercial-SB におけるカテゴリ毎の判定結果
（Sandbox 判定は PoC が解析環境の検知に成功）

定，およびどちらも観測できない場合を Unknown（判定キー未観測）として分析する．また，Sandbox 判定が一定数見られる場合には，実ユーザ環境との差が手掛かりとして残っている可能性がある，という観点で結果を解釈する．

5.1　商用サンドボックスの判定結果

Commercial-SB でデータセットを実行したところ，593 件の PoC が Sandbox 判定，201 件の PoC が Unknown 判定，726 件の PoC が User 判定となった．回避技術のカテゴリ毎の判定結果を図 3 に示す．(i) System Checks では User 53.3%，Sandbox 42.9%，Unknown 3.7% であった．Sandbox 判定が一定割合を占めることから，実ユーザ環境との差異が多く存在していることが示唆される．(ii) Time Based Checks では Unknown が 53.4% と最多であった．本カテゴリは長時間を伴う操作を利用するため，解析時間内に判定が完了しなかった可能性もあるが，それ以外に実行時エラーが発生した可能性も考えられるため，利用者の立場から行う今回の評価では内訳を分離できない．一方で，本データセットは実ユーザ環境での正常実行を確認していることから，Unknown のうち少なくとも一部は意図通りに回避に成功した PoC が存在すると考えられる．(iii) User Activity Based Checks では Sandbox 38.3%，Unknown 31.7%，User 30.0% であった．User 判定が一定数存在することは，少なくとも一部の検体で一般的なユーザ操作が再現され得ることと整合する．

Commercial-SB で Sandbox 判定となった PoC の例を表 7 に示す．表 7 に示す観測点は，主に二種類に整理できる．一つは，CPU や RAM，ストレージ容量，周辺機器の有無といった資源や機器に関する情報である．もう一つは，イベントログ件数，インストール済みアプリ，Downloads やごみ箱などの利用痕跡に関する情報である．とくに利用痕跡は実ユーザ環境では自然に蓄積される一方，サンドボックスで同程度の状態を一貫して再現することは難しく，判定の手掛かりとして残りやすい．

以上から，これらの観測点を整理することで，実ユーザ環境との差がどこに現れているかを把握することができ，サンドボックスの回避耐性を改善するための具体的な指針となる．このように，本データセットを用いた回避耐性の評価は，サンドボックスの回避耐性向上に有用であることが確認できた．

表 7　Sandbox 判定となった PoC の具体例

観測指標	実ユーザ環境とみなす条件の例
BIOS メーカー名	Manufacturer が主要ベンダ名に含まれる．
CPU 論理コア数	論理コア数が 8 以上である．
総 RAM 容量	物理メモリ総量が 15GB 以上である．
Application ログ件数	ログ総レコード数が 15,000 以上である．
インストール済みアプリ	一般的アプリ名の一致が 2 件以上ある．
ストレージ容量	論理ドライブ容量が 450GB 以上である．
拡張子の種類数	ファイル拡張子が 10 種類以上である．
ごみ箱内アイテム数	ごみ箱内アイテム数が 100 以上である．
大容量ファイルの有無	Downloads 直下に 50MB 超のファイルが 1 つ以上ある．
Downloads のアイテム数	Downloads 直下のアイテム数が 100 以上である．
物理プリンタ	物理プリンタが 1 台以上存在する．
キーボード操作	60 秒以内にキー入力を 1 回以上検知する．
スクロール操作	60 秒以内にスクロールイベントを 1 回以上検知する．
Sleep の実時間検証	120 秒スリープの実時間誤差が 10 秒以内である．
稼働時間	稼働時間が 1 時間以上である．

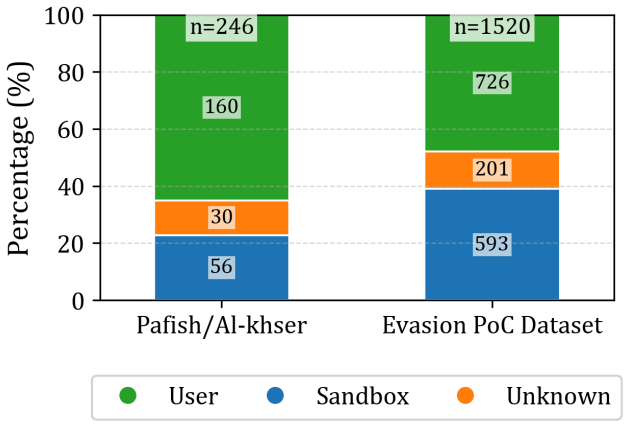

図 4　Commercial-SB における判定分布の比較（既存データセット vs. 本研究データセット）

5.2　既存ツールとの比較

既存ツールである Pafish および Al-khaser に含まれる PoC 合計 246 件と，本研究で構築したデータセット 1,520 件を Commercial-SB 上で実行した結果を比較した．それぞれの判定結果を図 4 に示す．この結果から，既存ツールの Sandbox 判定は 22.8%（56 件）であったのに対し，本データセットでは 39.0%（593 件）であった．Sandbox 判定を示す PoC が多いことは，Commercial-SB に残る実ユーザ環境との差分をより多く顕在化できていることを意味し，サンドボックスの改善点の抽出に直結する．以上より，本データセットは既存ツールと比較して，回避耐性向上に有用であると考えられる．

補足 検知回避と解析回避の違い

本章における Sandbox 判定は，PoC が実行環境をサンドボックスとして識別し，それに応じて別途定義したペイロードを実行しなかったことを表す．ただし，ペイロードの非実行は，最終的な検知回避すなわち悪性判定の回避を直接意味するものではない．一方，マルウェア解析の実運用では，本来の悪性挙動が観測できないことは，機能把握や IoC 抽出の妨げとなり得る．したがって，本章の結果は主として，解析対象の挙動観測を阻害するという意味での解析回避に対する耐性の観点から解釈する．

6. 考　　察

本章では，本データセット構築の限界と解釈上の注意点を考察する．本研究は多数のサンドボックスの回避耐性評価に資する PoC 群の整備を目的とするが，STEP3 における採否判定や採用数は，一般化された「回避性能」を直接示すものではない．とくに，採否判定は実行に用いる実ユーザ環境およびサンドボックス群の性質に依存するため，本来有用な PoC が除外される可能性，あるいは不適切な PoC が残存する可能性がある．

また，本研究ではベース PoC の重複除去を行っていないため，同一の判定ロジックを持つ PoC が複数含まれ得る．さらに，多様化（亜種生成）は判定ロジックの保持を意図しているものの，書き換え前後で機能が同一であることは原則として実行による検証が必要となる．しかし，生成規模が大きく，全 PoC について実行検証を行えていない点は今後の課題とする．

一方で，取得経路（API や参照経路）の違いは，サンドボックス側のフックやエミュレーションの実装差により観測結果を分岐させ得る．先行研究 [13] でも，同一の回避技術であっても実装差により取得値や解析レポート上の観測が変化し得ることが報告されている．本研究はこの実装依存性を評価対象に含めるため PoC の多様化を多数含める設計としたが，今後は（i）判定ロジックや観測量に基づく重複除去，（ii）亜種の等価性を確認する回帰テストの整備を進める予定である．

7. 研 究 倫 理

本研究では，サンドボックスにおける解析回避に利用可能なコードを多数作成した．これらはサンドボックスの評価・改善に資する一方，不適切に利用されるおそれがある．そこで，ソースコード等の詳細は論文中では開示しない．研究目的が明確で身元確認可能な研究者に限り，適切な手続きに従って共有する．

8. まとめと今後の課題

本研究では，サンドボックスの回避耐性評価に用いる検体セットの不足という課題に対し，LLM を用いて回避技術を実装した PoC を自動生成し，さらに取得経路や API 選択の書き換えにより実装バリエーションを展開することで，大規模かつ多様な PoC 群を構築した．その結果，合計 4,508 件の PoC を生成し，実ユーザ環境およびサンドボックス群での実行結果に基づいて 1,520 件をデータセットとして採用した．（System Checks 1,151 件，Time Based Checks 189 件，User Activity Based Checks 180 件）．また，亜種をグループ化した回避技術の種類数は 487 となった

構築したデータセットを商用サンドボックスに適用したところ，合計 593 件で Sandbox 判定が確認され，既存ツールと比べて Sandbox 判定の割合が高かった．これは，本データセットがサンドボックスに残存する環境差異をより顕在化させやすく，弱点の抽出と改善対象の特定に有用な評価基盤となり得ることを示している．

今後は，多様化した PoC の機能的等価性検証を強化するとともに，データセット内の技術的偏りを定量的に分析する．加えて，回避技術の探索アプローチと検証環境を拡張し，収録対象となる回避技術の範囲を拡大する予定である．

謝辞　この成果の一部は，NEDO(国立研究開発法人新エネルギー・産業技術総合開発機構) 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．

本研究の一部は、JSPS 科研費 JP23K16879 の助成を受けて行われた．

文　　献

[1] Amir Afianian, Salman Niksefat, Babak Sadeghiyan, and David Baptiste. Malware dynamic analysis evasion techniques: A survey. *ACM Computing Surveys*, 52(6):1–28, 2019.

[2] Alexei Bulazel and Bülent Yener. A survey on automated dynamic malware analysis evasion and counter-evasion: Pc, mobile, and web. In *Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium (ROOTS '17)*, pages 1–21. ACM, 2017.

[3] Ayoub Faouzi. al-khaser. GitHub repository.

[4] Nicola Galloro, Mario Polino, Michele Carminati, Andrea Continella, and Stefano Zanero. A systematical and longitudinal study of evasive behaviors in windows malware. *Computers & Security*, 113:102550, 2022.

[5] Floris Gorter, Cristiano Giuffrida, and Erik van der Kouwe. Enviral: Fuzzing the environment for evasive malware analysis. In *16th European Workshop on System Security (EUROSEC '23)*, pages 1–7, Rome, Italy, May 2023. ACM.

[6] Songsong Liu, Pengbin Feng, Shu Wang, Kun Sun, and Jiahao Cao. Enhancing malware analysis sandboxes with emulated user behavior. *Computers & Security*, 115:102613, April 2022.

[7] Lorenzo Maffia, Dario Nisi, Platon Kotzias, Giovanni Lagorio, Simone Aonzo, and Davide Balzarotti. Longitudinal study of the prevalence of malware evasive techniques. *arXiv*, 2021.

[8] Najmeh Miramirkhani, Mahathi Priya Appini, Nick Nikiforakis, and Michalis Polychronakis. Spotless sandboxes: Evading malware analysis systems using wear-and-tear artifacts. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 1009–1024. IEEE, 2017.

[9] MITRE ATT&CK. Virtualization/sandbox evasion (t1497), October 2025. Last Modified: 24 October 2025.

[10] Alberto Ortega. Pafish. GitHub repository (testing tool for detecting VMs and malware analysis environments).

[11] Chenglin Xie, Yujie Guo, Shaosen Shi, Yu Sheng, Xiarun Chen, Chengyang Li, and Weiping Wen. Envfaker: Towards general, practical and secure imitation of target environments in malware analysis. In *2021 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 667–677. IEEE, 2021.

[12] Akira Yokoyama, Kou Ishii, Rui Tanabe, Yinmin Papa, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, Daisuke Inoue, Michael Brengel, Michael Backes, and Christian Rossow. Sandprint:

Fingerprinting malware sandboxes to provide intelligence for sandbox evasion. In *Research in Attacks, Intrusions, and Defenses (RAID 2016)*, volume 9854 of *Lecture Notes in Computer Science*, pages 165–187. Springer, 2016.

[13] 松澤 輝, 田辺 瑠偉, インミン パパ, and 吉岡 克成. Llm を用いて作成した解析回避検体がサンドボックス解析に与える影響の調査. In コンピュータセキュリティシンポジウム *2024* 論文集, pages 1148–1155, October 2024.