レガシーx86マルウェアがx86_64マルウェアとして蘇る可能性の分析

奥住 七海[1,a)]　栗原 北斗[1]　岡本 剛[1]

概要：x86_64 命令セットアーキテクチャの Windows マルウェアは，x86 命令セットアーキテクチャのマルウェアと比較して検知率が低い可能性が指摘されている．この背景を踏まえて，過去の x86 マルウェアを x86_64 向けにビルドすることでアンチウイルス（AV）ソフトの検知を回避する，新たな脅威「マルウェアの蘇り」について検証した．公開された過去の x86 マルウェアのソースコードを x86_64 向けにビルドし，主要な 8 製品の AV ソフトで検知率を評価した結果，検知可能な x86 マルウェアに対して x86_64 マルウェアの検知率は 60%に留まり，40%が検知を回避するリスクがあることを明らかにした．この要因として，命令セットアーキテクチャの違いにより既存のシグネチャが機能しなくなることや，AV ソフトの機械学習モデルが x86 用のデータに偏っている点が考えられる．後者に関しては，少量の x86_64 検体を学習データに加えるだけで検知率が大幅に改善する可能性を示した．本研究の知見と検体を AV ベンダーへ提供し，検知率が 61%から 86%に改善したことを確認した．

キーワード：マルウェアの蘇り，マルウェア検知，Windows マルウェア，アンチウイルス，機械学習

Analysis of the Potential of Resurrecting Legacy x86 Malware as x86_64 Malware

Nanami Okuzumi[1,a)]　Hokuto Kurihara[1]　Takeshi Okamoto[1]

Abstract: The detection rate for x86_64 Windows malware is potentially lower than that for its x86 malware. This paper investigates a new threat we term “Malware Resurrection,” where legacy x86 malware is rebuilt for the x86_64 architecture to evade antivirus (AV) detection. We rebuilt legacy x86 malware for the x86_64 architecture and evaluated their detection by eight major AV products. The detection rate for the x86_64 malware, rebuilt from the source code of their detectable x86 originals, was 60%, which allowed the remaining 40% of samples to evade detection. We attribute this decrease in detection to two main factors: ineffective signatures for the new architecture, and biased AV machine learning models. Notably, we found that augmenting the training data with even a small set of x86_64 samples significantly improved the detection rate. Sharing our findings and samples with AV vendors led to an increase in the detection rate from 61% to 86%.

Keywords: Malware resurrection, malware detection, Windows malware, antivirus, machine learning

1. はじめに

マルウェアは，現代のデジタル社会における最も深刻かつ持続的な脅威の一つである．特に Windows は，長年にわたり脅威アクターの主要な標的となってきた．2024 年の報告によれば，年間 9,180 万件を超える新種のマルウェアが確認され，そのうち 81%以上が Windows を標的としていた [1]．Windows を標的にしたマルウェアは，依然として x86 命令セットアーキテクチャ（以下，32 ビット）向けに開発されたものが主流であることが報告されている [2], [3]．

32 ビットマルウェアが依然として多く利用される理由と

[1] 神奈川工科大学
Kanagawa Institute of Technology
[a)] s2485009@cco.kanagawa-it.ac.jp

して，32 ビットマルウェアが x86_64 命令セットアーキテクチャ（以下，64 ビット）の Windows 上でも動作することや，32 ビットマルウェアの開発に長年の実績があることが挙げられる [2].

しかし，現在ではほとんどの Windows が 64 ビットのプロセッサーで動作するため，32 ビットプロセッサーの互換性を考慮する必要性がなくなった．この環境変化は，脅威アクターにとって新たな攻撃機会を生み出している．すなわち，既存の 32 ビットマルウェアのソースコードを流用し，僅かな労力で新たな 64 ビットマルウェアを生成できる可能性が考えられる．実際に，64 ビットマルウェアは緩やかな増加傾向にあり [2]，その割合は 2024 年時点で約 10%に達している [3]．ランサムウェア Conti が 32 ビットと 64 ビットの両方を標的とした事例 [4] や，Visual Studio 2022 では，新規プロジェクト作成時のターゲットのアーキテクチャが 64 ビットとなったため，この傾向が今後さらに加速すると考えられる．

64 ビットマルウェアの対策に関しては，64 ビット向けに開発されたハッキングツールやシェルコードが，既存の AV (Antivirus) ソフトによる検知を回避しやすいとの結果から [5], [6]，64 ビットマルウェアも同様に検知されにくい可能性が指摘されている [7]．AV ソフトの根幹をなすシグネチャベース検知は，過去に解析されたマルウェアのバイナリパターン（シグネチャ）との照合に依存する．しかし，同一のソースコードであっても，命令セットアーキテクチャが異なればバイナリは変化する．したがって，32 ビットマルウェアのソースコードを 64 ビット用にビルドした場合，バイナリパターンが変化することで AV ソフトが検知できなくなる可能性がある．このことから，過去に検知可能であった無数の「死んだ 32 ビットマルウェア」が，単純なビルドによって「検知不可能な 64 ビットマルウェア」として出現する新たな脅威が生まれる．本稿ではこの現象を「マルウェアの蘇り（Malware Resurrection）」と定義する．

本研究の目的は，「マルウェアの蘇り」が起こる可能性を実験によって検証し，その脅威の程度を定量的に明らかにすることである．具体的には，研究者向けに公開されているマルウェアのソースコードアーカイブから，約 20 年前から 6 年前にかけて開発された C/C++言語のプロジェクト約 400 個を分析対象として選定した．これらの 32 ビットマルウェアを 64 ビット向けにビルドする．ビルドに失敗する場合は，プログラムロジックが変わらない範囲で手動で修正を加える．また，ビルドに成功した検体が正常に動作するかをサンドボックスなどを用いて確認する．最後に，主要な 8 製品の AV ソフトを用いて，32 ビット検体と 64 ビット検体の検知率を評価し，「マルウェアの蘇り」の可能性を明らかにする．

関連研究として，ウェブサイトで公開されているマルウェアの脅威を分析した文献 [8] がある．この文献は，C/C++言語で記述された Windows マルウェアを対象にしている点で本研究と共通する．しかし，この文献は命令セットアーキテクチャを区別せずに検知率を分析しているのに対し，本研究は命令セットアーキテクチャを区別し，「マルウェアの蘇り」の可能性を分析している点が異なる．著者らの知る限り，このような「マルウェアの蘇り」の可能性を体系的に調査・実証した研究はこれまで存在しない．

本稿の貢献を以下に示す．

- 32 ビットマルウェアを 64 ビット向けにビルドして正常に動作させるにはソースコードを個別に修正することが不可欠であり，このため短期間に大量のマルウェアが蘇る可能性は低いことを明らかにした．
- 正常に動作したマルウェアの検知率は，32 ビットマルウェアが 67.26%，64 ビットマルウェアが 40.77%であり，64 ビットマルウェアの検知率が相対的にかなり低いことを明らかにした．
- 検知可能な 32 ビットマルウェアのソースコードからビルドした 64 ビットマルウェアの検知率は 59.73%にとどまり，40.27%の 64 ビットマルウェアが検知不可能なマルウェアとして蘇るリスクがあることを明らかにした．
- 蘇りの技術的な原因の 1 つは，AV ソフトの機械学習モデルが 32 ビットマルウェアのデータに偏っていることであり，少量の 64 ビットマルウェアのデータを加えるだけで，検知率は大幅に改善できることを示した．
- 本研究で使用した AV ソフトのベンダーへ蘇りの脅威を報告し，ビルドしたマルウェア検体を提供したことで，蘇ったマルウェアの検知率は 60.67%から 85.58%まで大幅に改善した．

2. 本研究の脅威シナリオと研究課題

本研究は，**図 1** に示すように，既に検知され「死を迎えた 32 ビットマルウェア」が，64 ビット向けにビルドされることで「検知不可能な 64 ビットマルウェア」として蘇る，すなわち「マルウェアの蘇り」の可能性を明らかにする．「マルウェアの蘇り」は脅威アクターがマルウェアのソースコードを保有していることを前提とする．この前提は，研究目的でマルウェアのソースコードを収集・公開する大規模なリポジトリが存在しているため，妥当であると考える．

マルウェアが蘇るには，AV ソフトの検知機能を回避する必要がある．AV ソフトの主要な検知機能として，シグネチャベース検知，ヒューリスティック検知，機械学習による検知がある．これらの検知機能がどのように回避されうるかを以下に述べる．

シグネチャベース検知は，過去に解析されたマルウェアのシグネチャを照合してマルウェアを検知する．32 ビット

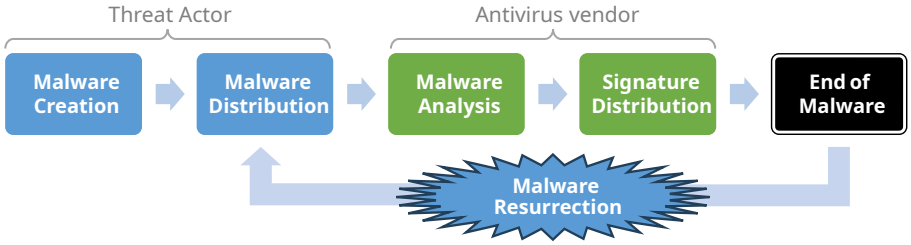

図 1 マルウェア蘇りの脅威シナリオ
Fig. 1 Malware resurrection scenario

から 64 ビット向けへのビルドは，オペコード列やデータ構造を大きく変化させる．これにより，シグネチャの対象コードが変化し，検知をすり抜ける恐れがある．

ヒューリスティック検知は，マルウェアに共通する特徴的なルールやパターンに基づき，プログラムのオペコード列やデータ構造，実行時の振る舞いなどを分析して疑わしい特徴を検出する．しかし，これらの検知ルールが 32 ビットの命令セットアーキテクチャを前提としている場合，64 ビットマルウェアのオペコード列やデータ構造に対しては有効性が低いと考えられる．

機械学習による検知は，大量のマルウェアと良性のプログラムから特徴を学習し，マルウェアか否かを識別する．しかし，1 章で述べた通り，収集されたマルウェアの多くは 32 ビットマルウェアであり，64 ビットマルウェアは極めて少ない．そのため，64 ビットマルウェアの特徴を十分に捉えきれておらず，64 ビットマルウェアを検知できない可能性がある．

以上の技術的考察に基づき，本研究では「マルウェアの蘇り」の可能性を明らかにするため，以下の研究課題（Research Question：RQ）を設定する．

- **RQ1**：32 ビットマルウェアのソースコードを 64 ビット用のマルウェアとしてビルドできるか
- **RQ2**：ビルドしたマルウェアは正常に動作するか
- **RQ3**：正常に動作したマルウェアは AV ソフトによって検知されるか
- **RQ4**：検知可能な 32 ビットマルウェアのソースコードからビルドした 64 ビットマルウェアはどの程度検知されるか

3. マルウェアの蘇りに関する実証実験

3.1 データセット

本研究では，研究者向けに公開されているリポジトリからマルウェアのソースコードを収集した．「マルウェアの蘇り」が特定の年代に限られた脅威ではなく，普遍的な脅威であることを示すため，Windows 95 や Windows XP といったレガシーな Windows を標的とするものから比較的近年に開発されたものまで，開発時期の異なるマルウェアを分析対象とした．分析対象のマルウェアは少なくとも 6 年以上前から存在し，長期間にわたり公開されていることから，既知のマルウェアとして分析が可能である．

これらのリポジトリには Visual C++，Visual Basic，Delphi，C#など多様な言語や開発環境で記述されたプロジェクトが含まれていたが，本研究では最も数が多い Visual C++で開発された約 400 個のプロジェクトを選定した．プロジェクトには C2 サーバと接続するボットが多く含まれているが，ファイル共有ソフトや，USB メモリを経由して拡散するマルウェア，情報窃取型マルウェアなども含まれている．また，ファイルが破損や欠損しているソースコードもあり，ビルドできることは保証されていない．

次に，選定したプロジェクトを精査した．バインダーやクリプターなど必ずしもマルウェアとは言えないプログラム，C2 サーバ，ドライバーなどのプロジェクト，64 ビット向けの実装を前提としたマルウェアを除外し，最終的に 183 個のプロジェクトを対象にビルドを行う．

20 年前のプロジェクトのほとんどは，Visual C++ 6.0 で作成されている．Visual C++ 6.0 は 64 ビットのビルドに対応していないため，本研究では Visual Studio 2019 を用いた．Visual C++ 6.0 と Visual Studio 2019 のプロジェクトファイルには互換性がないため，DSP2JCXPROJ ツール [9] を用いて旧形式から新形式へ変換した．さらに，32 ビットプラットフォームの構成をコピーして 64 ビットプラットフォームの構成を作成した．

倫理的な配慮から，本稿では使用したリポジトリの所在を明示しない．詳細は 5 章で後述する．

3.2 RQ1：検体のビルドと技術的要件の分析

本節では，RQ1 の「32 ビットマルウェアのソースコードを 64 ビット用のマルウェアとしてビルドできるか」を検証するため，ビルドを行い，その成功率およびビルドプロセスにおける技術的課題を分析する．

マルウェアのビルドは，Visual Studio 2019 を搭載した Windows 10 の仮想マシンで実施した．まず，ソースコードやプロジェクトファイルに一切の修正を加えない状態でのビルドを試行したところ，183 個のプロジェクト中，ビルドに成功したのは 7 個のみであった．

次に，ビルド成功の要件を特定するため，大量のコンパ

イルエラーが発生したプロジェクトを除いた 124 個を対象に，プログラムロジックが変更されない範囲で，プロジェクトファイルおよびソースコードの修正を試みた．本研究では，「プログラムロジックの変更」を，関数呼び出しの順序変更や条件分岐の追加・削除といった制御フローの変更と定義し，これに該当しない変数のデータ型の変更や設定値の追加，プリプロセッサによる分岐の修正のみを許容した．ビルド失敗の主な要因を以下に示す．エラーを解決する方法は倫理的な配慮から本稿では記載しない．

- **x86_64 用のインラインアセンブラの未対応：**Visual Studio は 64 ビット用のインラインアセンブラをサポートしない．そのため，`__asm` や `asm` キーワードで記述されたアセンブリ命令に対してコンパイルエラーが発生する．
- **C++言語仕様の変更：**Visual Studio .NET 2005 から `for` 文における変数スコープの設定オプションが変更され，Visual Studio .NET 2003 以前で開発されたコードでは，未宣言識別子のエラーが発生する．
- **Windows API の非互換性：**`_WIN32_WINNT` マクロの値が古いことが原因で Windows 10 で使用可能な API が定義されていない．その結果，多数の関数未定義のエラーが発生する．
- **ヘッダ・ライブラリの変更：**winnable.h など廃止されたヘッダーファイルの指定や，ライブラリ名やライブラリファイルパスの変更が原因で関数の未定義エラーや未解決の外部シンボル参照エラーが発生する．

これらを修正した結果，124 個のプロジェクトのうち 62 個（50%）において，32 ビットおよび 64 ビット双方のビルドに成功した．なお，修正作業に時間をかければビルド可能なプロジェクトはさらに増加すると考えられる．

この結果は，RQ1 に対して二つの重要な示唆を与える．第一に，32 ビットマルウェアから 64 ビットマルウェアを生成するには，インラインアセンブラの扱いや旧開発環境，さらには Windows OS の仕組みや Windows API の変更履歴に関する深い知識が不可欠である．これらの専門的知見が要求されるということは，「マルウェアの蘇り」がスキルレベルの低い脅威アクターによって安易に引き起こされるものではなく，特定の技術力を持つ脅威アクターによる脅威であることを示している．

第二に，ソースコードの個別修正が必須であったという結果は，既存の 32 ビットマルウェア群が，自動化された手法などによって短期間で大規模に 64 ビットマルウェアとして蘇る可能性が低いことを示している．

3.3 RQ2：ビルドした検体の動作検証

32 ビットマルウェアのソースコードを 64 ビット向けにビルドできたとしても，C2 サーバの設定不備やアーキテクチャの差異に起因する実行時エラーが動作を妨げる可能性がある．本節では，RQ2 の「ビルドしたマルウェアは正常に動作するか」を検証するため，前節でビルドした 62 検体を対象に動作検証を行い，その成功率と，動作を阻害する技術的要因を特定する．

3.3.1 動作検証の方法

検体の動作検証は，Windows 10 の仮想マシンで実施した．解析の効率化のため，Windows の UAC（User Account Control）と AV ソフトを無効にした．安全のため，インターネットから隔離して検体が C2（Command and Control）サーバから司令を受け取れるようにするため，仮想の LAN 内に偽の C2 サーバと DNS サーバを設置した．また，一部の検体は特定のパスワード入力を動作のトリガーとしていたため，偽の認証情報を用意した．各検体の実行前に仮想マシンのスナップショットを保存し，動作確認が終了した後にはスナップショットから実行前の状態に戻すことで，テスト環境の一貫性を確保した．

本研究では，以下のいずれかの条件を満たした場合に，検体が意図通りに動作したと判断した．

- C2 サーバからの指令を待ち受ける状態に至った場合
- 情報窃取など悪意のある振る舞いが確認できた場合

3.3.2 動作検証の結果

前節でビルドした 62 検体のうち，初期状態で正常に動作した検体は 2 検体のみであった．ほとんどの検体が正常に動作しなかったため，検体ごとに動作しない原因を調査し，プログラムロジックを変更しない範囲でソースコードを修正した．

動作を阻害した主要な原因を以下に示す．阻害する原因を解決する方法は倫理的な配慮から本稿では記載しない．

- **C2 サーバ設定の不備：**ソースコード内に C2 サーバのドメイン名が設定されておらず，接続に失敗する．
- **メモリアライメント違反：**64 ビット環境において，`#pragma pack(1)` によって 1 バイトにパックされた構造体メンバへのアクセスがメモリアクセス違反を誘発する．
- **ポインタの切り捨て（Pointer Truncation）：**64 ビット長のポインタ値を 32 ビット長の変数型に代入する際に上位 32 ビットが失われ，不正なメモリアドレスを参照する．

これらを修正した結果，62 検体のうち 42 検体において，32 ビットおよび 64 ビットの両方で正常な動作を確認した．残りの 20 検体は，動作させるためにプログラムロジックの変更が必要であったため，本研究の対象から除外した．

この結果から，32 ビットマルウェアのソースコードを単純に 64 ビット向けにビルドした場合，正常に動作しない可能性が高いことが示された．C2 サーバの設定に起因するエラーは比較的容易に修正可能である一方，メモリアライメントやポインタの切り捨てといった問題は，原因の特定と修正に時間を要することがわかった．本検証が示すよ

うに単純なビルドではそのほとんどが正常に動作しないため，大量の 32 ビットマルウェアが容易に 64 ビットの脅威となる可能性は低い．しかし，本研究で特定した修正は，いずれもアーキテクチャの差異に関する既知の問題であり，これらの知識を持つ開発者にとっては定型的な作業の範囲内である．したがって，マルウェア開発者にとって，手元にあるマルウェアのソースコードを動作可能な状態にビルドすることは，決して困難ではないと考えられる．

3.4 RQ3：検知率の評価

本節では，RQ3 の「正常に動作したマルウェアは AV ソフトによって検知されるか」を検証するため，前節で正常に動作することを確認した 42 検体を対象に，AV ソフトによる検知率を評価する．

3.4.1 評価の方法

評価には，国内外の主要な 10 製品の AV ソフトについて，それらの利用規約を調査し，本研究での利用が可能と判断した 8 製品を選定した．各 AV ソフトが相互に干渉することを防ぐため，製品ごとに独立した Windows 10 の仮想マシンを用意した．

評価は，インターネット接続の有無（オンライン／オフライン）と，検査種別（ファイル検査／実行検査）の組み合わせで実施した．オンライン検査はインターネットに接続した状態で，オフライン検査はすべてのインターネットとの接続を切断した状態で実施する．AV ソフトはクラウド上のシグネチャ，サンドボックス，機械学習などを活用するため，一般的にオンライン検査はオフライン検査よりも高い検知性能を示す [10]．ファイル検査とは，検体を実行せずにそのバイナリデータを静的に検査する手法を指す．一方，実行検査はと，検体を実行したときの振る舞いを検査する手法である．本研究では，感染時における外部への攻撃を完全に排除するため，オンライン環境での「実行検査」は実施していない．なお，すべての検査は各 AV ソフトの標準設定を用いて行った．

3.4.2 AV ソフトごとの検知率の評価

42 検体を 8 製品の AV ソフトで検査した結果を**図 2** に示す．AV ベンダーと AV ソフトの信用を毀損しないように各製品の検知率は公表しない．検査の結果，オンラインファイル検査における 32 ビット検体の平均検知率は 67.26%（標準偏差 15.78%）であったのに対し，同一のソースコードからビルドされた 64 ビット検体の平均検知率は 40.77%（標準偏差 17.10%）へと低下した．いずれの標準偏差も大きいことから，AV ベンダー間で検知性能に大きな格差があることがわかる．これは，オフラインファイル検査とオフライン実行検査でも同様の傾向であった．また，各 AV ソフトの 32 ビット検体と 64 ビット検体の検知率を比較すると，32 ビット検体の検知率が高くても 64 ビット検体の検知率が非常に低い AV ソフトが存在した．

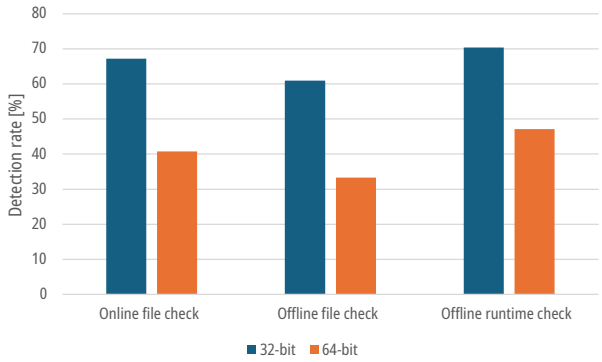

図 2 AV ソフトの平均検知率
Fig. 2 Average detection rates of AV software

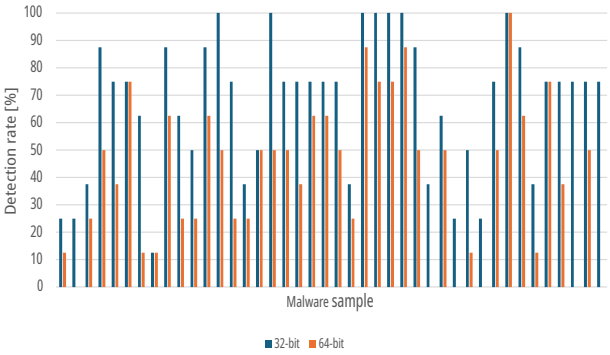

図 3 検体ごとの平均検知率
Fig. 3 Average detection rate per malware sample

3.4.3 検体ごとの検知率の評価

図 3 は横軸をマルウェアの各検体，縦軸を検知率 [%] として，検体ごとの検知率を示したものである．この図から，多くの検体において，同一のソースコードからビルドしたマルウェアでありながら 64 ビット検体の検知率が 32 ビット検体と比べて低下していることがわかる．さらに，32 ビット検体と 64 ビット検体の検知率における相関係数は 0.773 という強い正の相関を示した．これは，32 ビット検体で検知率が高かった検体は，64 ビット検体でも比較的検知されやすいことを示している．しかし，完全な一致ではないため，32 ビットマルウェアに対する検知性能の高さが，必ずしも 64 ビットマルウェアへの高い対応能力を直接保証するものではないことも示唆している．

これらの結果は，2014 年に報告された 64 ビット検体の検知率が低いという課題が，依然として業界全体で解決されていないことを示している．64 ビットマルウェアが増加傾向にある現状を鑑みれば，この弱点は看過できないリスクである．

3.4.4 32 ビットマルウェアの検知率が低い原因

本研究で用いたデータセットは，既知のマルウェアであるため 32 ビット検体は極めて高い確率で検知されると想定していた．しかし，実際の平均検知率は 67.26%と，想定を下回る結果となった．検知率が低い原因として，ビルド

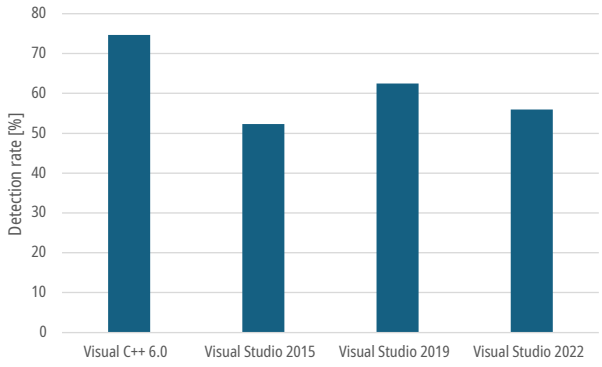

図 4 ツールセットごとの平均検知率
Fig. 4 Average detection rate per toolset

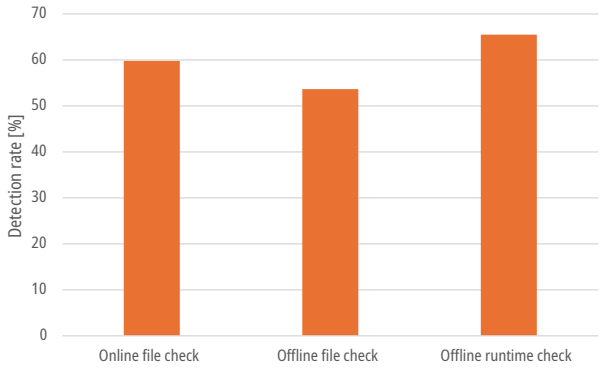

図 5 蘇り検知率
Fig. 5 Resurrection detection rate

に用いたツールセットが異なることを仮説として立てる．本研究で用いたツールセットがマルウェア開発当時と異なる場合，AV ソフトのシグネチャが検体のバイナリと一致せず，検知を回避する可能性がある．

この仮説を検証するため，Visual C++ 6.0 で開発された 20 年前の検体について，複数の異なるバージョンのツールセットでビルドし，オフラインファイル検査における検知率を比較した．**図 4** に示すように，Visual C++ 6.0 でビルドした 32 ビット検体の検知率は 72.43%を超えるのに対し，Visual Studio でビルドした検体の検知率は 60.29%を下回った．この結果は，ツールセットの差異が検知率に影響を与えることを示しており，当初の想定よりも検知率が低かった原因であることを確認できた．この他にも，AV ベンダーが検体を入手していない可能性や，オフライン検査という手法自体の制約も，検知率が伸び悩んだ要因として考えられる．

3.5 RQ4：マルウェアが蘇る可能性の分析

本節では，RQ4 の「検知可能な 32 ビットマルウェアのソースコードからビルドした 64 ビットマルウェアはどの程度検知されるか」を検証する．この「マルウェアの蘇り」の発生率を定量的に評価するため，前節の実験結果に基づき，以下の指標「蘇り検知率（Resurrection Detection Rate，R）」を定義し，算出した．

$$R = \frac{\sum_{i=1}^{8} N_{32\&64,i}}{\sum_{i=1}^{8} N_{32,i}}$$

ここで，$N_{32,i}$ は AV ソフト i が検知した 32 ビット検体の総数を示す．一方，$N_{32\&64,i}$ は，AV ソフト i が 32 ビット検体とそれに対応する 64 ビット検体の両方を検知した検体の総数を表す．この指標 R の値が低いほど，多くのマルウェアが検知を逃れ「蘇った」ことを意味する．

検知率の結果を**図 5** に示す．3 つの検査方法において，これらの平均検知率は 59.55%であった．これは，AV ソフトによって一度は検知可能とされた 32 ビットマルウェアのうち 40.45%が，検知不可能な 64 ビットマルウェアとして蘇る可能性を示唆している．

4. マルウェアの蘇りに関する原因分析

4.1 64 ビットマルウェアの検知を可能にした要因

3 章で示した通り，64 ビットマルウェアの検知率は 32 ビットマルウェアと比較して低下した．命令セットアーキテクチャの変更は，オペコード列を大きく変化させるため，当初の予想では，コード領域を対象とするシグネチャベース検知や静的ヒューリスティック検知はほぼ無力化され，検知率はさらに低くなると考えられた．しかし，オンラインファイル検査において 64 ビットマルウェアを 40.77%も検知できたという結果は，アーキテクチャの変更に影響されない，別の要因が検知に寄与した可能性を示唆している．

この要因として，複数の AV ソフトが，コード領域ではなくデータ領域に含まれる情報に基づいてマルウェアを検知している可能性が考えられる．具体的には，マルウェアが利用する特定の Windows API 名リストや，ハードコードされた C2 サーバのドメイン名といった文字列データが，アーキテクチャに依存しないシグネチャとして機能していたと推測される．

この仮説を裏付ける具体例として，ある AV ソフトが検知したマルウェアの検知名に特定の CVE 番号が含まれていたケースが挙げられる．この検体のソースコードを分析した結果，当該脆弱性を攻撃するためのエクスプロイトコードが，リテラル文字列として直接ハードコードされ，それが PE ファイルの `.rdata`（読み取り専用データ）セクションに含まれていることが確認できた．この文字列がシグネチャと一致し，検知につながった可能性が極めて高い．

PE ファイルの `.rdata` セクションや `.idata` セクション（インポート情報）などに含まれるデータは，その多くがアーキテクチャの変更に影響を受けない．したがって，これらのデータ領域から抽出されたシグネチャは，32 ビットと 64 ビットの両方のマルウェア亜種に対して有効性を維持できる．

以上の分析から，マルウェアの蘇りに対する一つの有効

な対策として，コード領域だけでなく，データ領域に含まれるアーキテクチャ非依存の静的データからシグネチャを抽出し，活用することが考えられる．ただし，この手法は，パッカーやクリプターによってデータ領域が難読化または暗号化された場合には，その有効性が著しく低下するという限界も存在する．

4.2 機械学習による検知の分析

40%のマルウェアが蘇ったという結果は，アーキテクチャの変更により既存の検知手法の有効性が低下するなど，より根本的な要因の存在を示唆している．本節ではこれらの要因の中でも特に機械学習に着目する．AV ソフトが学習するマルウェアデータが 32 ビットに偏っているという「学習データの偏り」を原因とする仮説を実験で検証し，今後のデータ収集戦略と防御性能向上のための具体的な指針を提示する．

4.2.1 検証に使用したデータセット

検証では，MWS Dataset の一部として提供される FFRI Dataset 2022 から 2024[11] までの 3 年分を使用した．各データセットには以下の 2 種類が含まれ，これらは PE ヘッダーの各種データやファイルに含まれる文字列情報など，主に静的解析で得られるデータから構成されている．

- **マルウェア：** 株式会社 FFRI セキュリティが収集したマルウェア 75,000 件（内，64 ビットマルウェアは約 3,000 件）
- **クリーンウェア：** AV-TEST 社の FLARE サービスから収集した良性のプログラム 75,000 件

4.2.2 検証の手順

本検証は，各試行において，データセット全体から学習用とテスト用をサンプリングした．テストデータはクリーンウェアと 64 ビットマルウェアを一対一の比率で構成した．学習データもクリーンウェアとマルウェアの比率を一対一に保ちつつ，そのマルウェア部分を 64 ビットマルウェアとそれ以外で構成し，64 ビットマルウェアが占める割合を 0%から 100%まで段階的に増加させた．各割合の学習データで機械学習モデル（ランダムフォレスト）を学習させ，テストデータで性能を評価した．評価指標には再現率（Recall）と正解率（Accuracy）を用いた．この一連の試行を各割合で 10 回繰り返し，その平均値を評価することで，サンプリングによる偶然性を排除し結果の信頼性を確保した．

4.2.3 検証結果と考察

学習データ中の 64 ビットマルウェアの割合と，テストデータに対する再現率と正解率の関係を**図 6** に示す．図中の実線は 10 回の試行における各指標の平均値を，帯状の領域（エラーバンド）は結果のばらつきを示す標準偏差を表している．また，両指標の変化が特に大きい 0%から 10%の区間は縮尺を拡大して表示している．なお，FFRI Dataset 2022 から 2024 までの各データセットで同様の検証を行ったが，結果にほとんど差異は見られなかったため，本稿では代表として FFRI Dataset 2024 を用いた結果を示す．

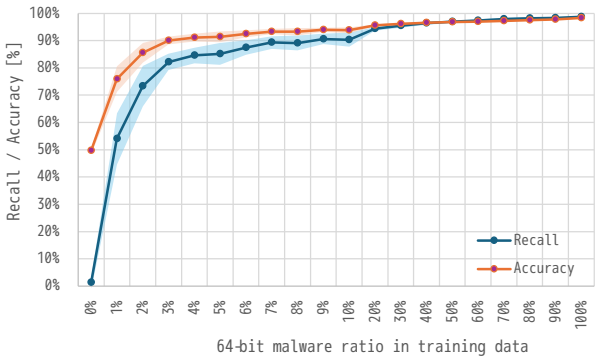

図 6 64 ビットマルウェアの割合と検知率の関係
Fig. 6 Relationship between 64-bit malware ratio in training data and detection rate

図 6 は，マルウェアの蘇りが，機械学習モデルの「学習データの偏り」に起因するという仮説を裏付けるものである．学習データに 64 ビットマルウェアを含まない場合，その検知率（再現率）は 1.3%と極めて低い値に留まった．これは，32 ビットと 64 ビットの命令セットアーキテクチャの違いにより，32 ビットマルウェアの学習で得られた知識が 64 ビットマルウェアの検知には適用が難しいことを示している．一方で，学習データに 64 ビットマルウェアをわずか 1%加えるだけで検知率は 54.0%へ，10%の追加で 90.3%へと急上昇した．この結果は，64 ビットマルウェアに特有のパターンを学習させることが，検知性能の向上に直結することを客観的に示している．

少量の検体を追加するだけで検知率が大幅に改善するという結果は，単にデータの量を増やすのではなく，その多様性を重視する戦略的なデータ収集が，極めて有効な対策であることを示している．例えば，ハニーポットや AV ソフトが動作するエンドポイントで検体をリアクティブに収集するだけでなく，本研究のように公開されたマルウェアのソースコードから 64 ビットマルウェアをビルドして検体をプロアクティブに収集することが効果的である．

さらに，検知率が約 99%で飽和する点は，残りの約 1%の検知には，単純なデータ量の追加とは異なるアプローチが必要であることを示唆している．その要因として，検体そのものの希少性により十分な学習ができていない可能性や，静的解析から抽出される現行の特徴量では悪性の特徴を捉えきれないタイプである可能性などが考えられる．したがって，マルウェアの蘇りへの包括的な対策としては，64 ビットの学習データを拡充するだけでなく，動的解析や振る舞い検知といった多層的な防御アプローチを組み合わせることが不可欠である．

5. 研究倫理

本研究で得られる知見は，未知の脅威に対する AV ソフトの検知ロジックの改善に貢献しうる高い公益性を持つ．その一方で我々は，特に「蘇り」の条件を明らかにすることが，悪意を持って利用されるリスクを伴うことも深く認識している．このようなリスクに対処するために，以下のような倫理的配慮と安全対策を行った．

まず，本研究で明らかにした「蘇り」の技術が悪用されるリスクを低減するため，論文公開に先立ち，AV ベンダー各社と連携した責任ある情報開示（Responsible Disclosure）を実施した．具体的には，各社への情報提供から本論文の公開まで，対策のための期間として 90 日以上を設けた．その上で，各社のマルウェア検体提出窓口等を通じて脅威の概要と対策の方向性を情報提供し，実験において検知できなかった全てのマルウェア検体を提出するとともに，AV ベンダーからの要請に応じてソースコードも提供可能であることを伝えた．その結果，全体の検知率は 60.67%から 85.58%へと改善した．ただし，この検知率は本稿の投稿直前に提供した 8 検体を含んでいない．なお，90 日という短い期間で検知ロジックを改良することは難しいと考えられるので，この検知率の改善は主にシグネチャの追加によるものと考えられる．

また，AV ソフトの検知率評価においては，特定の AV ベンダーへの信用を毀損しないよう，製品名や個別の検知率を公表しない方針とした．評価対象を 8 社の製品とすることで，特定の製品が標的となるリスクも低減している．さらに，マルウェアのソースコード提供元サイト運営者への倫理的配慮から，その具体的な URL などの情報は本論文に記載していない．

法令の遵守に関して，マルウェアのソースコードおよびビルド済みの検体は，著者らが管理する安全な環境下でのみ保管し，不特定の第三者に対して提供，公開，または売買することは一切行っていない．研究の実施にあたっては，第三章で述べた隔離環境に加え，誤操作による実行を防止するためにファイルの拡張子を変更する等の厳格な安全管理策を講じた．また，本稿では，研究の目的，方法，評価，および結果に関して，悪用されないように必要な範囲で説明することに努めた．

以上の通り，本研究は「マルウェアの蘇り」という機微なテーマを扱いながらも，その公益性を追求すると同時に，成果の悪用や実験に伴うリスクを徹底して管理・低減するための方策を講じた上で実施された．

6. おわりに

本研究では，検知可能な 32 ビットマルウェアが検知不可能な 64 ビットマルウェアとして蘇る可能性を明らかにするために，公開された 32 ビットマルウェアのソースコードを 64 ビット向けにビルドし，正常な動作を確認し，主要な 8 製品の AV ソフトで検知率を評価した．実験の結果，検知可能な 32 ビットマルウェアの 40%が，検知不可能な 64 ビットマルウェアとして蘇る可能性を明らかにした．この要因として，命令セットアーキテクチャの違いにより既存のシグネチャが機能しなくなることや，AV ソフトの機械学習モデルが 32 ビットマルウェアに偏っている点が考えられる．後者に関しては，MWS Dataset を使って少量の 64 ビット検体を学習データに加えるだけで検知率が大幅に改善する可能性を示した．本研究の知見と評価に用いた検体を AV ベンダーへ提供した結果，検知率が 61%から 86%へと大幅に改善した．

今後の課題として，Go 言語や Rust 言語で開発された新しいマルウェアや，AArch64 命令セットアーキテクチャにおける蘇りの可能性についても脅威の検証を進める．

謝辞　本研究は JSPS 科研費 24K14954 の助成を受けたものである．

参考文献

[1] AV-TEST: AV-ATLAS - Malware & PUA, https://portal.av-atlas.org/malware (2024).
[2] Mertens, X.: 32 or 64 bits Malware?, https://isc.sans.edu/diary/32+or+64+bits+Malware%3F/28968 (2022).
[3] Wilson, S.: Malware Trends: Yearly 2023, https://blog.unpac.me/2024/01/30/malware-trends-yearly/ (2024).
[4] Forescout: Anlaysis of Conti leaks, https://www.forescout.com/resources/analysis-of-conti-leaks/ (2022).
[5] Swinnen, A. and Mesbahi, A.: One packer to rule them all: Empirical identification, comparison and circumvention of current antivirus detection techniques, *BlackHat USA* (2014).
[6] NirSoft: Amazing difference between Antivirus false alerts on 32-bit and 64-bit builds of exactly the same tool, https://blog.nirsoft.net/2012/10/10/amazing-difference-between-antivirus-false-alerts-on-32-bit-and-64-bit-builds-of-exactly-the-same-tool/ (2012).
[7] Deep Instinct: Beware of the 64-bit Malware (2017).
[8] La Cholter, W., Elder, M. and Stalick, A.: Windows Malware Binaries in C/C++ GitHub Repositories: Prevalence and Lessons Learned., *ICISSP*, pp. 475–484 (2021).
[9] Qashat, N.: DSPtoVCXPROJ, https://github.com/CyberBotX/DSPtoVCXPROJ (2023).
[10] AV-Comparatives: Malware Protection Test September 2024, https://www.av-comparatives.org/tests/malware-protection-test-september-2024/ (2024).
[11] 株式会社 FFRI セキュリティ：FFRI Dataset 2024. https://www.iwsec.org/mws/2024/files/MWS2024_dataset_FFRI.pdf.