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

ChatGPTを用いたソースコードの書き換えが アンチウイルスの検知に与える影響

黄　哲偉† 　インミンパパ†† 　吉岡　克成†††,†† 　松本　勉†††,††

† 横浜国立大学大学院環境情報学府
†† 横浜国立大学先端科学高等研究院
††† 横浜国立大学環境情報研究院
E-mail: †kou-tetsui-ry@ynu.jp, ††{yinminn-papa-jp,yoshioka,tsutomu}@ynu.ac.jp

あらまし　大規模言語モデル (LLM) の急速な社会実装にともない，サイバー攻撃への悪用が懸念されている．実際に，大規模言語モデルによってマルウェアのソースコードが生成できることは既に知られている．また，アンチウイルスの検知を回避するマルウェア亜種を作成する方法として，従来からのパッカーによる難読化やダミーコードの挿入等の方法に加えて, LLM を利用したマルウェアのソースコードの書き換えが考えられる．しかし，LLM が生成したマルウェア亜種がアンチウイルスの検知結果に与える影響については十分に検討されているとは言えない．本研究では，典型的な悪意のある挙動を備える 4 つのマルウェアテスト検体のソースコードを ChatGPT を用いて書き換えることで検体の亜種を複数生成し，これらの亜種検体がアンチウイルス製品の検知へ与える影響について調査を行った．

キーワード　ChatGPT，マルウェア，亜種，アンチウイルス

Impact of ChatGPT Assisted Polymorphic Malware on Antivirus Detection

Zhewei HUANG†, Yin Minn Pa Pa††, Katsunari YOSHIOKA†††,††, and Tsutomu MATSUMOTO†††,††

† Graduate School of Environment and Information Sciences, Yokohama National University
†† Institute of Advanced Sciences, Yokohama National University
††† Faculty of Environment and Information Sciences, Yokohama National University
E-mail: †kou-tetsui-ry@ynu.jp, ††{yinminn-papa-jp,yoshioka,tsutomu}@ynu.ac.jp

Abstract　With the rapid implementation of large-scale language models (LLMs), there are concerns about their misuse in cyberattacks. Previous studies have shown that large-scale language models can generate malware source code. In addition, although obfuscation by packers and insertion of dummy code were used to create malware variants in the past to evade malware detection, with the spread of LLMs, generating malware variants by rewriting the malware source code with LLMs can be a new threat. However, the impact of LLM-generated malware variants on malware detection has not been fully studied. In this study, we generate multiple malware variants of four test malware samples by rewriting their source code using ChatGPT, and investigate if these samples can be detected by anti-virus products.

Key words　ChatGPT, malware, variants, anti-virus

1. はじめに

近年の計算機の発展と深層学習技術の進展によって大規模言語モデルの社会実装が急速に進んだことにより，大規模言語モデルの悪用がサイバーセキュリティにおける新たな脅威になっている．特に，多くのテキスト生成モデルは事前学習のデータセットにプログラミング言語から構成されるデータを含めていることから，自然言語からプログラムのソースコードを生成でき，マルウェア開発への悪用が懸念されている．大規模言語モデルの提供者は悪用への対策として，社会実装にあたり倫理に反する内容の出力が低減するように出力のフィルタリングやモデルの微調整による制御を行っているが，実際には特殊な入力を与えることでマルウェアソースコードを生成できることが知られている [1], [2]．また，攻撃者の観点で，大規模言語モデルを利用したマルウェアソースコードの書き換えによるマルウェア亜種開発が考えられる．しかし，この

方法によって開発されるマルウェア亜種に対して，アンチウイルスに代表される既存の防御技術が対応できるかは十分に検証されていない．本研究では，大規模言語モデルによるマルウェア亜種開発が既存の防御技術であるアンチウイルス製品の検知機能へ与える影響に着目する．

本研究では以下の研究課題 (RQ) を設定した．

RQ1) ChatGPT を利用したソースコードの書き換えによってマルウェア亜種は開発可能か

RQ2) ChatGPT を利用して作成したマルウェア亜種がアンチウイルスの検知へどのような影響を与えるか

上記の RQ を明らかにするために，OpenAI が提供する大規模言語モデルに基づくサービスである ChatGPT を利用し，4 種類のマルウェアテスト検体のソースコードを入力として与えることで，複数の亜種検体の開発を試行した．また，作成した亜種検体に対して既存のウイルス対策製品の検知機能への影響を調査するために 6 つの製品について検体実行前のファイルスキャンによる検知機能と検体実行時の振る舞い監視による検知機能に対して調査を行った．アンチウイルス製品は侵入検知等複数の機能を有しているが，本研究においては，マルウェアの侵入を許した際のスキャンおよび振る舞いによる検知，除去の機能を対象とする．また，本研究におけるマルウェア亜種とは，元のマルウェアと同一の機能を有すると共にソースコードの実装が異なる検体のことを指す．

本研究における貢献は以下の通りである．

- 4 種類のテスト検体に対して，ChatGPT を利用したソースコードの書き換えによって生成した各 100 サンプルのソースコードのうち，5 割程度が実行ファイルへ変換可能，かつテスト検体と同じ機能を有することを確認した．
- 6 つのアンチウイルスソフトについて，ソースコードの書き換えによって作成された亜種検体を用いて調査を実施し，一部の検体がアンチウイルスの検知を回避することを確認した．アンチウイルスについて，既知のマルウェアの亜種を検知する能力が十分でない可能性を示した．

2. 関 連 研 究

大規模言語モデルを利用したマルウェア開発に関して, Botacin の研究 [3] と Yin Minn Pa Pa らの研究 [1] がある．Botacin は OpenAI が提供する text-davinci-003 を利用してマルウェアの文字列の XOR エンコード，デバッガー検知，ファイル削除，ペイロードのロード，CPUID のチェック，メモリ上での実行，レジストリへの登録，Base64 のデコード，DLL インジェクションのスニペットを作成し，それらを組み合わせてマルウェアを開発できることを示している．また，スニペットの実装をモデルに変更させることでマルウェア亜種を開発し，VirusTotal の検査の結果，ウイルス対策製品の検知を回避する検体が作成できることを確認している．Yin Minn Pa Pa らは OpenAI が提供する ChatGPT の Web サービス，および Open AI API の gpt-3.5-turbo に基づくサービスが，ワーム，キーロガー，ランサムウェア，ファイルレスマルウェア等のマルウェアを含む悪意のあるソフトウェアの開発へ利用できることを示している．

これらの研究では，自然言語のみを与えることによるマルウェアおよびマルウェア亜種の開発を扱っているのに対し，本研究では LLM に自然言語とマルウェアソースコードを与えることによる亜種検体の開発を扱う．

既存のウイルス対策製品の検知機能を評価する研究に Botacin らの研究 [4] がある．その他に，MITRE 社による EDR 評価 [5] や，アンチウイルスの評価を専門とする独立機関である AV-TEST [6]，AV Comparatives [7] による評価レポートがある．

既存のアンチウイルス評価の研究では，大規模言語モデルを用いて開発したマルウェア亜種がアンチウイルスへ及ぼす影響について議論していない．そこで本研究では，4 つのテスト検体について ChatGPT を利用したソースコードの書き換えによって亜種検体を複数作成し，それらがアンチウイルスの検知へ及ぼす影響について着目する．

3. 調 査 手 法

調査の流れを図 1 に示す．RQ1) について，まず，4 種類のテスト検体 (以降，元検体) を用意し，それぞれの検体のソースコードを ChatGPT へ与え，検体と同様の機能をもちプログラムとしての実装が異なる亜種検体のソースコード 100 サンプルを得る．これらのソースコードを実行ファイルに変換し，元の検体と同じ機能を有するか後述する方法で確認する．元の検体と同じ機能を持つ亜種検体について，ソースコードおよび実行バイナリについて元検体との類似度を調査する．次に，RQ2) について，各種類の亜種検体を 2 つのグループへ分割し，期間を変えて，アンチウイルスをインストールした仮想環境上での実行，およびオンラインのファイル検査サービスである VirusTotal [8] へアップロードを行い，これらの亜種検体がアンチウイルスの検知へ与える影響に関して調査を実施した．

以降では，ChatGPT を利用したソースコードの書き換えによる亜種検体の作成，および亜種検体がアンチウイルスの検知へ与える影響の調査の詳細について述べる．

3.1 ChatGPT を利用したソースコードの書き換えによる亜種検体の作成

まず，調査に利用する亜種検体作成のために以下の 4 種類のテスト検体を用意した．

ワーム：network.txt ファイルに記載されている内部ネットワークの 23 番ポートに対してスキャンを行い，telnet サービスが稼働している端末を探索する．telnet サービスに対してログインを実行する．ログイン先において，ローカルホストの HTTP サーバからシェルコマンドを取得して実行する．

ランサムウェア：Windows OS のユーザプロファイル以下に存在する特定のフォルダにあるファイルを，AES アルゴリズムによって暗号化し，元のファイルの削除を行う．

キーロガー：WindowsAPI によるキーボードフッキングを利用してユーザのキーボード入力を窃取し，ローカルホストの HTTP サーバへ送信する．

ファイルレスマルウェア：WindowsAPI を用いて正規のプロセスに対してメモリを確保し，メッセージボックスを表示するシェルコードを挿入して実行する．

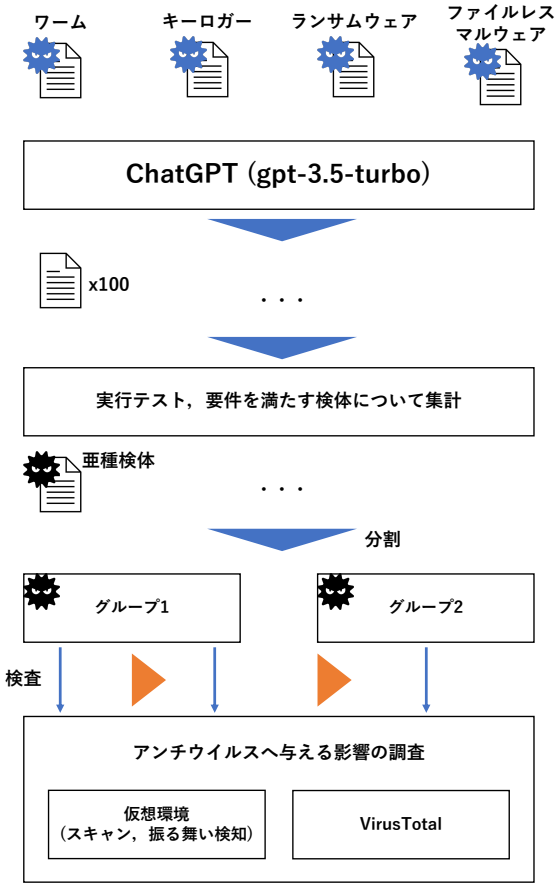


図 1: 調査の全体像

上記検体の対象とする動作環境は Windows とし，実際のマルウェアによく見られる典型的な機能を含めた．亜種検体作成の際に言語ごとに違いが現れることを期待し，複数の言語でテスト検体を実装した．各検体を実装する言語について，ワームは Python，ランサムウェアおよびキーロガーは Go，ファイルレスマルウェアについては C++ で実装した．以降，これらの検体を元検体と呼ぶ．作成した元検体について，ウイルス対策製品によって検知されることを確認するために VirusTotal で検査を実施した．

次に，OpenAI API [9] が提供するモデルの gpt-3.5-turbo を API 経由で利用し，亜種検体を作成する．本研究における亜種とは，元検体のソースコードと実装が異なるが，元検体と同一の機能を有する検体のことを指す．モデルのプロンプトに，亜種検体のソースコードを作成する指示と各元検体のソースコードを与え，各元検体について 100 サンプル，合計 400 のソースコードを生成した．

得られたソースコードについて，Python は PyInstaller [10]，Go と C++ はコンパイルによって実行ファイルへ変換し，元検体の機能を有しているか確認した．機能の確認について，ワームは telnet サーバにおけるシェルコマンドの実行結果，ランサムウェアは暗号化されたファイルの存在，キーロガーは HTTP サーバへのログの送信を確認した．ファイルレスマルウェアは各検体を手動で実行し，検体実行後にメッセージボックスが表示されていることをもって元検体の機能を有していると した．

また，作成した亜種検体について，元検体に対してどの程度異なる実装がされたかについて確認するために，ソースコードの観点から，機械翻訳において翻訳文と参照訳の類似度の評価に利用される BLEU スコア [11]，およびソースコードに含まれるトークン数の増減と行数の増減で評価する．なお，ソースコードの類似度の評価では空行とコメントは除いている．また，バイトコードの観点から，ファジーハッシュの ssdeep [12]，および局所性鋭敏型ハッシュの TLSH [13] を用いて元検体との類似度を計算した．各評価に利用した指標の詳細については結果の節で述べる．

3.2 アンチウイルス製品へ与える影響の調査

以下に示す手順でアンチウイルス製品への影響の調査を行った．調査にあたり，まず，前節で得た亜種検体をグループ 1 とグループ 2 に分割した．分割は，各種類がそれぞれ 20 検体ずつ含まれるようにし，各グループに合計 80 検体が含まれる．亜種検体を 2 つのグループに分割したのは，完全に未知のテスト検体に対するアンチウイルスの検知と，既知のテスト検体の亜種検体に対するアンチウイルスの検知について調査を行うためである．

（1） Windows 仮想環境上にアンチウイルスソフトをインストールし，グループ 1 について，1 日ごとに検体に対するスキャンと検体の実行を行い，1 つのエンドポイントで未知検体が出現した際の対応について 7 日間観測．

（2） VirusTotal に検体グループ 1 をアップロードし，検知するアンチウイルスエンジンの数を記録．未知検体が確認されるエンドポイントを増やすことを目的とする．

（3） アンチウイルスソフトをインストールした仮想環境上において，検体グループ 1 について，検体のスキャンと実行を行い，検体が確認されるエンドポイントが増えた場合のアンチウイルスの対応について 7 日間観測．

（4） アンチウイルスソフトをインストールした仮想環境上において，検体グループ 2 のについて，検体のスキャンと実行を行い，新たな亜種検体が登場した際のアンチウイルスの対応について 7 日間観測．

（5） VirusTotal に検体グループ 2 をアップロードし，検知するアンチウイルスエンジンの数を記録．

各手順の実施期間について，VirusTotal へのサンプルアップロードの影響を排除するため，(2) は (1) の終了後に実施した．VirusTotal へのアップロードにより，検体グループ 1 がアンチウイルスの定義データベースに登録された際のアンチウイルスの検知を調査するために，(3) は (1) の終了から 92 日後に実施した．グループ 2 の出現に対するアンチウイルスの検知を調査するため，(4) は (3) の 1 日後に実施した．VirusTotal へのアップロードにより，検体が確認されるエンドポイントが増えた場合の影響について調査するために，(5) は (4) の 4 日後に実施している．仮想環境におけるアンチウイルスの調査について，定義データベースの更新によるアンチウイルスの検知の変化について調査するために 7 日間継続して観測を行っ

表 1: 元検体の VirusTotal における検知数

検体名	検知した AV 製品の数
ワーム	38 / 72
ランサムウェア	6 / 72
キーロガー	12 / 72
ファイルレスマルウェア	7 / 72

表 2: 機能を満たす検体数

検体名	元検体と同じ機能を満たす検体数
ワーム	58 / 100
ランサムウェア	51 / 100
キーロガー	46 / 100
ファイルレスマルウェア	56 / 100

た．検知機能の調査では，テスト検体に対してスキャンによる検知を実施した後，スキャンで検知されなかった検体を実行し振る舞い検知の結果を記録した．また，仮想環境において調査対象とした 6 つのアンチウイルスソフトのマルウェア実行前と実行時の検知機能を表 4 に示す．6 製品ともスキャンによる検知機能を有しており，Anti-Virus3 と Anti-Virus5 は振る舞い監視機能を持たない．調査対象としたアンチウイルスソフトは，知名度とシェアの観点から選定を行った．実際の製品名に関しては，研究倫理の観点から公開を控える．

3.3 実 験 環 境

6 つのアンチウイルスソフト製品について，それぞれ Windows 仮想環境を準備し，合計 6 つの仮想環境で実験を実施した．Windows 仮想環境は Windows 11 Enterprise Evaluation を利用し，ウイルス対策製品の更新を受け取るためにインターネットへ接続している．仮想環境にホストマシンとの共有フォルダを設定することでテスト検体を環境に導入した．

4. 調 査 結 果

4.1 亜種検体の作成について

元検体の検知状況： 元検体について，ウイルス対策製品によって検知されることを確認するために VirusTotal で検査を実施し検査結果を表 1 に示す．なお，VirusTotal における Windows を対象とする検査エンジンの総数は 72 である．

ChatGPT による亜種検体の生成： 亜種検体について、得られたソースコードのうち，元検体と同じ機能を有していた検体の数を表 2 に示す．いずれの種類のテスト検体について亜種検体作成の成功率は 5 割程度であることを確認した．実行ファイルに変換できなかったソースコードについては，文法のエラーに加え，ソースコードの生成が途切れる，開発環境に導入されていないライブラリの利用，ハードコーディングされた定数の書き換えがなされていた．

元検体と亜種検体の類似度： 各亜種検体の元検体に対する BLUE スコア [11], トークン数と行数の増減, および ssdeep [12], TLSH [13] による類似度の平均と標準偏差を表 3 に示す．BLEU は $[0, 1]$ の範囲の値をとる．1 に近いほど元検体のソースコードとの類似度が高い．Token，Line は元検体のソースコードに対するトークン数，行数の増減である．ここで，トークンは正規表現 $\backslash b[a-zA-Z0-9\_]+\backslash b$ に一致する文字列としている．$\backslash b$ は単語境界である．ssdeep による類似度は $[0, 100]$ の範囲の値をとり，100 に近いほど類似しているバイナリである．TLSH による類似度は非負の値をとり，0 に近いほど類似しているバイナリであることを表す．

4.2 アンチウイルスの検知へ与える影響

4.2.1 検体グループ 1 のオンプレミス環境における結果 (1)

検体グループ 1 に対する仮想環境におけるアンチウイルス検知の 1 度目の結果を図 2 に示す．図において，wrm，klg，rsm，fls はそれぞれ，ワーム，キーロガー，ランサムウェア，ファイルレスマルウェアを指す．ワーム検体について，6 つのアンチウイルス中，3 製品 (AV1，AV3，AV4) がスキャンで検知した．このうち，2 製品 (AV1,AV4) が 6 日目と 7 日目には検知しなくなっている．キーロガー検体について，6 製品すべてで検知されなかった．ランサムウェア検体について，6 製品ともスキャンで検知しなかったが，3 製品 (AV2，AV4，AV6) が振る舞いによって全検体を検知した．ファイルレスマルウェア検体について，2 製品 (AV1，AV5) がスキャンで検知した．AV5 は 7 日間継続して全検体をスキャンで検知した．AV1 が 2 検体を検知したが，6 日目には検知しなくなった．

各検体の検知の傾向として，ワームはシグネチャによって不審なファイルとして検知される場合があるが，一定の期間が経過すると検知されなくなる．ランサムウェアは振る舞いによって対処される．ファイルレスマルウェアはシグネチャによって対処しているベンダが存在することを確認した．

4.2.2 検体グループ 1 のオンプレミス環境における結果 (2)

検体グループ 1 に対する仮想環境におけるアンチウイルス検知の 2 度目の結果を図 3 に示す．2 日目の検査後から検体グループ 2 の検査，5 日目の検査後から VirusTotal における検体グループ 2 の検査が実施されているワーム検体について，AV2 を除く 5 製品がスキャンによって検知した．キーロガー検体について，AV1 を除く 5 製品がスキャンによって検知した．また，AV4 について，スキャンで検知されない検体が振る舞いによって検知されることを確認した．ランサムウェアについて，AV5 が 6 日目からスキャンによって 8 検体を検知した．振る舞い検知の結果については 1 度目の調査結果から変化が見られなかった．ファイルレスマルウェア検体について，新たに AV3 が 2 検体を検知するようになった．

1 度目の調査と比較した際のアンチウイルスの検知の傾向として，ワーム，キーロガーについて，スキャンで検知される検体が増加した．これは，検体グループ 1 を VirusTotal へアップロードしたことで，製品の定義データベースにテスト検体に関するシグネチャが追加されたためであると考えられる．

4.2.3 検体グループ 2 のオンプレミス環境における結果

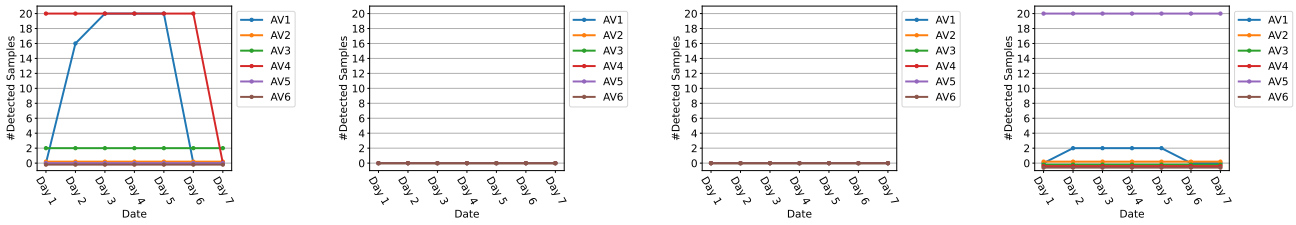
検体グループ 2 に対する仮想環境におけるアンチウイルス検知の結果を図 4 に示す．開始から 4 日目の観測後，VirusTotal へ検体グループ 2 をアップロードしている．ワーム検体について，4 日目に AV6 が振る舞いで検知している．5 日目から

表 3: 亜種検体の元検体に対する類似度 [平均 ± 標準偏差]

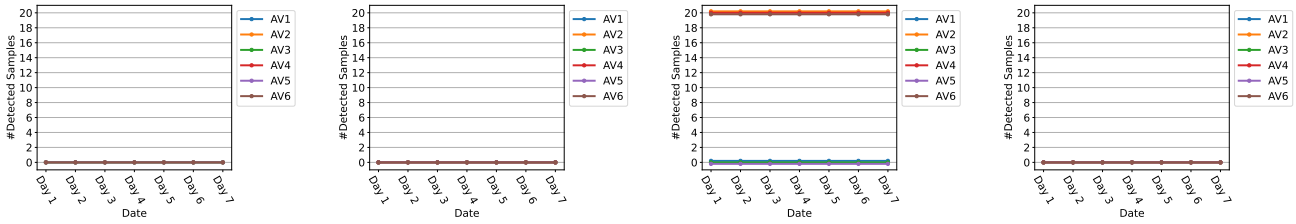
検体名	Text			Binary	
	BLEU	Token	Line	ssdeep	TLSH
ワーム	0.83 ± 0.08	-2.7 ± 3.3	-1.4 ± 4.7	96 ± 0.0	1.5 ± 0.95
キーロガー	0.96 ± 0.06	-2.6 ± 5.2	-3.0 ± 5.0	4.3 ± 16	8.1 ± 4.2
ランサムウェア	0.87 ± 0.05	-1.8 ± 3.9	0.15 ± 4.2	11 ± 23	12 ± 5.1
ファイルレスマルウェア	0.59 ± 0.09	-19 ± 9.6	-10 ± 7.0	0.0 ± 0.0	93 ± 55

表 4: 対象とするアンチウイルス製品の検知機能

	Detection before Malware Runtime			Detection during Malware Runtime	
	Manual Scan	Scheduled Scan	Real Time Scan	Real Time Scan	Behavior Monitoring
Anti-Virus 1	o	o	o	-	o
Anti-Virus 2	o	o	o	-	o
Anti-Virus 3	o	o	o	o	-
Anti-Virus 4	o	o	o	-	o
Anti-Virus 5	o	o	o	o	-
Anti-Virus 6	o	o	o	o	o



(a) wrm: スキャンによる検知　(b) klg: スキャンによる検知　(c) rsm: スキャンによる検知　(d) fls: スキャンによる検知



(e) wrm: 振る舞い検知　(f) klg: 振る舞い検知　(g) rsm: 振る舞い検知　(h) fls: 振る舞い検知

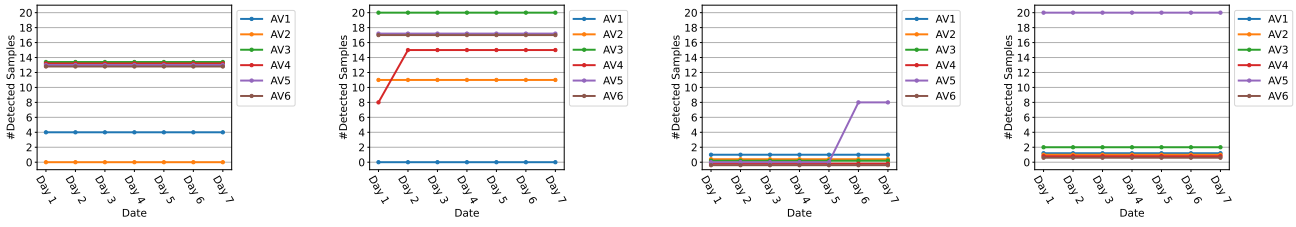
図 2: アンチウイルスが検知したグループ 1 の検体数の推移 [1 度目]

AV3 が，7 日目から AV6 がいくつかの検体をスキャンで検知するようになっている．キーロガー検体について，2 製品 (AV2，AV4) がいくつかの検体を 1 日目からスキャンによって検知した．また，すべての製品について，5 日目以降にスキャンで検知する検体数が増加した．2 製品 (AV4，AV6) が 4 日目に振る舞いによってキーロガー検体を検知した．5 日目以降，キーロガー検体を検知した 2 製品について，スキャンによる検知増加したため，振る舞いで検知される検体の数は減少している．ランサムウェア検体について，4 日目までスキャンによって検知する製品は AV1 を除き，存在しなかったが，5 日目以降は AV2 を除き，5 製品でスキャンによって検知される検体の数が増加した．振る舞い検知に関して，3 製品 (AV2，AV4，AV6) が 1 日目から検知した．6 日目以降，先に実施されるスキャンによる検知で検体が削除されたため，3 製品が検知する検体数が減少している．ファイルレスマルウェア検体に関して，AV5 のみが 1 日目から全検体をスキャンによって検知した．また，AV4 が 3 日目と 4 日目に振る舞いによって全検体を検知し，5 日目以降は振る舞いで検知しなくなった．

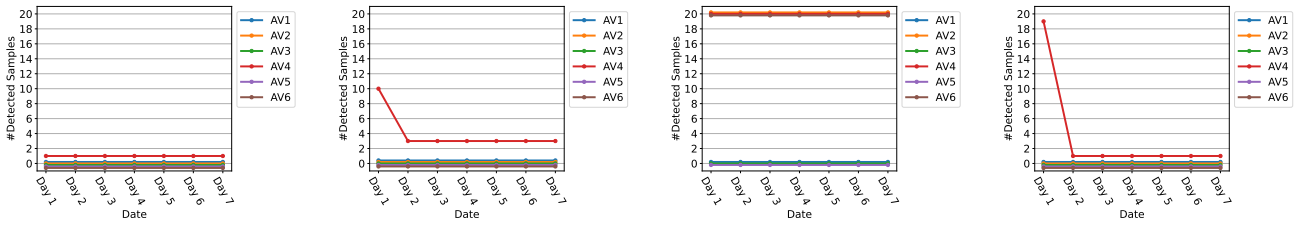
アンチウイルスの検知の傾向として，検体グループ 2 の VirusTotal へのアップロード実施後の 5 日目から，キーロガー検体とランサムウェア検体の検知数が増加した．これは，VirusTotal でインテリジェンスを収集した各製品ベンダが定義ファイルを更新したためであると考えられる．ワーム検体については不審なファイルとして検出された後，検知される検体の数が減少した．

4.2.4 VirusTotal における検査

検体グループ 1 と検体グループ 2 のの VirusTotal における検査結果を図 5 に示す．図 5 は横軸が VirusuTotal において検知

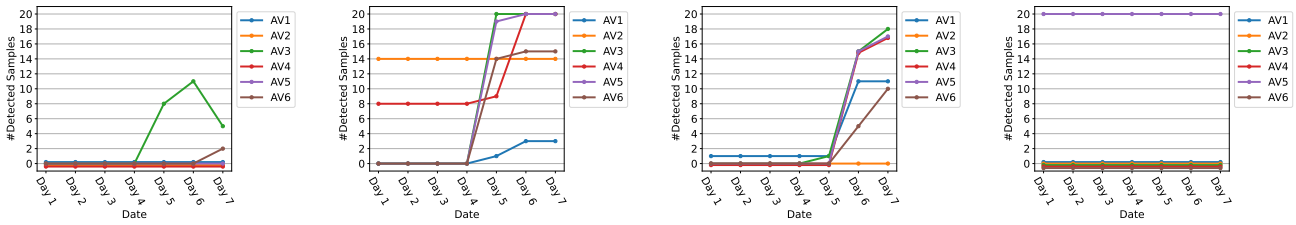


(a) wrm: スキャンによる検知　(b) klg: スキャンによる検知　(c) rsm: スキャンによる検知　(d) fls: スキャンによる検知

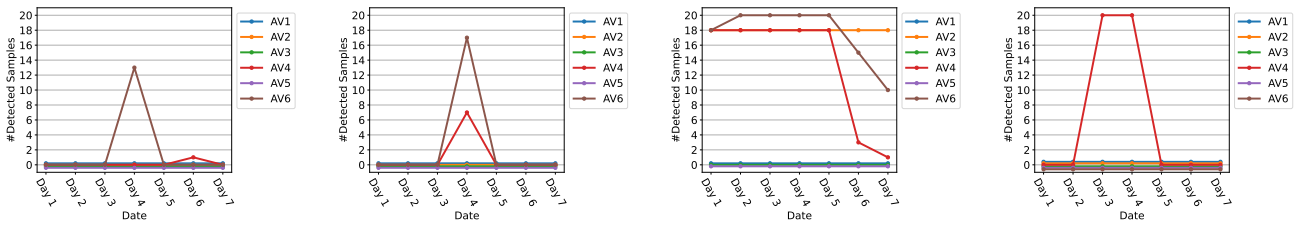


(e) wrm: 振る舞い検知　(f) klg: 振る舞い検知　(g) rsm: 振る舞い検知　(h) fls: 振る舞い検知

図 3: アンチウイルスが検知したグループ 1 の検体の数の推移 [2 度目]



(a) wrm: スキャンによる検知　(b) klg: スキャンによる検知　(c) rsm: スキャンによる検知　(d) fls: スキャンによる検知



(e) wrm: 振る舞い検知　(f) klg: 振る舞い検知　(g) rsm: 振る舞い検知　(h) fls: 振る舞い検知

図 4: アンチウイルスが検知したグループ 2 の検体の数の推移

する製品の数の階級，縦軸が検体の数のヒストグラムである．図におけるラベル [*groupN*]-[*label*]-[*n*] はグループ N における *label* 検体に対する検査結果を指す．ここで，複数回検査を実施したグループ 1 について，異なる期間の検査結果を検査の実施順に $n = 1, 2$ としてラベル付けしている．$n = 1$ は仮想環境における 1 度目の調査後，VirusTotal へアップロードした際の結果である．$n = 2$ はグループ 1 を用いた 2 度目の調査の開始時の検査結果である．グループ 1 の 1 つ目の検査結果について，いずれの検体 wrm，klg，rsm，fls についても検知した製品の数は 0 から 10 の範囲である．グループ 1 の 2 つ目の検査結果について，ワーム検体 (group1-wrm-2) は半数以上の検体を検知する製品数が 35 から 40 に増加した．キーロガー，ランサムウェア，ファイルレスマルウェア検体に関して，半数以上の検体について検知する製品数が増加している．グループ 2 の検査結果について，いずれの検体についても検体グループ 1 の 1 度目の結果に対して，検知する製品数が増加していた．

5. 考　　察

5.1 RQ1) ChatGPT を用いたマルウェア亜種開発の可能性について

各元検体のソースコードのコメントと空行を除く行数について，ワームが 51，キーロガーが 59，ランサムウェアが 96，ファイルレスマルウェアが 84 である．本研究で利用したテスト検体のソースコード程度の規模において，ChatGPT による書き換えによって 5 割程度のソースコードが元検体の機能を満たしていることから，100 行以下で記述されるマルウェアやモジュールについては，大規模言語モデルによって実装を変えることが可能であることを示唆している．

また，亜種検体の元検体に対する類似の度合いについて，ワーム検体のバイナリの類似度が高い．これは，ワーム検体

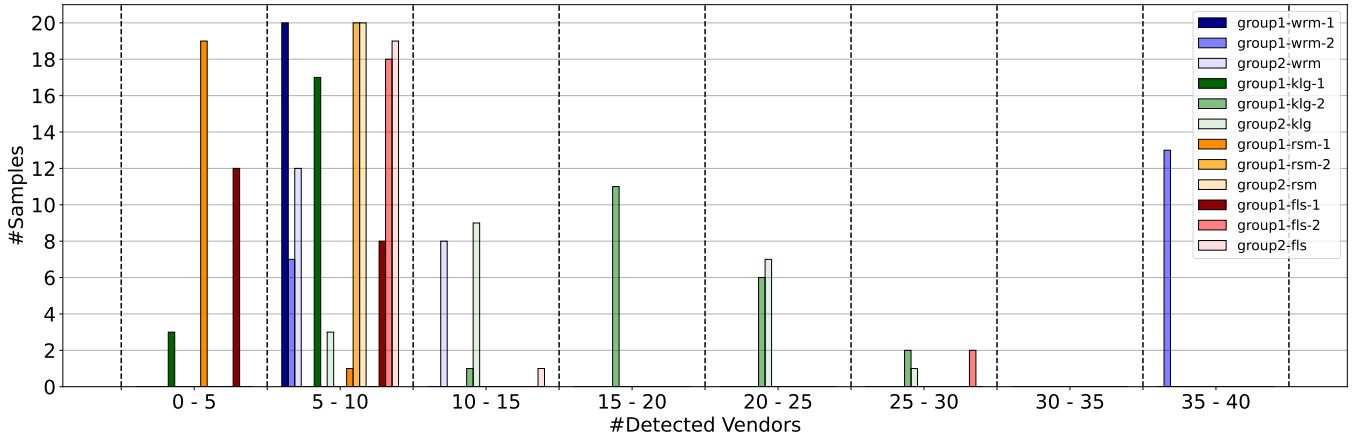


図 5: 各亜種検体の VirusTotal における検査結果の分布

が同一のライブラリを利用していることから，ソースコードを実行ファイルに変換する際に PyInstaller がライブラリに関連するバイナリを 1 つの実行ファイルにまとめた結果，バイナリの類似度が高くなったと考えられる．

5.2 RQ2) アンチウイルスの検知機能へ与える影響について

本研究では，ChatGPT によるソースコードの書き換えによって，典型的なマルウェアの振る舞いを有する元検体について，複数の亜種検体を開発し，それらを用いてアンチウイルスの検知機能について調査を行った．アンチウイルスをインストールしたオンプレミスの仮想環境におけるテスト検体の実行の結果より，ランサムウェアに特有の暗号化と削除の振る舞いは検知する製品が存在するのに対し，ネットワークスキャンによる情報の収集の振る舞いを検知する製品は少なかった．また，キーボードフッキングの振る舞いを有する検体について，1 つのエンドポイントではスキャン，振る舞いによって検知しないが，複数のエンドポイントで確認された際にシグネチャとして登録されている可能性がある．また，プロセスインジェクションの振る舞いについて，シグネチャや振る舞いで検知する製品が存在するが，多くの製品では検知されないことを確認した．これは，アンチウイルスがプロセスインジェクションの振る舞いを悪意のあるものとして対処しておらず，インジェクションされたプロセスの振る舞いによって悪意の有無を判断するためであると考えられる．

VirusTotal におけるグループ 1 とグループ 2 の検査結果，および仮想環境におけるスキャンの結果より，VirusTotal がアンチウイルスのインテリジェンス収集元として利用されていることが示唆される．また，仮想環境におけるランサムウェア検体に対する AV2，AV4，AV6 の検知結果から，未知検体を振る舞い検知機能によって検知した場合にも，当該検体のシグネチャを必ずしも登録するわけではないことがわかった．当該検体を検知した件数やエンドポイントの分布など多様な観点でシグネチャ登録を決定している可能性があるが製品ベンダの未知検体への対応やシグネチャ追加の詳細については更なる分析が必要である．

ChatGPT を用いたソースコードの書き換えによるマルウェア亜種作成の脅威について，キーロガーの検知に関して着目すると，検体グループ 1 を用いた仮想環境における調査結果より，すべての検体が検知されるようになるわけではないことがわかった．

5.3 研究倫理に関する考察

本研究では大規模言語モデルのサイバー攻撃への悪用の可能性について検証を行った．大規模言語モデルのサイバー攻撃への悪用については多くの検討がなされているが，亜種作成の可能性と現行のセキュリティ対策技術の有効性への影響の分析は不十分であり，これらを正確に把握することはサイバー攻撃の高度化への適切な対応に資するものであると考え，論文として報告する次第である．研究成果の悪用を防ぐため，実験に使用したプロンプトは論文内では開示しないが，実験に関する詳細情報を希望する研究者との情報共有は適切な手続きに従い行う予定である．アンチウイルス製品については一般的な製品への影響を示すことを目的とし，特定製品への言及は避けることとした．

6. 結　　論

本研究では，ChatGPT にテスト検体のソースコードを与え，書き換えを行うことで，マルウェアテスト検体の亜種を開発し，それらに対するアンチウイルスの検知について調査を行った．その結果，亜種検体の作成に関して，各種類の検体について 100 サンプル生成した亜種検体のソースコードのうち 5 割程度のサンプルが実行ファイルに変換でき，かつ元検体と同じ機能を有していた．また，亜種検体について，ファイルレスマルウェアはソースコード，バイナリともに元検体との違いが大きい結果となった．これらの亜種検体がアンチウイルスの検知へ与える影響について調査を行った結果，アンチウイルスの未知検体および既知検体への対処の傾向を観測した．

今後の展望として，他の LLM，プロンプトを用いた亜種検体の作成を試行し，生成される亜種検体の傾向の違いに関する調査，本研究で対象としなかったアンチウイルスの検知に関する調査，および実際のマルウェアソースコードを利用した調査によって，LLM 悪用とその対策に関する検討を進めたい．

謝辞：本研究の一部は日本学術振興会日中韓フォーサイト事業 JPJSA3F20200001，JSPS 科研費 21H03444，および JSPS 科研費 21KK0178 の助成を受けて行われた．

文　　献

[1] Y.M. Pa Pa, S. Tanizaki, T. Kou, M. Van Eeten, K. Yoshioka, and T. Matsumoto, “An attacker’s dream? exploring the capabilities of chat-gpt for developing malware,” Proceedings of the 16th Cyber Security Experimentation and Test Workshop, pp.10–18, 2023.

[2] A. Mulgrew, “I built a Zero Day virus with undetectable exfiltration using only ChatGPT prompts”. https://www.forcepoint.com/blog/x-labs/zero-day-exfiltration-using-chatgpt-prompts

[3] M. Botacin, “Gpthreats-3: Is automatic malware generation a threat?,” 2023 IEEE Security and Privacy Workshops (SPW)IEEE, pp.238–254 2023.

[4] M. Botacin, F. Ceschin, P. De Geus, and A. Grégio, “We need to talk about antiviruses: challenges & pitfalls of av evaluations,” Computers & Security, vol.95, p.101859, 2020.

[5] MITRE ENGENUITY, “ATT&CK Evaluations”. https://attackevals.mitre-engenuity.org

[6] AV-TEST, “Test Modules under Windows”. https://www.av-test.org/en/about-the-institute/test-procedures/test-modules-under-windows-protection/

[7] AV Comparatives, “Malware Protection Test September 2023”. https://www.av-comparatives.org/tests/malware-protection-test-september-2023/

[8] VirusTotal. https://www.virustotal.com

[9] OpenAI, “OpenAI API”. https://openai.com/blog/openai-api

[10] PyInstaller. https://pyinstaller.org/

[11] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pp.311–318, 2002.

[12] J. Kornblum, “Identifying almost identical files using context triggered piecewise hashing,” Digital investigation, vol.3, pp.91–97, 2006.

[13] J. Oliver, C. Cheng, and Y. Chen, “Tlsh–a locality sensitive hash,” 2013 Fourth Cybercrime and Trustworthy Computing WorkshopIEEE, pp.7–13 2013.