6ZC-05

ChatGPT-4を使用したマルウェア作成に関する考察

A Study on Implemantation of Malwares using ChatGPT-4

伊藤 穂来人 †　杉尾 信行 †

北海道科学大学 †

1　はじめに

2023 年 3 月に登場した ChatGPT-4 は，人工知能技術の飛躍的な進歩を象徴している．従来のモデルを大きく超える理解力と応答能力を持ち，様々な質問に対して精度の高い回答を提供する事が可能となった．今では多岐にわたる分野で多くのユーザに利用されており，その可能性は底知れない．しかし， この技術の進展には負の側面も存在する．その 1 つとして挙げられるのが，ChatGPT-4 の高度な生成能力が，悪意のあるソフトウェアの作成に悪用される事例である．そこで本研究では，コーディング作業を極力せずに，ChatGPT-4 によるマルウェアの作成を試み，仮想環境で実行させる事で，熟練した技術者でなくとも，高度なマルウェアを作成可能か検証した. また．作成したマルウェアのソースコードを難読化する事で，セキュリティ分析ツールによるマルウェア検出率を低減させられるか検証した.

2　関連研究

本来，ChatGPT を運営する OpenAI の倫理規定により，マルウェアの作成を，直接または間接的に示唆する要求は拒否される．しかし実際には，ChatGPT-4 を使用してマルウェアの作成に成功した研究は複数報告されている [1][2][3][4]．それらの研究で用いられた手法は，(1)JailBreak を利用して作成する方法と，(2)JailBreak を利用せず，正規の使用法のみで作成する方法，との 2 つに分類できる．(2) では各機能ごとにソースコードを生成させ，最後にそれらを全て結合する事で，セーフガードを破らずにマルウェアの完成を実現していた.

3　関連技術

3.1　ChatGPT-4

ChatGPT-4 は，OpenAI によって開発された最新の人工知能言語モデルである．自然言語処理において顕著な進歩を遂げており，高度な文章生成と理解能力を持っている.

3.2　ステガノグラフィ

ステガノグラフィは情報を隠蔽する技術である．メッセージやデータを，画像・音声ファイル，その他のメディアに隠す為に使用される．データを隠蔽する事により，第三者に存在を気付かれずに送信する事が出来る.

3.3　静的及び動的解析

静的解析では，ソフトウェアを実行せずにコードを分析するのに対し，動的解析では，ソフトウェアを実際にサンドボックス上で実行して挙動を観察する．Hybrid Analysis は，この静的解析と動的解析の両方を行う事が可能で，マルウェアの包括的な特性を把握する為に使用される [5].

3.4　アイコン偽装

アイコン偽装は，マルウェアを正規のソフトウェアに見せかける為に使用される技術である．アイコンウィザードで作成したアイコンファイルを，Resource Hacker で設定する事で，マルウェアの外観を変え，発見を困難にする事ができる [6][7].

3.5　python コード難読化

難読化は，ソースコードの可読性を下げ，読みにくく改変する事である．著作権の保護やリバースエンジニアリングの防止の為に行われる．PyObfuscator や Pyarmor などの難読化ツールを使用する事で python コードを難読化する事が出来る [8][9].

4　提案手法

本研究では，先行研究の内，OpenAI による修正に依存しない (2) の手法に倣い，以下の 4 種類のマルウェアを Python コードで開発した.

a ファイル **hash** 化マルウェア
windows に標準で用意されている，デスクトップ・ダウンロード・ドキュメント・ピクチャのフォルダ内の全てのファイルを hash 化し，元データを削除する.

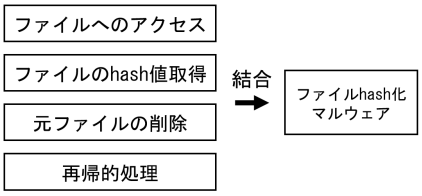

図 1: ファイル hash 化マルウェアの構成図

† Hokuto Ito and Nobuyuki Sugio, Hokkaido University of Science

b キーログ転送マルウェア

キーボード入力の内容をはじめ，アクティブウィンドウ，PC のホスト名，IP アドレス，接続しているネットワーク情報を記録し，攻撃者へ転送する．

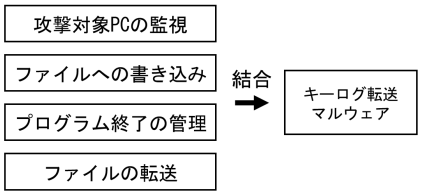

図 2: キーログ転送マルウェアの構成図

c 画像ステガノグラフィプログラム

png 画像に，先に紹介した a，b どちらか一方のマルウェアを埋め込み，マルウェアが隠蔽された画像を生成する．

図 3: 元画像

図 4: 実行後の画像

d 抽出・自動実行マルウェア

ステガノグラフィによりマルウェアが埋め込まれた画像から，マルウェアを抽出する．同時に，対象者のホームディレクトリに sample フォルダを作成し，抽出されたマルウェアはそのフォルダ内に配置される．次回の PC 起動以降，抽出されたマルウェアを自動で実行させる．

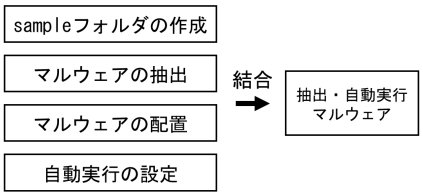

図 5: 抽出・自動実行マルウェアの構成図

まず攻撃者は，(d) 画像ステガノグラフィプログラムを実行する．次に (c) 抽出・自動実行マルウェアに，pdf ファイルを装ったアイコン偽装を施し，実行を急かす様なファイル名を付けておく．最後に，マルウェアが埋め込まれた画像と，アイコン偽装された (c) 抽出・自動実行マルウェアを対象者へ渡す．対象者がそれらを受け取った後，マルウェア抽出プログラムを pdf ファイルと見誤って実行すると，次回の PC 起動以降，毎度自動で (a または b) マルウェアが実行される．

5 評価と考察

作成したマルウェアを，Virtual Box 上に建てた windows11 で実行した際，正常に動作する事が確認できた．これらを Hybrid Analysis で分析すると，対象者の PC で実行される exe ファイル形式のマルウェアが，全て危険性が高く，脅威であると判定された．一方で，攻撃者の PC で実行する python ファイル形式のマルウェアは，マルウェアとして検出されず，安全なファイルであると判定された．これより，セキュリティ分析ツールがスクリプトファイルよりも実行可能ファイルに対してより厳しい検出基準を適用するのではないかと考えられる．また，商用ソフトウェアに付与されるデジタル署名の不足も不審とみなされる一因かもしれない．脅威であると判定されたマルウェアについて，(1)PyObfuscator による難読化，(2)PyArmor による難読化をそれぞれ試みた．その結果，(1) の方法では，検出率が変化せず，(2) の方法では，僅かながら検出率の減少を実現する事が出来た．

6 まとめと今後の課題

本研究では，コーディング作業を極力しなくとも，ChatGPT-4 を使用して，マルウェアを作成できる事を明らかにした．一方で，マルウェア分析の結果から，高度なマルウェアとは言えず，PyObfuscator 及び PyArmor による難読化も，劇的な改善が見られなかった．実際に攻撃に使用する為には，これらのツールの複数回の実行や併用を含め，より高度な難読化により，セキュリティ分析ツールを完全に回避する必要がある．

参考文献

[1] Aaron Mulgrew. I built a Zero Day virus with undetectable exfiltration using only ChatGPT prompts. Forcepoint. https://www.forcepoint.com/blog/x-labs/zero-day-exfiltration-using-chatgpt-prompts (2023)

[2] Erik Derner, Kristina Batistic. Beyond the Safeguards Exploring the Security Risks of ChatGPT. Czech Technical University in Prague. https://doi.org/10.48550/arXiv.2305.08005 (2023)

[3] Sayak Saha Roy, Krishna Vamsi Naragam, Shirin Nilizadeh. Generating Phishing Attacks using ChatGPT. The University of Texas at Arlington. https://doi.org/10.48550/arXiv.2305.05133 (2023)

[4] Maanak Gupta, Charan Kumar Akiri et al. From ChatGPT to ThreatGPT：Impact of Generative AI in Cybersecurity and Privacy. IEEE Access. https://doi.org/10.48550/arXiv.2307.00691 (2023)

[5] Hybrid Analysis. https://www.hybrid-analysis.com/

[6] アイコンウィザード. https://www.vector.co.jp/soft/winnt/amuse/se430751.html

[7] Resource acker. ttps://resource-hacker.softonic.jp/

[8] PyObfuscator. https://pypi.org/project/PyObfuscator/

[9] Pyarmor. https://pypi.org/project/pyarmor/