
ARTIFICIAL INTELLIGENCE IN THE CYBERSECURITY FIELD

DOI 10.20535/2411-1031.2023.11.2.293778

УДК 004(8+05)

ВОЛОДИМИР СОКОЛОВ

АНАЛІЗ МОЖЛИВОСТЕЙ CHATGPT ЩОДО РОЗВ'ЯЗКУ ЗАДАЧ РЕВЕРС-ІНЖИНІРИНГУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В статті представлено результати аналізу та оцінки можливостей відкритої версії чат-боту зі штучним інтелектом ChatGPT 3.5 щодо розв'язку типових задач реверс-інжинірингу програмного забезпечення. Для аналізу було обрано три класи задач реверс-інжинірингу: аналіз вихідного коду, аналіз бінарного коду та аналіз моделей даних. В кожному класі задач було обрано найбільш типові задачі з урахуванням обмежень ChatGPT щодо обробки графічних моделей та обсягу вхідних та вихідних даних, а також розроблено набори тестових завдань по кожній задачі. В якості підходу до оцінювання обрано підхід, аналогічний оцінюванню компетентностей здобувачів вищої освіти після вивчення відповідної дисципліни. В якості критеріїв оцінювання відповідей по тестовим завданням розглядалися наступні: коректність (правильність, збіг з очікуваннями); повнота (отримання кінцевого результату); точність (розв'язок завдання без додаткових питань); обґрунтованість (наявність пояснень та відповіді на питання). Для оцінок використано та описано наступну шкалу: відмінно, дуже добре, добре, задовільно, достатньо, незадовільно. Під час проведення тестування по кожному тестовому завданню через інтерфейс ChatGPT вводилися завдання та всі необхідні дані. В результаті проведеного аналізу визначено, що ChatGPT краще розв'язує задачі аналізу вихідного коду (відмінні та дуже добрі оцінки отримано за семантичний та структурний аналіз, відновлення математичного забезпечення, оцінку якості, аудит безпеки та рефакторинг, а також за конвертацію в іншу мову програмування), задачі декомпіляції псевдокоду IDA в закінчений вихідний код мовою C, задачі реверс-інжинірингу реляційних баз даних та задачі генерації правил YARA для розпізнавання форматів файлів. Незадовільні оцінки були отримані за динамічний аналіз асемблерного коду та визначення структур бінарних файлів нестандартних форматів. Решту задач ChatGPT розв'язує добре або задовільно та потребує перевірки отриманих результатів, уточнюючих запитів та підказок, а також ручного виправлення помилок в деяких випадках. Було виявлено помилки ChatGPT під час аналізу бінарних даних, представлених символами в шістнадцятковій системі числення, а також помилки в згенерованих скриптах для програмування в IDA. На основі виставлених оцінок зроблено висновки щодо доцільності, можливості або недоцільності застосування ChatGPT для розв'язку кожного типу задач реверс-інжинірингу програмного забезпечення, а також надано відповідні рекомендації.

Ключові слова: ChatGPT, реверс-інжиніринг, програмне забезпечення.

Постановка проблеми. Поява широкодоступного чат-боту зі штучним інтелектом (ШІ) ChatGPT від лабораторії OpenAI, який побудовано на основі моделі GPT-3.5 (Generative Pre-trained Transformer), спонукала широкий загал спеціалістів різних галузей до застосування ШІ для рішення своїх професійних задач, в тому числі і в галузі інформаційних технологій (ІТ). Дослідники можливостей ChatGPT в першу чергу задаються питанням ефективності застосування ШІ для розв'язку як відносно простих рутинних задач, так і складних, дійсно інтелектуальних задач, які можна розділити на задачі аналізу (розпізнавання) та задачі синтезу (генерування) інформації. Усвідомлюючи, що ШІ створено людиною та навчено на

інформації, що створена людиною, можна передбачити потенційну недосконалість, а інколи й суперечність та навіть можливу хибність знань, отриманих ШІ. Особливістю ChatGPT як об'єкту аналізу є те, що його можна розглядати в двох аспектах: 1) ChatGPT - це програмне забезпечення (ПЗ), яке має певний функціонал, що підлягає тестуванню з метою оцінки його якості на основі певних стандартів якості ПЗ; 2) ChatGPT – це фахівець, який володіє спеціальними компетентностями (знаннями та вміннями) в певній галузі, і для його оцінки можна використати підходи, аналогічні тим, що застосовуються для оцінки якості фахової підготовки (освіти) фахівців. Але при будь-якому сприйнятті ChatGPT, без оцінки якості навряд чи можна довіряти йому відповідальну роботу, що робить задачу аналізу з метою оцінки можливостей ChatGPT актуальним завданням.

Звісно, що для глибокої оцінки якості ChatGPT мають бути розроблені відповідні методики та набори тестів для кожної спеціалізованої області застосування, але в цій статті застосовано традиційний освітянський підхід до оцінювання знань та вмінь ChatGPT як фахівця, який пройшов навчання по курсу “Реверс-інжиніринг програмного забезпечення” (РПЗ).

РПЗ як сукупність методів аналізу ПЗ, використовує багато різноманітних технік та інструментів розв'язку задач, які охоплюють такі складові ІТ, як комп'ютерні науки, програмна інженерія, кібербезпека та захист інформації. Відновлення проектних моделей, дослідження вихідного та виконуваного коду програм, пошук уразливостей та аналіз шкідливого програмного коду, декомпіляція, розпізнавання форматів файлів та інші задачі РПЗ потребують багато зусиль спеціалістів, тому застосування ChatGPT для рішення подібних задач виглядає очевидним вибором його в якості додаткового інструменту РПЗ.

Суть проблеми, що вирішується в цій статті, полягає в оцінювання знань та вмінь відкритої версії ChatGPT 3.5 щодо розв'язку типових задач РПЗ.

Аналіз останніх досліджень та публікацій. Сучасні дослідження із застосування ChatGPT в ІТ, що корелюються з задачами РПЗ, можна умовно розбити на наступні напрямки: реверс-інжиніринг (РІ) і дослідження якості коду, генерація програмного коду за допомогою ШІ в програмній інженерії, вирішення задач кібербезпеки, а також освіта та самоосвіта за цими напрямками із застосуванням ChatGPT.

Так, в роботі [1] представлено результати досліджень з відновлення назв ідентифікаторів в процесі декомпіляції бінарного коду в мову С. Показано, що застосування ChatGPT шляхом генерації потрібних запитів та автоматичного синтаксичного аналізу його відповідей по заданому шаблону спілкування з використанням ітераційного алгоритму дозволяє якісно відновити 75% ідентифікаторів по критерію семантичної відповідності назв їх призначенню, що покращує існуючі альтернативні методики на 16 - 20%.

В роботі [2] проведена оцінка математичних здібностей ChatGPT на основі набору даних математичних лінгвістичних моделей GHOSTS та визначено, що його знання відповідають рівню бакалавра з математики з середнім обсягом знань у 70%. Математичні здібності ChatGPT потрібні в РПЗ для відновлення математичного забезпечення (формул, застосованих математичних методів та алгоритмів) під час аналізу, наприклад, вихідного коду програм.

Робота [3] надає оцінку можливостей ChatGPT та CodeBert з аналізу безпеки програм за наступними напрямками: семантичний аналіз коду, аналіз уразливостей, здатність до узагальнення під час аналізу коду, динамічне визначення уразливостей та аналіз асемблерного коду. Визначено високу ефективність застосування ChatGPT для рішення подібних задач, але виявлено сильну залежність результатів від чіткості визначення змінних і назв функцій та класів, а також низьку продуктивність на великих обсягах коду. ChatGPT також продемонстрував можливості щодо генерації fuzzy-тестів для динамічного аналізу уразливостей, здатність до дизасемблювання бінарного коду, спроможність аналізувати хід виконання фрагментів асемблерного коду та навіть визначати значення регістрів, але виявив труднощі з високорівневим аналізом семантики асемблерного коду та пошуком в ньому уразливостей.

В роботі [4] досліджено здібності OpenAI ChatGPT, Google Bard та Microsoft Bing у вирішенні завдань CTF (capture the flag): веб-безпека, експлуатація уразливостей, криптографія, реверс-інжиніринг та форензика. ChatGPT показав найкращі результати, виконавши правильно шість завдань з семи.

В роботах [5] та [6] представлені уразливості ChatGPT, які можуть бути використані зловмисними користувачами для отримання шкідливої інформації в обхід етичних обмежень моделі. Досліджується, як кіберзловмисники можуть використовувати ChatGPT для розробки кібератак, а також сценарії, за якими зловмисники можуть використовувати ChatGPT для створення атак на основі соціальної інженерії, фішингові атаки, автоматизований злом, генерування корисного навантаження атак та поліморфне шкідливе програмне забезпечення (ШПЗ). Також розглядаються методи захисту з використанням ChatGPT для покращення заходів безпеки, включаючи автоматизацію кіберзахисту, звітність, розвідку загроз та ідентифікацію атак. Робиться висновок, що ChatGPT може зменшити навантаження на аналітиків центру безпеки (SOC), автоматично аналізуючи інциденти кібербезпеки. ChatGPT також допомагає аналітику давати стратегічні рекомендації для підтримки поточних та довгострокових заходів захисту.

Робота [7] присвячена порівнянню продуктивності інструментів генерації коду GitHub Copilot, Amazon CodeWhisperer та OpenAI ChatGPT з точки зору показників якості коду, таких як валідність, коректність, безпека, надійність та придатність коду до супроводу, для визначення їх переваг та недоліків. Аналіз показав, що ChatGPT має найкращі показники коректності коду і складає 65,2% при тестуванні на наборі даних HumanEval.

Робота [8] висвітлює кілька областей, де ChatGPT може бути ефективно застосований, включаючи виявлення неякісного коду, рефакторинг, генерацію, покрокове виконання вихідного коду, виправлення помилок, покращення розуміння коду та узагальнений аналіз коду. Зроблено висновки, що ChatGPT – це потужний інструмент, який можна використовувати для проектування ПЗ з використанням методів сучасної інженерії, а також для рефакторингу існуючого ПЗ за допомогою методів реверс-інжинірингу.

Стаття [9] містить каталог шаблонів запитів (prompt), спеціально створених для розробки програмного забезпечення. Ці шаблони охоплюють такі області, як виявлення вимог, проектування систем, генерація коду, рефакторинг коду тощо та підкреслюється, що такі шаблони забезпечують структурований підхід до взаємодії з ChatGPT і покращують якість коду, автоматизують загальні завдання розробки ПЗ та зменшують кількість помилок. Подібний підхід щодо створення шаблонів може використовуватись і для задач РППЗ із застосуванням ChatGPT.

В дослідженні [10] проведено порівняльний аналіз продуктивності ChatGPT 3.5 щодо генерації коду для 10 різних мов програмування. Зроблено висновки, що ChatGPT 3.5 демонструє здатність генерувати код для широкого спектру завдань, але його ефективність є непостійною: в одних випадках він створює синтаксично правильний код, тоді як в інших випадках може генерувати код з помилками (більше ніж у 50% випадків) або взагалі не генерувати код. Виявлено, що ChatGPT краще працює з мовами високого рівня ніж з мовами низького рівня.

В роботі [11] проаналізовано можливості ChatGPT для навчання програмуванню, а також його можливості стосовно допомоги у створенні програм, пошуку помилок, удосконаленні коду, а також для підтримки досліджень. Зроблено позитивний висновок щодо придатності ChatGPT для застосування в цих напрямках, але показано, що він все ще має певні обмеження, такі як відсутність здорового глузду, потенційну упередженість, труднощі зі складними міркуваннями і нездатність обробляти візуальну інформацію.

Таким чином, аналіз останніх публікацій свідчить про актуальність досліджень можливостей ChatGPT щодо його застосування для аналізу ПЗ в різних аспектах, в тому числі й для реверс-інжинірингу.

Метою статті є оцінка можливостей ChatGPT щодо розв'язку основних класів задач РППЗ.

Виклад основного матеріалу дослідження. Для проведення аналізу було сформовано загальний підхід, який полягає у виконанні наступних завдань:

- вибір класів задач РППЗ для оцінки можливостей ChatGPT щодо їх розв'язку з урахуванням наперед відомих обмежень;
- створення наборів тестових завдань та розробка методики проведення тестування;
- формування критеріїв оцінки якості розв'язку задач;
- проведення тестування;
- виставлення оцінок, узагальнення результатів та формування висновків.

Вибір класів задач РППЗ для тестування. ChatGPT має певні обмеження, які треба врахувати для виключення деяких класів задач РППЗ, з якими він точно не впорається:

- наявність веб-інтерфейсу тільки для ручного введення текстів або файлів в форматі PDF виключає задачі, вхідні дані яких представлені бінарними файлами або графічними моделями;
- неспроможність створювати графічні схеми при генеруванні відповідей виключає також задачі, результатами розв'язку яких мають бути графічні моделі;
- обмеження в 4096 токенів (слів або знаків пунктуації) у вхідному запиті та відповіді виключає задачі з великим обсягом вхідного коду;
- низька продуктивність обмежує можливість розв'язку задач в режимі реального часу.

З огляду на зазначені обмеження та загальне розуміння можливостей ChatGPT, для проведення тестування було обрано наступні класи задач РППЗ:

1. Аналіз вихідного коду ПЗ.
2. Аналіз бінарного коду ПЗ.
3. Аналіз моделей даних.

В кожному класі задач було обрано найбільш типові задачі РППЗ, які потребують високої кваліфікації, відповідних компетентностей та великих витрат часу спеціалістів для їх розв'язку.

Набори тестів та методика тестування. Для проведення тестування для кожного класу задач було виділено типові задачі, для кожної з яких було обрано по декілька тестових завдань, які розв'язувались здобувачами вищої освіти (магістрами) під час вивчення дисципліни “РППЗ”, а також було розроблено додаткові завдання. З огляду на те, що кінцевою метою дослідження є отримання висновків щодо придатності ChatGPT для розв'язку конкретних задач РППЗ, було обрано підхід щодо оцінки його компетентностей саме по окремих задачах без застосування накопичувальної системи.

Застосовано наступну методику тестування:

1. Для кожної задачі кожного класу пропонувалось для розв'язку декілька тестових завдань різного рівня складності, деякі з яких пропонувались для розв'язку повторно для оцінки різноманіття варіантів відповідей та стабільності здатності щодо їх розв'язку.
2. Для кожного тестового завдання формулювалось загальне питання (постановка задачі) та вводились всі необхідні дані.
3. При отримання неповної відповіді вводились додаткові питання.
4. При отриманні неправильної відповіді (розв'язку) вказувалось на місце та тип помилки.
5. При повторній помилці задавались навідні питання або підказки для самостійного виправлення.
6. Якщо помилка не виправлялась, надавались рекомендації щодо її виправлення.

При формуванні оцінки враховувались повнота та обґрунтованість відповіді, кількість заданих додаткових питань, кількість зроблених помилок та здатність до їх самостійного виправлення, отримання чи неотримання кінцевої правильної відповіді (розв'язку).

Критерії оцінювання. Для оцінювання якості розв'язку тестових завдань використано наступні критерії: коректність (правильність, збіг з очікуваннями); повнота (отримання кінцевого результату); точність (розв'язок завдання без додаткових питань); обґрунтованість (наявність пояснень та відповідь на питання). На основі цих критеріїв розроблено наступну систему оцінок:

А ("відмінно") – задача розв'язана коректно, повністю, з першої спроби, з поясненнями процесу розв'язку та правильними відповідями на додаткові уточнюючі питання;

В ("дуже добре") – задача розв'язана майже відмінно, але потребувала додаткових запитів;

С ("добре") – задача розв'язана, але потребувала додаткових запитів та підказок;

Д ("задовільно") – задача розв'язана не повністю або з помилками, потребувала додаткових запитів та підказок, але з самостійним виправленням помилок;

Е ("достатньо") – задача розв'язана частково, з помилками, що не виправлялись самостійно, але відповідь може бути використана як основа з подальшим ручним корегуванням;

Ф ("незадовільно") – задача не розв'язана або розв'язана з суттєвими помилками, потребує повторного навчання.

Отримані результати. Для проведення тестування було обрано наступні задачі визначених класів РППЗ та проведено тестування, результати яких викладено в наступному матеріалі.

1. Аналіз вихідного коду ПЗ. Реверс-інжиніринг вихідного коду містить безліч задач, які можуть включати як глобальні задачі відновлення проектних моделей у складі проектної документації, так і різноманітні часткові задачі аналізу коду в різних аспектах. В цьому класі задач було виділено часткові задачі, розроблено тестові завдання у вигляді прикладів програм на мовах C/C++ та отримано наступні результати.

1.1. Семантичний та структурний аналіз. Для тестування здатності ChatGPT щодо семантичного та структурного аналізу було використано наступні види програм:

- консольні процедурні програми, що склалися з декількох функцій;
- консольні об'єктно-орієнтовані програми, що склалися з декількох класів та шаблонів класів із застосуванням успадкування;
- фрагменти об'єктно-орієнтованих програм з графічним інтерфейсом користувача (Windows Forms Application), що містили інтерфейсні компоненти системи програмування Embarcadero C++Builder 10.3, велику кількість класів та складних структур даних.

Для кожної програми було сформульовано наступні тестові завдання:

- виконати семантичний аналіз програмного коду;
- виконати структурний аналіз програмного коду.

В усіх випадках ChatGPT відмінно проаналізував як суть програм, так і їхню структуру, надав пояснення по кожній функції або класу, надавав правильні відповіді по коду, а також рекомендації щодо його покращення. При виконанні завдань ChatGPT активно використовує змістовні назви ідентифікаторів та коментарі в програмі.

Висновок. ChatGPT вміє аналізувати як суть програм, так і їхню структуру. Оцінка – А.

1.2. Відновлення математичного забезпечення. Для тестування здатності ChatGPT відновлювати математичне забезпечення (алгоритми, визначати математичні методи та формули) було надано тестові приклади програм, що виконували розв'язок наступних задач:

- розв'язок квадратного рівняння;
- пошук найбільшого спільного дільника двох натуральних чисел за алгоритмом Евкліда;
- обчислення факторіалу рекурсивною функцією;
- обчислення числа комбінацій на основі функції обчислення факторіалу;
- знаходження площі фігури (визначений інтеграл) методом центральних прямокутників;

- обчислення тригонометричних функцій з використанням рядів Тейлора;
- сортування масивів на основі класичних алгоритмів;
- бінарний пошук в упорядкованому масиві чисел;
- просте шифрування даних на основі операції xor;
- генерація простих чисел;
- розкладання цілих чисел на прості множники з використанням генератора простих чисел;
- генерація чисел Фібоначчі.

Всі алгоритми, математичні методи та формули були правильно визначені (рис.1), хоча в деяких випадках приходилось задавати додаткові запити для отримання кінцевої математичної формули, а також виправляти неточності, як от неправильно ідентифікований метод центральних прямокутників (переплутав з методом правих прямокутників) для обчислення визначеного інтегралу.

<p><i>Визначити, яку математичну формулу обчислює програма:</i></p> <pre>float f(float a, float b, float (*g)(float)){ float dx = 0.001, s=0, x; for(x = a; x <= b; x += dx) s+=fabs(g(x + dx / 2) * dx); return s; } int main() { printf(“%f”, 2 * f(0, 1.57, cos); return 0; }</pre> <p><i>Кінцева відповідь ChatGPT:</i></p> $I = \int_a^b g(x) dx \approx 2 \cdot \sum_{x=a}^b g(x + \Delta x/2) \cdot \Delta x$ $2 \cdot \int_0^{1.57} \cos(x) dx$	<p><i>Відновити формулу з цієї програми:</i></p> <pre>int main() { double E, logarithm = 0, ai = 1, p = 1, x; int i = 1; cin >> x >> E; while(ai > E){ p *= x * x; ai = p / i; logarithm -= ai; i++; } cout << logarithm; return 0; }</pre> <p><i>Кінцева відповідь ChatGPT:</i></p> $\ln(x) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} (x - 1)^n$
--	---

Рисунок 1 – Приклади відновлення формул

Висновок. ChatGPT вміє розпізнавати класичні алгоритми та обчислювальні методи, відновлювати математичні формули. Оцінка – В.

1.3. Оцінка якості, аудит безпеки та рефакторинг. Ці три задачі були об’єднані в один тест, так як в процесі тестування ChatGPT саме так комплексно підходив до виконання тестових завдань, оцінюючи якість, безпеку та надаючи рекомендації щодо покращення коду. Для тестування було запропоновано декілька текстів програм, які мали проблеми з якістю та безпекою. ChatGPT визначав проблеми, надавав поради та самостійно виправляв наступні недоліки:

- читабельність і структура коду (форматував текст, переписував деякі фрагменти коду, створював нові функції для покращення структури коду);
- іменування змінних і функцій (перейменовував ідентифікатори на більш змістовні);
- відсутність коментарів (додавав коментарі);
- некоректність коду або неефективність алгоритму (пропонував та виконував заміну);
- наявність перевірок на помилки, такі як невірне введення даних користувача (виправляв);
- використання небезпечних функцій (надавав пояснення та робив заміну);
- переповнення буферу, як явні, так і неявні, які потребували аналізу виконання програми;

- низька стійкість алгоритму шифрування (визначав та виконував заміну на більш ефективний алгоритм);
- безпечна робота з файлами (виконував заміну коду з перевіркою наявності файлу, наявність прав доступу до файлу та обробкою помилок роботи з файлами);
- відповідність стандартам безпеки OWASP (виконав загальну оцінку);
- відповідність стандарту "CERT C++ Secure Coding Standard" (виконав загальну оцінку).

Висновок. ChatGPT відмінно аналізує якість вихідного коду, знаходить та виправляє знайдені проблеми в процесі рефакторингу. Але під час рефакторингу потребував додаткових запитів при виправленні проблем з безпекою коду, зокрема, щодо перевірки області допустимих значень для обчислень. Оцінка – В.

1.4. Генерація тестів. Для оцінки цієї спроможності ChatGPT було запропоновано:

- декілька процедурних та об'єктно-орієнтованих програм, що містили розрахунки по певних математичних формулах;
- програми з обробкою масивів чисел;
- програми, що містили уразливості переповнення буферу.

Було сформовано наступні тестові завдання:

- згенерувати набори тестів для заданої програми у вигляді таблиці (номер тесту, вхідні дані, очікувані результати);
- згенерувати модульні тести (unit test) для заданої програми;
- згенерувати вхідні дані для експлуатації уразливості переповнення буферу.

В цілому ChatGPT з усіма задачами впорався, згенерував таблицю тестів, підібрав значення для тестування програм з масивами, а також створив модульні тести для заданого шаблону класу. Потребував додаткових запитів та підказок щодо тестування допустимих значень для тригонометричних функцій. Також виявлено проблему з обчисленням довжини вхідних даних для експлуатації вразливості переповнення буферу (надав довжину вхідних даних вдвічі більше, ніж потрібно, після підказок виправився). Виявлено проблему ChatGPT у неправильному сприйнятті довжини та порядку розташування байтів даних, особливо у шістнадцятковій системі числення, що також було виявлено і в інших тестах під час оцінки його спроможностей щодо аналізу бінарного коду.

Висновок. ChatGPT здатен генерувати тести для тестування ПЗ, але потребує додаткових запитів та підказок. Оцінка – С.

1.5. Динамічний аналіз. Для оцінки спроможності ChatGPT аналізувати процес виконання програми за її текстом було запропоновано декілька тестових процедурних програм, що містили декілька функцій, в тому числі рекурсивних, та декілька об'єктно-орієнтованих програм, що містили як константні дані для обробки, так і вхідні дані, що вводились користувачем. Було сформульовано наступні тестові завдання:

- визначити результат виконання програми;
- визначити значення змінних в заданих точках програми;
- визначити наявність переповнення буферу заданої змінної в програмі;
- визначити результат програми, якщо користувач введе задані вхідні значення.

Висновок. Загалом із заданими тестами ChatGPT впорався, виявивши спроможність аналізувати процес виконання програми по заданому тексту, але у випадку програми з множинним успадкуванням класів були неточності, а також потребувались додаткові запити та підказки при підборі значень для переповнення буферу. Оцінка – С.

1.6. Деобфускація. Для оцінки здатності ChatGPT щодо "розплутування" спеціально ускладненого вихідного коду для захисту від РПЗ було запропоновано тестові приклади програм, що були обфусковані наступними методами:

- заміна інструкцій та фрагментів коду директивами препроцесора;
- додавання беззмістовних фрагментів коду ("мертвий код");
- ускладнення математичних виразів та умов циклів;

- заплутування логіки управління послідовністю обчислень.

Висновок. В процесі деобфускації ChatGPT впорався з усіма тестами, застосовуючи на додачу методи рефакторингу для покращення коду, але в деяких тестах не видаляв весь “мертвий код” з першого разу або без підказки. Оцінка – С.

1.7. Конвертація в іншу мову програмування. Для перевірки спроможності ChatGPT конвертувати програму з однієї мови програмування в іншу було запропоновано наступні тестові завдання для конвертації:

- з мови C/C++ в мову Pascal та навпаки;
- з мови C/C++ в мову Python та навпаки;
- з мови C/C++ в мову Java та навпаки;
- з мови C/C++ в мову Assembler.

Висновок. Загалом ChatGPT добре впорався із заданими тестами, конвертовані програми не містили синтаксичних помилок, але містили потенційні помилки при конвертації з мови Python в мову C, що було пов’язано з відсутністю визначення типів даних змінних в мові Python. ChatGPT при конвертації в мову Assembler зізнався в труднощах, але в цілому із завданням впорався, хоча й використовував нетипові для мови асемблера функції з мови C. Оцінка – В.

2. Аналіз бінарного коду. В цій групі тестів оцінювались спроможності ChatGPT щодо аналізу виконуваного коду у взаємодії з інструментом реверс-інжинірингу Hex-rays IDA.

2.1. Статичний аналіз. Для тестування було запропоновано як фрагменти, так і окремі секції асемблерного коду, отриманого в результаті дизасемблювання IDA. Було сформульовано наступні завдання:

- проаналізувати суть коду та надати пояснення;
- виконати структурний аналіз коду;
- визначити спосіб передачі параметрів функції;
- визначити значення параметрів виклику функцій;
- виконати рефакторинг коду.

Висновок. ChatGPT виявив добрі спроможності щодо розуміння асемблерного коду та суті всієї програми, вміє розпізнавати прологи та епілоги функцій, значення параметрів виклику функцій, розгалуження та цикли, але мав труднощі з розумінням представлення числових даних в секції даних, що задані шістнадцятковими значеннями, та потребував додаткових запитів щодо визначення точної назви способу передачі параметрів функції. Оцінка – С.

2.2. Аналіз уразливостей та шкідливого бінарного коду. Для тестування було надано дизасембльовані IDA коди програм з відповідними проблемами, та сформульовані завдання:

- проаналізувати ознаки шкідливого коду;
- знайти потенційні місця переповнення буферу.

В процесі аналізу ознак ШПЗ ChatGPT застосував стандартний підхід – аналіз викликів небезпечних функцій та їх аргументів, таких як створення файлів в системних каталогах, доступ до реєстру, запуск процесів, підключення до вікон інших застосунків, емуляція натискання клавіш клавіатури та інші, та пояснив суть небезпеки таких викликів. На додаткові конкретні питання надавав тільки загальні відповіді та рекомендації.

При пошуку місць переповнення буферу ChatGPT також обмежився аналізом небезпечних функцій та загальними рекомендаціями стосовно необхідності контролю розміру буферу.

Висновок. ChatGPT добре впорався з пошуком ознак ШПЗ на підставі системних викликів та обмежився тільки аналізом потенційно уразливих функцій. Оцінка – D.

2.3. Динамічний аналіз. Для оцінки здатності аналізувати процес виконання асемблерного коду по його тексту було запропоновано декілька простих програм. Правильні відповіді були надані тільки для коротких фрагментів програм з очевидними значеннями, а з більш складними завданнями ChatGPT не впорався: робив помилки, повідомляв про

неможливість проаналізувати хід виконання програми, потребував додаткових запитів та підказок, робив помилки при переведення кодів ASCII в символи.

Висновок. ChatGPT незадовільно виконує динамічний аналіз коду асемблера. Оцінка – F.

2.4. Декомпіляція. Для оцінки спроможностей ChatGPT щодо декомпіляції асемблерного коду в мову C було проведено два типи тестувань:

– введення дизасембльованих IDA асемблерних кодів секцій .text, .data, .rdata для самостійної декомпіляції ChatGPT;

– введення декомпільованих IDA псевдокодів мовою C для рефакторингу в ChatGPT.

В перших тестах ChatGPT добре впорався з завданням, але робив помилки при інтерпретації значень констант (масивів та рядків). Виявлено також властивість ChatGPT брати значення констант з їх ідентифікаторів без звертання до секції .rdata. В других тестах ChatGPT відмінно доводив псевдокод IDA до майже закінченого вихідного коду програми мовою C (рис.2).

Код ChatGPT	Псевдокод IDA	Код ChatGPT після IDA
<pre>#include <stdio.h> #include <string.h> #include <stdlib.h> int f(int arr[], int size) { int result = 0; for (int i = 0; i < size; ++i) { result += arr[i]; } return result; } int main(int argc, const char **argv, const char **envp) { int m[5]; char s2[12]; char s1[8]; int i; int n; m[0] = 1; m[1] = 0; m[2] = 2; m[3] = 3; m[4] = 4; strcpy(s1, "C++1.0"); s1[7] = '0'; n = 5; printf("Enter password:"); gets(s2); if (!strcmp(s1, s2)) { printf("Welcome!\n"); for (i = 0; i < n; ++i) { printf("%d ", m[i]); } int result = f(m, n); printf("Result = %d\n", result); system("Notepad"); system("pause"); return 0; } else { printf("\nWrong password!\n"); system("pause"); return -1; } }</pre>	<pre>int __cdecl f(int m, int n){ int s; // [esp+0h] [ebp-8h] int i; // [esp+4h] [ebp-4h] s = 0; for (i = 0; i < n; ++i) s += *(__DWORD *) (m + 4 * i); return s; } int __cdecl main(int argc, const char **argv, const char **envp){ int v4; // eax int m[6]; // [esp+8h] int i; // [esp+20h] [ebp-20h] int n; // [esp+24h] [ebp-1Ch] char s2[12]; // [esp+28h] char s1[8]; // [esp+34h] __int16 v10; // [esp+3Ch] [ebp-4h] strcpy(s1, "C++1.0"); s1[7] = HIBYTE(dword_4020CC); v10 = word_4020D0; n = 5; qmemcpy(m,&unk_4020D4, 0x14u); m[5] = 0; printf("Enter password:"); gets(s2); if (!strcmp(s1, s2)) { printf("\nWelcome!\n"); for (i = 0; i < n; ++i) printf("%d ", m[i]); v4 = f((int)m, n); printf("Result = %d\n", v4); system(aNotepad); system(aPause_0); return 0; } else { printf("\nWrong password!\n"); system(command); return -1; } }</pre>	<pre>#include <stdio.h> #include <string.h> #include <stdlib.h> int f(int arr[], int size) { int result = 0; for (int i = 0; i < size; ++i) { result += arr[i]; } return result; } int main(int argc, const char **argv, const char **envp) { int v4; int m[5] = {1, 2, 3, 4, 5}; int i; int n; char s2[12]; char s1[8]; __int16 v10; strcpy(s1, "C++1.0"); s1[7] = '0'; v10 = 0; n = 5; printf("Enter password:"); gets(s2); if (!strcmp(s1, s2)) { printf("Welcome!\n"); for (i = 0; i < n; ++i) printf("%d ", m[i]); v4 = f(m, n); printf("Result = %d\n", v4); system("Notepad"); system("pause"); return 0; } else { printf("Wrong password!\n"); system("pause"); return -1; } }</pre>

Рисунок 2 – Приклад результатів декомпіляції

Висновок. Декомпілятор IDA робить точний, але тільки псевдокод, який відмінно покращується в процесі рефакторингу ChatGPT з отриманням закінченого вихідного коду. Самостійно ж ChatGPT робить закінчений, але неточний та інколи помилковий код. Оцінка – C. Тому є сенс використовувати ChatGPT в кооперації з IDA, тоді це може бути оцінено на A.

2.5. Програмування в IDA. В цих тестах оцінювалась здатність ChatGPT створювати програми-помічники для автоматизації розв'язку задач РІПЗ в середовищі IDA шляхом генерації скриптів на мовах IDC (C-подібна скриптова мова) та IDAPython (спеціальна версія Python для IDA). В якості тестів було запропоновано згенерувати скрипти на обох мовах для 10 завдань, які стосувались пошуку потрібних даних в асемблерному коді з виведенням результатів в середині IDA або з експортом в зовнішній файл.

ChatGPT краще генерував скрипти на IDAPython, але жодного скрипта не було згенеровано без синтаксичних помилок (вставляв невідомі функції, пропускав оголошення змінних, не імпортував потрібні модулі), або скрипт не виконував свою задачу (не виводив потрібний результат або “зависав”). Тільки частину скриптів вдалося запустити після ручного корегування.

Висновок. ChatGPT може генерувати загальну логіку скриптів, але робить помилки, потребує додаткових запитів, підказок та ручного редагування. Оцінка – D.

3. Аналіз моделей даних. В цій групі тестів оцінювалась спроможність ChatGPT щодо реверс-інжинірингу моделей даних, які зберігаються в БД та бінарних файлах даних.

3.1. Реверс-інжиніринг реляційних БД. З огляду на те, що існує безліч спеціалізованих інструментів для РІ БД, в якості тестів було обрано наступні завдання РІ:

– по заданому опису структур таблиць, що задані по формі Назва_таблиці(атрибут₁, атрибут₂, ... , атрибут_n) згенерувати SQL-запит для їх створення для певної СУБД та заповнити тестовими даними;

– перетворити декілька SQL-запитів вводу даних (Insert) в табличне представлення;

– виконати міграцію БД з MySQL в Oracle з перенесенням структури БД та всіх даних.

В усіх тестах на вхід ChatGPT вводилось завдання та необхідні тексти SQL-запитів. З усіма задачами ChatGPT відмінно впорався.

Висновок. ChatGPT розуміє структуру БД та мову запитів SQL та спроможний розв'язувати задачі РІ БД. Оцінка – А.

3.2. Визначення структур бінарних файлів даних. В цих тестах оцінювалась здатність ChatGPT щодо визначення форматів нестандартних бінарних файлів, які створюються під час виконання певних програм. Ця задача в принципі є достатньо складною й для фахівців, тому для тестів було обрано достатньо прості задачі, такі як:

– знайти кореляцію між вхідними даними, що вводяться користувачем, результатами, що виводить програма на екран та вмістом бінарного файлу даних, що створює програма;

– перетворити фрагменти даних, що записано в бінарному файлі, у зрозумілу для людини форму (наприклад, чотирьохбайтні цілі числа, чотирьохбайтні дійсні числа).

В усіх тестах бінарні дані для ChatGPT вводились як текст в шістнадцятковій системі числення (один байт бінарних даних – дві символні шістнадцяткові цифри).

Висновок. ChatGPT незадовільно впорався з цими задачами, тому що робив багато помилок при перетворенні даних, не міг самостійно знайти кореляцію між вхідними і вихідними даними та вмістом файлу даних (зміг тільки підтвердити гіпотезу, яка була надана йому в якості підказки). Оцінка – F.

3.3. Створення правил YARA. В цьому тесті оцінювалась здатність ChatGPT генерувати правила для сигнатурного аналізатора YARA з метою відновлення типів файлів даних стандартних форматів для випадків, якщо розширення файлу в його назві відсутнє або спотворене. Було запропоновано створити правила, які б дозволяли розпізнати формати файлів Microsoft Office, зображень, веб-сторінок та виконуваних файлів (для випадку, якщо виконуваний код маскується під файл даних). ChatGPT з усіма задачами відмінно впорався, виявивши знання сигнатур файлів та розуміння синтаксису правил YARA.

Висновок. ChatGPT вміє генерувати правила YARA для визначення форматів бінарних файлів даних на основі сигнатур. Оцінка – А.

Висновки. Проведений аналіз можливостей ChatGPT 3.5 щодо розв'язку задач РІПЗ дозволяє зробити наступні висновки про те, що він:

- краще розв’язує задачі аналізу ніж задачі синтезу;
- краще розуміє програми на мовах високого рівня, ніж на мові Асемблера;
- найкраще розв’язує задачі семантичного та структурного аналізу вихідного коду, відновлення математичного забезпечення, оцінки якості, аудит безпеки та рефакторинг вихідного коду, декомпіляцію в кооперації з IDA, аналіз БД та створення правил для YARA;
- посередньо виконує аналіз уразливостей та шкідливого асемблерного коду, а також генерацію скриптів для IDA;
- не спроможний до динамічного аналізу асемблерного коду великого обсягу та аналізу бінарних файлів даних.

Загалом ChatGPT можна використовувати як додатковий до професійних інструмент РППЗ з обов’язковим контролем його рішень, а також для навчання реверс-інжинірингу.

Зведені оцінки та рекомендації щодо доцільності застосування ChatGPT 3.5 для розв’язку задач РППЗ представлені в табл.1.

Перспективами подальших досліджень є тестування нових версій ChatGPT та інших аналогічних систем ШІ щодо можливостей аналізу та синтезу графічних моделей ПЗ.

Таблиця 1 – Зведені оцінки ChatGPT по задачах

Задача	Оцінка	Висновки та рекомендації
1. Аналіз вихідного коду		
1.1. Семантичний та структурний аналіз	A	Доцільно застосовувати
1.2. Відновлення математичного забезпечення	B	Потребує контролю результату та уточнень
1.3. Оцінка якості, аудит безпеки та рефакторинг	B	Потребує контролю результату та уточнень
1.4. Генерація тестів	C	Потребує контролю та додаткових запитів
1.5. Динамічний аналіз	C	Потребує контролю та додаткових запитів
1.6. Деобфускація	C	Потребує контролю та додаткових запитів
1.7. Конвертація в іншу мову програмування	B	Потребує контролю та уточнень
2. Аналіз бінарного коду		
2.1. Статичний аналіз	C	Доцільно застосовувати для фрагментів коду
2.2. Аналіз уразливостей та шкідливого коду	D	Тільки для пошуку небезпечних функцій
2.3. Динамічний аналіз	F	Недоцільно застосовувати
2.4. Декомпіляція	C(A)	Доцільно застосовувати до псевдокоду IDA
2.5. Програмування в IDA	D	Можна застосовувати лише для генерації прототипу
3. Аналіз моделей даних		
3.1. Реверс-інжиніринг реляційних БД	A	Доцільно застосовувати
3.2. Визначення структур бінарних файлів даних	F	Недоцільно застосовувати
3.3. Створення правил YARA	A	Доцільно застосовувати

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] X. Xu et al., “LmPa: Improving Decompilation by Synergy of Large Language Model and Program Analysis”, *arXiv preprint arXiv:2306.02546*, 2023. doi: <https://doi.org/10.48550/arXiv.2306.02546>.
- [2] S. Frieder et al. “Mathematical capabilities of chatgpt”, *arXiv preprint arXiv:2301.13867*, 2023. doi: <https://doi.org/10.48550/arXiv.2301.13867>.
- [3] Z. Wang, Z. Lan, and L. Peng, “ChatGPT for Software Security: Exploring the Strengths and Limitations of ChatGPT in the Security Applications”, *arXiv preprint arXiv:2307.12488*, 2023. doi: <https://doi.org/10.48550/arXiv.2307.12488>.
- [4] W. Tann et al., “Using Large Language Models for Cybersecurity Capture-The-Flag Challenges and Certification Questions”, *arXiv preprint arXiv:2308.10443*, 2023. doi: <https://doi.org/10.48550/arXiv.2308.10443>.
- [5] M. Gupta, C. kiri, K. Aryal, E. Parker, and L. Praharaaj, “From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy”, *arXiv preprint arXiv:2307.00691*, 2023. doi: <https://doi.org/10.48550/arXiv.2307.00691>.

- [6] A. Qammar et al., “Chatbots to ChatGPT in a Cybersecurity Space: Evolution, Vulnerabilities, Attacks, Challenges, and Future Recommendations”, *arXiv preprint arXiv:2306.09255*, 2023. doi: <https://doi.org/10.48550/arXiv.2306.09255>.
- [7] B. Yetiştirten et al., “Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT”, *arXiv preprint arXiv:2304.10778*, 2023. doi: <https://doi.org/10.48550/arXiv.2304.10778>.
- [8] A. Sadik et al., “Analysis of ChatGPT on Source Code”, *arXiv preprint arXiv:2306.00597*, 2023. doi: <https://doi.org/10.48550/arXiv.2306.00597>.
- [9] J. White et al., “Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design”, *arXiv preprint arXiv:2303.07839*, 2023. doi: <https://doi.org/10.48550/arXiv.2303.07839>.
- [10] A. Buscemi, “A Comparative Study of Code Generation using ChatGPT 3.5 across 10 Programming Languages”, *arXiv preprint arXiv:2308.04477*, 2023. doi: <https://doi.org/10.48550/arXiv.2308.04477>.
- [11] Md. M. Rahman, and W. Yutaka, “ChatGPT for Education and Research: Opportunities, Threats, and Strategies”, *Applied Sciences*, vol. 13 (9), art. 5783, 2023. doi: <https://doi.org/10.3390/app13095783>.

Стаття надійшла 25.10.2023.

REFERENCE

- [1] X. Xu et al., “LmPa: Improving Decompilation by Synergy of Large Language Model and Program Analysis”, *arXiv preprint arXiv:2306.02546*, 2023. doi: <https://doi.org/10.48550/arXiv.2306.02546>.
- [2] S. Frieder et al. “Mathematical capabilities of chatgpt”, *arXiv preprint arXiv:2301.13867*, 2023. doi: <https://doi.org/10.48550/arXiv.2301.13867>.
- [3] Z. Wang, Z. Lan, and L. Peng, “ChatGPT for Software Security: Exploring the Strengths and Limitations of ChatGPT in the Security Applications”, *arXiv preprint arXiv:2307.12488*, 2023. doi: <https://doi.org/10.48550/arXiv.2307.12488>.
- [4] W. Tann et al., “Using Large Language Models for Cybersecurity Capture-The-Flag Challenges and Certification Questions”, *arXiv preprint arXiv:2308.10443*, 2023. doi: <https://doi.org/10.48550/arXiv.2308.10443>.
- [5] M. Gupta, C. Kiri, K. Aryal, E. Parker, and L. Praharaj, “From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy”, *arXiv preprint arXiv:2307.00691*, 2023. doi: <https://doi.org/10.48550/arXiv.2307.00691>.
- [6] A. Qammar et al., “Chatbots to ChatGPT in a Cybersecurity Space: Evolution, Vulnerabilities, Attacks, Challenges, and Future Recommendations”, *arXiv preprint arXiv:2306.09255*, 2023. doi: <https://doi.org/10.48550/arXiv.2306.09255>.
- [7] B. Yetiştirten et al., “Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT”, *arXiv preprint arXiv:2304.10778*, 2023. doi: <https://doi.org/10.48550/arXiv.2304.10778>.
- [8] A. Sadik et al., “Analysis of ChatGPT on Source Code”, *arXiv preprint arXiv:2306.00597*, 2023. doi: <https://doi.org/10.48550/arXiv.2306.00597>.
- [9] J. White et al., “Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design”, *arXiv preprint arXiv:2303.07839*, 2023. doi: <https://doi.org/10.48550/arXiv.2303.07839>.
- [10] A. Buscemi, “A Comparative Study of Code Generation using ChatGPT 3.5 across 10 Programming Languages”, *arXiv preprint arXiv:2308.04477*, 2023. doi: <https://doi.org/10.48550/arXiv.2308.04477>.

- [11] Md. M. Rahman, and W. Yutaka, "ChatGPT for Education and Research: Opportunities, Threats, and Strategies", *Applied Sciences*, vol. 13 (9), art. 5783, 2023. doi: <https://doi.org/10.3390/app13095783>.

VOLODYMYR SOKOLOV

ANALYSIS OF CHATGPT'S CAPABILITIES FOR SOLVING PROBLEMS OF REVERSE-ENGINEERING OF SOFTWARE

The article presents the results of the analysis and evaluation of the capabilities of the open version of the chatbot with artificial intelligence ChatGPT 3.5 for solving typical problems of software reverse engineering. Three classes of reverse engineering tasks were selected for analysis: source code analysis, binary code analysis, and data models analysis. In each class of tasks, the most typical tasks were selected taking into account the limitations of ChatGPT regarding the processing of graphical models and the amount of input and output data, and sets of test tasks were developed for each task. As an approach to assessment, an approach similar to the assessment of competencies of higher education applicants after studying the relevant discipline was chosen. The following were considered as criteria for evaluating answers to test tasks: correctness (rightness, coincidence with expectations); completeness (obtaining the final result); accuracy (solving the task without additional questions); reasonableness (availability of explanations and answers to questions). The following scale was used and described for evaluations: excellent, very good, good, satisfactory, sufficient, unsatisfactory. During the testing for each test task, tasks statement and all necessary data were entered through the ChatGPT interface. As a result of the analysis, it was determined that ChatGPT better solves the problems of source code analysis (excellent and very good grades were obtained for semantic and structural analysis, restoration of mathematical support, quality assessment, security audit and refactoring, as well as for conversion to another programming language), tasks of decompilation of IDA pseudocode into complete C source code, tasks of reverse engineering of relational databases, and tasks of generating YARA rules for recognizing file formats. Unsatisfactory grades were obtained for dynamic analysis of assembly code and determination of binary file structures of non-standard formats. ChatGPT solves the rest of the problems well and satisfactorily, but requires checking the results, clarifying queries and prompts, as well as manual error correction in some cases. ChatGPT errors were observed when analyzing binary data represented by hexadecimal symbols, as well as errors in generated scripts for programming in IDA. On the basis of the set grades, conclusions were made regarding the expediency, possibility, or impracticality of using ChatGPT to solve each type of software reverse engineering problems, and appropriate recommendations were also provided. Prospects for further research include testing new versions of ChatGPT and other similar artificial intelligence systems regarding the capabilities of analyzing and synthesizing graphical models of software.

Keywords: ChatGPT, reverse engineering, software.

Соколов Володимир Володимирович, кандидат технічних наук, доцент, доцент кафедри кібербезпеки і застосування інформаційних систем і технологій, Інститут спеціального зв'язку та захисту інформації Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна, ORCID 0000-0002-5779-7167, v.sokolov@kpi.ua.

Sokolov Volodymyr, candidate of technical sciences, associate professor, associate professor at the cybersecurity and application of information systems and technologies academic department, Institute of special communication and information protection of National technical university of Ukraine "Igor Sikorsky Kyiv polytechnic institute", Kyiv, Ukraine.