

РЕІНЖИНІРИНГ У СУЧАСНОМУ ІНФОРМАЦІЙНОМУ СВІТІ

Вінницький національний технічний університет

Анотація. У статті розглянуто поняття реінжинірингу програмного забезпечення, його еволюцію та сучасне застосування. Проаналізовано законні та нелегальні аспекти реверс-інжинірингу, зокрема використання методів декомпіляції, аналізу пам'яті, обходу захисту та модифікації виконуваних файлів. Особливу увагу приділено етичному хакінгу та використанню реінжинірингу для аудиту безпеки програмного забезпечення. Також висвітлено основні методи та інструменти реверс-інжинірингу та їхній вплив на захист сучасного програмного забезпечення.

Ключові слова: реінжиніринг програмного забезпечення, реверс-інжиніринг, етичний хакінг.

Abstract. The article examines the concept of software reengineering, its evolution, and modern applications. It analyzes the legal and illegal aspects of reverse engineering, including the use of decompilation methods, memory analysis, protection bypassing, and executable file modification. Special attention is given to ethical hacking and the use of reengineering for software security auditing. The article also highlights key methods and tools of reverse engineering and their impact on the protection of modern software.

Keywords: Software Reengineering, Reverse Engineering, Ethical Hacking.

Вступ

Вперше термін "реінжиніринг" (Reengineering) став широко використовуватись у 1990-х роках, коли його почали активно застосовувати у сфері бізнесу та програмної інженерії. У сфері програмного забезпечення (ПЗ) реінжиніринг означає аналіз, модифікацію та покращення існуючих програм. Сьогодні це поняття використовується для оптимізації, модернізації та забезпечення безпеки програмного забезпечення.

Реверс-інжиніринг та злам ПЗ не завжди є законним. У багатьох країнах, зокрема в США (DMCA), ЄС (EU Copyright Directive), Україні та інших, незаконний злам ПЗ карається штрафами або кримінальною відповідальністю.

Законний реінжиніринг дозволяється у випадках:

- аналізу безпеки (пентестинг, white-hat hacking);
- легальної модифікації open-source ПЗ;
- відновлення програмного коду, якщо компанія втратила вихідні файли тощо.

Автоматизація процесів реверс-інжинірингу для пошуку вразливостей у ПЗ привело до появи такого поняття, як «етичний хакінг».

Реінжиніринг для зламу програмного забезпечення

Реінжиніринг може використовуватись не тільки в легальних цілях (наприклад, для оновлення або аналізу безпеки ПЗ), але й для несанкціонованого доступу, тобто зламу програмного забезпечення:

- для зняття захистів: обхід DRM (Digital Rights Management, управління цифровими правами), ліцензійних ключів – видалення або підміна механізмів активації програм;
- для пошуку вразливостей, тобто для знаходження експлойтів для атак на ПЗ;
- для модифікації функціоналу – зміни коду, додавання прихованих можливостей або бекдорів;
- для крадіжки інтелектуальної власності, а саме для отримання алгоритмів, секретного коду або технологій компаній;
- для створення шкідливих програм (Malware Analysis) – аналізу вірусів для їх модифікації або обходу антивірусного захисту.

Для зламу програм через реінжиніринг використовувались такі методи:

1) Декомпіляція та дизасемблювання за допомогою таких інструментів, як IDA Pro, Ghidra, dnSpy, JADX, ILSpy, що відновлюють вихідний код програми для аналізу та модифікації.

2) Аналіз та модифікація пам'яті (Memory Hacking) за допомогою Cheat Engine, x64dbg, які змінюють значення змінних у пам'яті в реальному часі. Особливо це використовувалось для зламу ігор (зміна ресурсів, обходи перевірок) або витягування паролів та ключів шифрування без зміни коду.

3) Обхід механізмів захисту за допомогою OllyDbg, x64dbg, Frida, radare2 для видалення перевірки ліцензії, активації, DRM: аналіз точок входу та виходу функцій без необхідності повного декомпілювання або обхід антивідладочних механізмів.

4) Ін'єкція коду та модифікація виконуваних файлів шляхом додавання або зміни інструкції в .exe, .dll, .so файлах, що призводило до підміни функцій для зміни поведінки програми або до вставлення бекдорів у програмний код. Для цього використовувались такі інструменти, як PE Explorer, Hex Editors (010 Editor, HxD), API Hooking та інші.

У якості прикладів нелегального застосування реінжинірингу можна навести таке:

- злам ліцензії Windows – зміна реєстру або файлів активації;
- злам ігор – обхід античит-систем (наприклад, BattleEye, EasyAntiCheat);
- витягування паролів із програм – аналіз шифрування та алгоритмів автентифікації;
- модифікація мобільних додатків – патчинг .apk для обходу покупок в іграх.
- створення модифікованих версій програм – додавання прихованих функцій у сторонні програми.

Реінжиніринг як елемент етичного хакінгу

Реінжиніринг програмного забезпечення (ПЗ) значно змінився за останні десятиліття, враховуючи технологічні досягнення, нові виклики в кібербезпеці та зміну підходів до розробки.

Особливого розвитку реінжиніринг у хорошому сенсі цього поняття набув у 1990–2000-ті роки. В цей час основними цілями реінжинірингу було:

- перенесення старих програм на нові апаратні платформи;
- заміна застарілих мов програмування (наприклад, на C, C++, C#, Java);
- оптимізація продуктивності ПЗ без змін основної логіки роботи;
- покращення підтримки та документування коду.

Для цього використовувались такі методи, як

- декомпілювання та дизасемблювання для отримання вихідного (початкового) коду;
- ручний аналіз коду та його поступова зміна;
- використання статичних методів аналізу (наприклад, пошук «мертвого» або «надлишкового» коду);
- частковий або повний рефакторинг.

Спеціалісти, які займалися реінжинірингом, стикались з такими труднощами, як: відсутність ефективних автоматизованих засобів аналізу, обмежені можливості реверс-інжинірингу для обфускованих або зашифрованих програм, проблеми з ліцензійними обмеженнями на зміну програмного забезпечення.

Реінжиніринг 2020-их років має істотно інші цілі, а саме:

- мігрування ПЗ на хмарні платформи та мікросервісну архітектуру;
- автоматизація процесу реінжинірингу за допомогою AI та машинного навчання;
- підвищення безпеки шляхом аналізу вразливостей у старому коді;
- зміна технологічного стеку (наприклад, перехід з монолітної архітектури на контейнеризовані рішення, такі як Docker і Kubernetes).

Звичайно, змінились і методи, які використовують для цього. Серед них такі:

- 1) Використання інструментів штучного інтелекту для автоматичного аналізу та покращення коду (наприклад, GPT-4 для генерування та оптимізації коду).
- 2) Динамічний аналіз коду (Runtime Analysis) для оцінки продуктивності та вразливостей.
- 3) Автоматизовані інструменти реверс-інжинірингу, такі як AI-кодогенерація, Ghidra, IDA Pro, Radare2.
- 4) CI/CD (Continuous Integration/Continuous Deployment) для оновлення та оптимізації ПЗ без зупинки його роботи.

Але використання нових можливостей не означає, що проблеми у царині реінжинірингу зникли. Перед спеціалістами постали нові виклики. По-перше, кібербезпека – складність захисту коду через велику кількість уразливостей та експлойтів. По-друге, юридичні питання: посилення регулювання використання реверс-інжинірингу (GDPR, авторське право на програмне забезпечення і т. і.). По-третє, складність роботи з хмарними сервісами: необхідність оптимізації ПЗ під serverless-архітектуру.

Реінжиніринг у хорошому сенсі використовується для перевірки та підвищення рівня захисту програмного забезпечення. Ось основні причини його застосування:

1. *Аналіз вразливостей*, який дозволяє виявити слабкі місця в програмному коді, через які зловмисники можуть здійснювати атаки, наприклад, використання уразливостей типу *buffer overflow*, SQL-ін'єкцій тощо. Тобто, це спрямовано на виявлення його слабких місць, які можуть бути використані зловмисниками для атак. Аналіз вразливостей допомагає розробникам зрозуміти, наскільки їхній код захищений, і покращити безпеку системи.

2. *Оцінювання стійкості до реверс-інжинірингу*. Цей процес визначає, наскільки складно зловмиснику відновити вихідний код програми, проаналізувати його або внести несанкціоновані зміни. Якщо програмне забезпечення недостатньо захищене, хакери можуть легко здійснити реверс-інжиніринг, знайти вразливості або обійти захисні механізми, такі як активація, ліцензування чи перевірки цілісності.

3. *Перевірка ефективності захисних механізмів* – тестування ефективності методів шифрування, обфускації, антивідладкових механізмів та інших засобів захисту. Цей процес полягає у тестуванні та аналізі механізмів безпеки, які використовуються в програмному забезпеченні, щоб визначити, наскільки вони здатні протистояти атакам і несанкціонованим модифікаціям. Це допомагає виявити слабкі місця та підвищити рівень захисту програми.

4. *Аудит програмного забезпечення* – використовується для контролю безпеки коду перед випуском продукту або після виявлення загроз. Це процес аналізу та оцінки програмного коду, архітектури, безпеки, ліцензування та відповідності стандартам. Він допомагає виявити проблеми в коді, знайти вразливості, перевірити ефективність захисту та оцінити якість розробки. Аудит ПЗ включає:

- безпековий аудит (Security Audit);
- аудит вихідного коду (Code Review);
- ліцензійний аудит (License Compliance Audit);
- аудит продуктивності (Performance Audit);
- функціональний аудит (Functional Audit).

5. *Формування рекомендацій* щодо покращення безпеки – результати аналізу допомагають розробникам удосконалити архітектуру та алгоритми безпеки програмного засобу.

Висновки

Реінжиніринг у сучасному світі став набагато ефективнішим завдяки автоматизації та новітнім технологіям. Однак він став також складнішим через нові виклики у сфері кібербезпеки та хмарних обчислень. Якщо раніше реінжиніринг був спрямований на переведення старих програм на нові платформи, то сьогодні він охоплює питання безпеки, масштабованості та інтелектуальної автоматизації процесів. Разом з тим, кожний випадок використання реінжинірингу має свої причини, свої методи здійснення та інструменти для реалізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зворотний інжиніринг – це що таке, як працює, навіщо потрібен. URL : https://termin.in.ua/zvorotnyy-inzhynirynh-revers-inzhynirynh/?utm_source=chatgpt.com (дата звернення 22.02.2025).
2. Що таке зворотна розробка й де вона застосовується. Електронний журнал. URL : https://robotdreams.cc/uk/blog/274-hto-takoe-obratnaya-razrabotka-i-gde-ona-primenyaetsya?utm_source=chatgpt.com (дата звернення 03.03.2025).
3. National Institute of Standards and Technology (NIST). (2020). *Cybersecurity Framework*. Available at: <https://www.nist.gov/cyberframework>.
4. Cracknell, D. (2021). *The Ethics of Ethical Hacking: A Deep Dive into Cybersecurity Practices*. *Cybersecurity Journal*, 15(2), 45-62.
5. Microsoft Security Response Center (MSRC). (2023). *Reverse Engineering and Software Security Best Practices*. Available at: <https://msrc.microsoft.com/>.

КАПЛУН Валентина, ст. викл. кафедри захисту інформації, Вінницький національний технічний університет, Вінниця, valuka8379@gmail.com.

KAPLUN V. – Lecturer of the Chair of Safety of Infomation and Communication Systems, NTU, Vinnytsia.