УДК 004.056

doi.org/10.15407/fmmit2023.37.012

Абстракції зворотного інжинірингу при контролі та побудові захищених програмних реалізацій

Наталія Маслова

к. т. н., доцент, Донецький національний технічний університет, вул. Потебні, 56, м. Луцьк, Волинська обл., 43018, e-mail: nataliia.maslova@donntu.edu.ua

Зворотній інжиніринг має багато застосувань в ІТ. Це задачі сумісності, відтворення застарілих компонентів, аналізу та захисту програмних продуктів. Не зважаючи на актуальність, комплексний підхід до вирішення задач контролю та побудови захищених програмних реалізацій з урахуванням можливостей програмної інженерії та потреб інформаційної безпеки відсутній. В роботі продемонстрована можливість представлення процесу зворотного проєктування у вигляді послідовності абстракцій. Методи аналізу програмних реалізацій статичний, динамічний й, зокрема, експериментів, поєднуються моделлю абстракцій в логічну послідовність. А застосування метрик програмних продуктів надає можливість прив'язати контроль та розробку захисних механізмів програмних реалізацій до базових понять інформаційної безпеки, а саме – до розрахунку ризиків.

Ключові слова: зворотній інжиніринг, абстракція, захист, програмний продукт, метрики, метод

Вступ. Зворотна інженерія (Reverse Engineering) – це метод аналізу об'єкта, який допомагає зрозуміти, як сконструйовано систему або частину програмного забезпечення, загальну логіку роботи, при необхідності покращити функціонал, підвищити рівень захисту або провести перевірку рівня безпеки програмного продукту.

У галузі комп'ютерної та програмної інженерії зворотний інжиніринг застосовується при розробці та захисті програмного забезпечення, у комп'ютерній гральній індустрії, дизайні.

1. Постановка задачі

Процес зворотного інжинірингу має всі ознаки послідовного наукового дослідження [1]. Основними етапами процесу зворотного інжинірингу є збір інформації щодо програмного продукту; аналіз та структуризація програмного коду з виділенням процесної та інтерфейсної частин, структур даних тощо; побудова таблиць рішень; опис функціоналу; створення діаграм потоку даних між процесами; запис керуючої структури (ПЗ високого рівня); контроль коректності створеної структури; документування.

Мета роботи – визначити рівні абстракції й їх застосування для методів зворотного інжинірингу та аналізу програмних продуктів, прив'язати

застосування метрик програмної інженерії до процедури розрахунку ризиків й продемонструвати це на макеті програмної реалізації.

Вхідною інформацією процесу зворотного інжинірингу є необроблений (досліджуваний) код програмного продукту (файл, бітові дані). Результат - удосконалений, захищений, модифікований програмний код та документи специфікації до нього. Наявність первинної специфікації бажана, але не вимагається.

2. Методи й інструментальні засоби реверс-інжинірингу.

Основними методами, застосовними в галузі зворотного інжинірингу є методи експериментів («чорна скринька»), статичний, динамічний та статистичний аналізи [2].

Інструментальними засобами реінжинірингу є інтерактивні дизасемблери PE-редактори, API-монітори, шістнадцядкові редактори, фреймворки зворотного проєктування та аналізу двійкових файлів, програми для створення дампів, редагування таблиць переміщень. Крім того, наприклад, у [3] пропонується застосування графічних навігаторів програмного забезпечення (CIAO та CIA), візуальних інструментів розуміння, аналізу та відображення програмного забезпечення (Rigi), інструментів кластеризації, генераторів програм (GEN++).

З оглядом на таку кількість різноманітних інструментів зрозуміло, процес дослідження програмного забезпечення є трудомістким та складним. Тому розробка інтегрованого засобу аналізу коду є необхідною процедурою.

3. Вимоги та метрики

З метою планування часу, необхідного для виконання робіт з аналізу коду й оцінювання якості та складності виконаних робіт застосовуються спеціальні метрики (Холстеда, Джилба, Мак-Кейба, метрики програмної інженерії, як то кількість знайдених дефектів, їх щільність та ін). Приклад розрахунку метрик наведено на рис.1. Комплексний показник (гибридна метрика) розраховується за формулою

$$MI=(M_0 + K_1*F(M_1) + K_2* F(M_2) + \ldots.+K_n*F(M_n))/(K_1 + K_2+\ldots+Kn),$$

де M_0 – базова метрика, $F(M_i)$ – функціональна залежність, Ki - коефіцієнти, котрі визначаються за допомогою регресійного аналізу.

Metrics

Modules Names	CodeLine	Halstead	Cyclomatic Complexity	Complex indicator, %
sub_416BA0	175	80	215	8,83
sub_416BA1	84	53	112	30,89
sub_416BA2	205	77	53	29,23
sub_416BA4	57	18	47	46,59

Рис. 1. Розрахунок метрик

Комплексний показник може набувати значень від 0 до 100 і відображати: кількість структурних змін, зроблених з моменту попередньої перевірки; процент

помилок, виявлених під час перегляду коду або тестування програми; необхідність структурних змін для коректної роботи програми. Проводяться дослідження щодо застосування метрик якості програмного забезпечення до розрахунку ризиків (трьох факторна модель – загрози, вразливості та фінансові втрати, які пов'язані з фактором часу).

3. Рівні абстракції

З метою спрощення процесу зворотного інжинірингу вважаємо наявність трьох рівнів абстракції. Абстракція нижнього (низького) рівня – машинний код системи або програмного продукту. Може включати інформацію про збірку, формат заголовку файлу, застосовані регістри, розподіл пам'яті, перелік команд. Абстракція середнього рівня – компоненти, модулі системи, перелік функцій, інформація про інтерфейси, структури даних, потоки керування системою. Абстракція верхнього (високого) рівня – архітектура та дизайн системи. На цьому рівні створюються та відображаються діаграми, блок-схеми, структури програмного продукту.

4. Абстракція нижнього рівня

Спочатку аналізується бітовий код програми. Це аналог «чорної скриньки». Якщо досліднику нічого не відомо про програмний продукт, тоді на низькому рівні з'ясовується система, яка є середовищем відпрацьовування програмного продукту, тип аналізованого файлу та його особливості (наприклад, чи є файл виконуваним, архівованим, стисненим або шифрованим). Перевіряється наявність маркантів (часових, або спеціальних). Якщо файл не є виконуваним, то виявляється тип самого файлу й інструментальне середовище, у якому файл відпрацьовує.

При виявленні, що маємо виконуваний файлом, призначений для роботи в системі Windows (сигнатура 0x4D 0x5A (MZ) в перших двох байтах), на другому кроці аналізуємо PE-заголовок. Відшукують початкові й прикінцеві блоки PE-заголовку з метою аналізу можливості вбудовування захисних механізмів, адреси спеціальних структурних блоків, таких, як, наприклад, початок та довжину стеку даних, розподіл пам'яті тощо.

Загальний вигляд характеристик у вигляді блоку пам'яті зображено на рисунку 2.

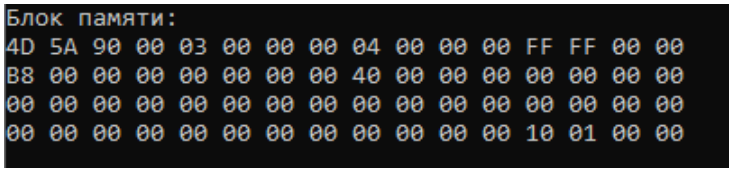

Рис. 2. Запис характеристик у пам'яті

Загалом, на першому рівні абстракції отримуємо близько 40 параметрів, що характеризують операційне середовище, на виконання в якому орієнтовано програмний продукт та ряд характеристик, серед яких є, наприклад, ознака

операційного середовища; розмір заголовка виконуваного файлу в абзацах; початкове значення регістру SP - адреси вершини стека; контрольна сума вмісту виконуваного файлу; ознака передачі управління програмі; адреса таблиці переміщень й т.п.

Отримані дані, застосовані на другому й третьому рівнях абстракцій, будуть сприяти прискоренню проведення аналізу. Етап чорної скриньки не є основним в класичному реінжинірингу. Але в галузі захисту програм та даних він надає до 40% необхідної на подальших етапах інформації.

6. Виявлення та відображення структури

Процедура відноситься до абстракції середнього рівня виявлення компонентів, модулів системи. Визначення переліку й структури функцій, отримання інформації про інтерфейси, структури даних, потоки керування системою.

Це етап статичного аналізу. Програма не запускається на виконання, а досліджується її виконуваний код.

На етапі статичного аналізу застосовуються модулі візуалізації, будуються та відображаються схеми розподілу пам’яті, структура програмної системи та взаємозв’язки між її модулями та функціями. Створюються таблиці з переліком функцій та їх взаємозв’язками. Виявляються змінні, їх формати, моменти застосування, взаємозв’язок та перевіряється інформація, отримана на низькому рівні абстракції щодо розміщення блоків даних у пам’яті.

Структура досліджуваного програмного продукту може бути відображена структурною схемою (рис.3), або схемою потоку даних DFD й може бути побудована на будь-якому рівні абстракції.

При виконанні задачі захисту програмного продукту на середньому рівні будується таблиця виявлених функцій зі зв’язками, переліком початкових адрес кожної функції та параметрів виклику, виявляються алгоритми роботи та призначення. Приймається рішення щодо можливості вдосконалення окремих блоків або заміни на спеціалізовані функції захисту.

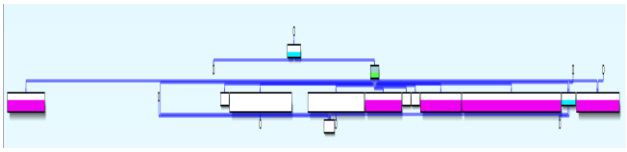

Рис. 3. Приклад відображення структури

Слід згадати й про статистичний аналіз, який також застосовується на цьому етапі для виявлення взаємозв’язку між змінними, частоти та місця їх застосування.

7. Вищій рівень абстракції

Останній, вищий рівень абстракції є етапом запуску програм в безпечному середовищі, покрокового виконання, уточнення отриманих на попередніх етапах характеристик та знань щодо структури програмного продукту, призначення

окремих блоків та модулів й програмного продукту в цілому. Виявляється, чи має програмний продукт оверлейну структуру, чи застосовуються динамічні бібліотеки. На цьому етапі тестуються застосовані засоби безпеки, проводяться експерименти з запуску блоків програм та часові заміри, перевіряється працездатність оновленого програмного продукту. Основним методом цього рівня є динамічний. Але основою його успішності й вхідними даними для роботи є інформація, здобута на попередніх рівнях.

Висновки. В роботі продемонстрована можливість представлення процесу зворотного інжинірингу в вигляді послідовності абстракцій, їх поєднання з етапами дослідження програмних продуктів. Вказано на можливість застосування метрик програмної інженерії для оцінки складності виконання задач аналізу й подальшої прив'язки отриманих показників до процедур інформаційної безпеки, а саме – до розрахунку ризиків. Продемонстровано макет програмної реалізації процедури зворотного інжинірингу.

Література

[1] Методологія наукових досліджень: навчальний посібник для підготовки докторів філософії/ КПІ ім. Ігоря Сікорського ; уклад.: *Астрелін І. М., Косогіна І.В., Кирій С.О*. – Електронні текстові данні. – Київ : КПІ ім. Ігоря Сікорського, 2021. – 121 с.
[2] *Каплун В.А.* Захист програмного забезпечення. Частина 2 : навчальний посібник / *В.А.Каплун, О.В.Дмитришин, Ю. В. Баришев* – Вінниця : ВНТУ, 2014. – 105с
[3] Software Engineering | Reverse Engineering, URL: https://www.geeksforgeeks.org/software-engineering-reverse-engineering/

Abstractions of reverse engineering in the control and construction of protected software implementations

Nataliya Maslova

Reverse engineering has many applications in IT. These are tasks of compatibility, reproduction of outdated components, analysis and protection of software products. Despite its relevance, there is no comprehensive approach to solving the problems of control and construction of protected software implementations, taking into account the capabilities of software engineering and the needs of information security. The paper demonstrates the possibility of presenting the process of reverse engineering in the form of a sequence of abstractions. The methods of analyzing static, dynamic, and, in particular, experimental software implementations are combined by a model of abstractions into a logical sequence. And the application of metrics of software products provides an opportunity to tie the control and development of protective mechanisms of software implementations to the basic concepts of information security, namely to the calculation of risks.

Отримано 14.03.23