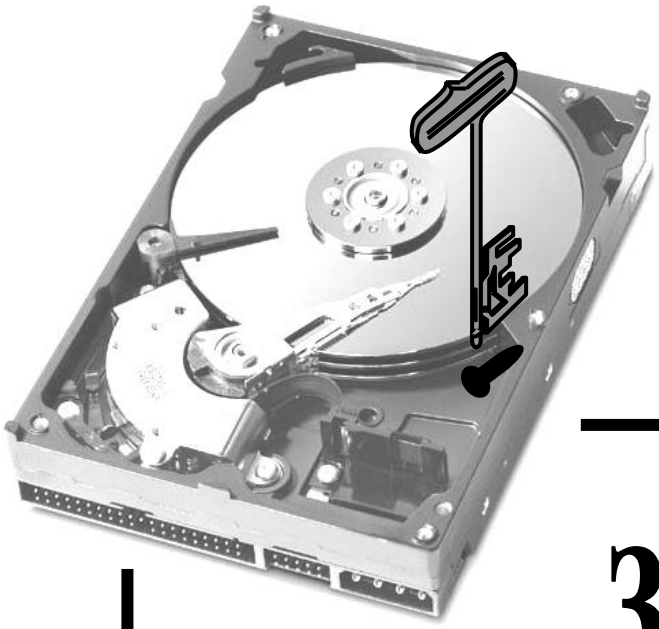
В.А. Каплун, А.В. Дудатьєв, В.П. Семеренко



ЗАХИСТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ЧАСТИНА 1

Міністерство освіти і науки України
Вінницький національний технічний університет

А.В. Дудатьєв, В.А. Каплун, В.П. Семеренко

ЗАХИСТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Частина 1

Затверджено Вченою радою Вінницького національного технічного університету як навчальний посібник для студентів напряму підготовки 1601 – «Інформаційна безпека». Протокол №4 від 25 листопада 2005 року

Вінниця ВНТУ 2005

УДК 681.3.07
Д 81

Рецензенти:

Р.Н. Квєтний, доктор технічних наук, професор
В.М. Михалевич, доктор технічних наук, професор
Р.Г. Тадевосян, кандидат технічних наук, доцент

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України

Дудатьєв А.В., Каплун В.А., Семеренко В.П.
Д 81 **Захист програмного забезпечення.** Частина 1. Навчальний посібник. – Вінниця: ВНТУ, 2005. – 140 с.

У навчальному посібнику наведено теоретичні відомості щодо методів захисту програмного забезпечення. Наведено характеристику сучасних систем захисту програмних продуктів та різних засобів, що застосовуються для зламу існуючих систем захисту. Розглянуто методи реалізації захисту програмного забезпечення від несанкціонованого копіювання.

Навчальний посібник призначений для студентів напряму підготовки 1601 – «Інформаційна безпека» для вивчення дисципліни «Захист програмного забезпечення».

УДК 681.3.07

ЗМІСТ

ВСТУП 6
1 ЗАГАЛЬНИЙ ОГЛЯД СИСТЕМ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 7
1.1 Мета і доцільність використання систем захисту 7
1.2 Класифікація системи захисту інформації 8
1.2.1 Пакувальники/шифратори 9
1.2.2 Системи захисту від несанкціонованого копіювання 10
1.2.3 Системи захисту від несанкціонованого доступу 10
1.3 Основні алгоритми захисту програмного забезпечення 15
1.4 Показники застосовності та критерії оцінювання СЗПЗ 17
1.5 Основні вимоги до розробки систем захисту ПЗ 19
1.6 Розповсюджені типи захистів та їх недоліки 20
1.6.1 Демо-версії програмного забезпечення 20
1.6.2 Захист обмеженням часу роботи програми 21
1.6.3 Захист, заснований на генерації серійного номера або на використанні зовнішнього “ключового файлу” 21
1.6.4 Використання унікальних ідентифікаторів носія програмного продукту 22
1.6.5 Навісні захисти 23
1.6.6 Захисти за допомогою електронних ключів 24
1.6.7 Надання розробникам ПЗ засобів SDK 24
1.7 Аналіз сучасних програмних продуктів для захисту програмного забезпечення 25
Контрольні питання 30
2 СУЧАСНИЙ СТАН ЗАСОБІВ ПОДОЛАННЯ СИСТЕМ ЗАХИСТУ 31
2.1 Проблема існування засобів зламу захистів ПЗ 31
2.2 Класифікація засобів подолання СЗПЗ 32
2.2.1 Програмні оболонки операційних систем 32
2.2.2 Програми-монітори 33
2.2.3 Сервісні програми операційних систем 36
2.2.4 Програми статичного та динамічного аналізу 39
2.2.5 Програми статичної та динамічної модифікації 41
2.2.6 Програми-емулятори 42
2.2.7 Апаратні засоби, що полегшують задачу зламу СЗПЗ 43
2.3 Програми розпакування, дешифрування та криптоаналізу 44
2.4 Висновки 45
Контрольні питання 46
3 ЗАХИСТ ВІД НЕСАНКЦІОНОВАНОГО КОПІЮВАННЯ 47
3.1 Методи розповсюдження програмного забезпечення 47

3.2 Технології захистів програмного забезпечення 48
3.3 Основні поняття ОС, необхідні для створення систем захисту програмного забезпечення 49
3.3.1 Склад операційної системи 49
3.3.2 BIOS: Базова система введення-виведення 49
3.3.3 CMOS: Complementary Metal Oxide Semiconductor 50
3.3.4 Переривання, їх роль і процедура звертання в програмах 50
3.4 Робота з дисками на фізичному рівні 53
3.4.1 Дискові накопичувачі та їх будова 54
3.4.2 Характеристики дискових накопичувачів 56
3.4.3 Програмування контролера НГМД 61
3.4.4 Функції BIOS для роботи з дисками 61
3.4.5 Функція _bios_disk 62
3.5 Форматування дискети 64
3.6 Захист від НСК методом прив'язки до дискети 65
3.6.1 Використання додаткової доріжки 66
3.6.2 Перестановка в нумерації секторів 66
3.6.3 Введення однакових номерів секторів на доріжці 66
3.6.4 Введення міжсекторних зв'язків 67
3.6.5 Зміна довжини секторів 68
3.6.6 Зміна міжсекторних проміжків 68
3.6.7 Інформація в проміжках 68
3.6.8 Перевищення обсягу доріжки 69
3.6.9 Введення логічних дефектів у заданий сектор 69
3.6.10 Контроль довжини доріжки 71
3.6.11 Зміна параметрів дисковода 71
3.6.12 Технологія "ослаблених" бітів 72
3.6.13 Фізичне маркування дискети 72
3.6.14 Збій синхронізації 73
3.6.15 Збійні чи відсутні адресні маркери 74
3.6.16 Різношвидкісні доріжки 74
3.6.17 Застосування фізичного захисного пристрою 74
Контрольні питання 75
3.7 Прив’язка до компакт-дисків 76
3.7.1 Найпростіші захисти 76
3.7.2 Диски великої ємності 77
3.7.3 Відхилення від стандарту записування на диск 77
3.7.4 Фізичні помилки на диску 77
3.8 Логічна структура диска 78
3.8.1 Розділи диска 79
3.8.2 Головний завантажувальний запис та його складові 80
3.8.3 Завантажувальний запис та процес завантаження ОС 81
3.8.4 Логічний номер сектора та переривання INT 25h і INT 26h 85

3.8.5 Таблиця розміщення файлів (FAT-таблиця) 86
3.8.6 Файли і каталоги 89
3.9 Доступ до файлової системи ОС 93
3.9.1 Одержання довідкової інформації 95
3.9.2 Робота з каталогами 96
3.9.3 Пошук у каталогах 98
3.9.4 Робота з файлами 99
3.9.5 Зчитування і записування інформації файлів 101
3.9.6 Позиціювання 102
3.9.7 Зміна дескриптора файлу 103
3.9.8 Таблиця відкритих файлів 105
3.9.9 Оброблення критичних помилок 105
3.10 Захист ПЗ методом прив'язки до комп'ютера 108
3.10.1 Прив’язка до вінчестера 109
3.10.2 Прив’язка до BIOS 111
3.10.3 Прив’язка до архітектури, до набору ПЗ і т.д. 113
3.10.4 Вимірювання продуктивності апаратури 116
3.10.5 Метод з частковим стиранням пам’яті 117
Контрольні питання 117
3.11 Електронні ключі захисту 119
3.11.1 Поняття електронного ключа, доцільність його використання 119
3.11.2 Будова електронного ключа 119
3.11.3 Класифікація електронних ключів за їх будовою 120
3.11.4 Класифікація електронних ключів за їх програмною частиною 123
3.11.5 Захист програм за допомогою електронного ключа 124
3.11.6 Методи зламу та протидії йому 126
3.11.7 Підвищення стійкості захисту 128
3.11.8 Можливості електронного ключа 130
3.11.9 Огляд та характеристика відомих ключів захисту 131
3.11.10 Перспективи розвитку електронних ключів 133
3.11.11 Правила використання електронних ключів 134
3.12 Захист ПЗ за допомогою опитування довідників 135
3.13 Введення обмежень на використання ПЗ 136
Контрольні питання 137
ЛІТЕРАТУРА 138
ДОДАТОК А. Параметри для стандартних типів НМД 139

ВСТУП

Бурхливий розвиток інформаційних технологій і використання їх у різних областях людської діяльності привело до того, що крім задач передачі, збереження й обробки інформації виникла не менш, а в ряді випадків і більш важлива задача захисту інформації. Постали такі проблеми, як незаконне використання алгоритмів, що є інтелектуальною власністю автора, несанкціоноване використання, модифікація, поширення і збут програмних продуктів. Це обумовило необхідність використання відповідних систем захисту.

На сьогоднішній день системи захисту програмного забезпечення широко поширені і знаходяться в постійному розвитку завдяки розширенню ринку програмних продуктів і телекомунікаційних технологій. У навчальному посібнику представлено класифікацію систем захисту програмних розробок, їх види, основні алгоритми і методи, які використовуються при побудові систем захисту. Велика увага приділена аналізу сучасного стану програмних і апаратних засобів для захисту програм як вбудованих, так і навісних, а також засобам зламування існуючих захистів.

У навчальному посібнику детально розглядаються методи захисту від несанкціонованого копіювання шляхом прив'язування до дистрибутивних носіїв програмних продуктів, до архітектури комп'ютера, шляхом опитування довідників, шляхом використання електронних ключів і т.д.

Навчальний посібник орієнтований на практичне вивчення засобів мов програмування для забезпечення вбудованого захисту від копіювання програмних продуктів. Представлений теоретичний матеріал може бути практично використаний при розробці системи захисту програмного забезпечення від несанкціонованого використання.

1 ЗАГАЛЬНИЙ ОГЛЯД СИСТЕМ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Мета і доцільність використання систем захисту

Перед початком проектування будь-якої систему захисту програмного забезпечення (СЗПЗ) треба цілком чітко собі уявляти, що саме і від кого ми збираємось “захищати”. Не існує абсолютно надійних методів захисту. Кваліфіковані системні програмісти, що користуються сучасними засобами аналізу програмного забезпечення (ПЗ) (налагоджувачі, дизасемблери, перехоплювачі переривань і т.п.), маючи досить часу, зможуть подолати практично будь-який захист. І тому при проектуванні систем захисту (СЗ) слід передбачати те, що рано чи пізно цей захист буде знято.

Метою проектування повинен бути вибір такого методу захисту, що забезпечить неможливість несанкціонованого копіювання для заздалегідь визначеного кола осіб і на обмежений час. Наприклад, якщо ми збираємось захистити від копіювання комерційну версію програми, нам не обов’язково захищати цю версію “назавжди”, оскільки вартість такого захисту перевищуватиме вартість самої програми. Цілком досить того, щоб спосіб захисту неможливо було “розгадати” до моменту з’явлення наступної версії нашої програми, а в новій версії змінити спосіб захисту. Коротше кажучи, рівень захисту повинен бути таким, щоб вигідніше було купити програму, ніж займатись зламом захисту від копіювання.

Іноді захищати програми від копіювання взагалі немає необхідності. Фірма MicroSoft і багато інших фірм не захищають свої програмні продукти від копіювання. Для приваблювання покупців ці фірми встановлюють низькі ціни на свої вироби, забезпечують супроводження їх на високому рівні та якісну документацію. Нові версії продаються зареєстрованим користувачам із значними знижками. З’являються нові версії швидко, і тому є сенс купляти нові версії програм, а не копіювати старі.

Таким чином, для вибору способу організації захисту необхідним є індивідуальний підхід у кожному конкретному випадку.

Системи захисту ПЗ широко поширені і знаходяться у постійному розвитку завдяки розширенню ринку ПЗ і телекомунікаційних технологій. Необхідність використання систем захисту програмного забезпечення обумовлена рядом проблем, серед яких варто виділити:

- *промислове шпигунство* - незаконне використання алгоритмів, що є інтелектуальною власністю автора, при написанні аналогів продукту;
- *крадіжка і копіювання* - несанкціоноване використання ПЗ ;
- *несанкціонована модифікація ПЗ* з метою впровадження програмних зловживань;
- *піратство* - незаконне поширення і збут ПЗ.

1.2 Класифікація системи захисту інформації

Існуючі системи захисту програмного забезпечення можна класифікувати за рядом ознак, серед яких можна виділити:

- метод установлення,
- використовувані механізми захисту;
- принцип функціонування.

За методом установлення системи захисту ПЗ можна розділити на:

1. ***Системи, що встановлюються на скомпільовані модулі ПЗ***. Такі системи найбільш зручні для виробника ПЗ, оскільки легко можна захистити вже цілком готове й протестоване ПЗ (звичайно процес установлення захисту максимально автоматизований і зводиться до вказання імені файлу, що захищається, і натисканню клавіші "Enter"), а тому і найбільш популярні. У той же час стійкість цих систем досить низька (у залежності від принципу дії СЗ), оскільки для обходу захисту досить визначити точку завершення роботи "конверта" захисту і передачі керування захищеній програмі, а потім примусово її зберегти в незахищеному вигляді.
2. ***Системи, що вбудовуються у початковий код ПЗ до компіляції.*** Системи цього типу незручні для розробника ПЗ, оскільки виникає необхідність навчати персонал роботі з програмним інтерфейсом (API) системи захисту, внаслідок чого збільшуються грошові і часові витрати. Крім того, ускладнюється процес тестування ПЗ і знижується його надійність, оскільки крім самого ПЗ помилки може містити API системи захисту або процедури, що його використовують. Але такі системи є більш стійкими до атак, оскільки тут зникає чітка границя між системою захисту і ПЗ як таким.
3. ***Комбіновані***. Це найбільш “живучі” системи захисту. Зберігаючи переваги систем першого типу і недоліки систем другого типу, вони максимально утрудняють аналіз і дезактивацію своїх алгоритмів.

За механізмами захисту СЗ можна класифікувати так:

1. Системи, що використовують ***складні логічні механізми***. Системи цього типу використовують різні методи і прийоми, орієнтовані на утруднення дизасемблювання, налагодження та аналіз алгоритму СЗ і програмного забезпечення, що захищається. Цей тип СЗПЗ найменш стійкий до атак, оскільки для подолання захисту досить проаналізувати логіку процедур перевірки і належним чином їх модифікувати.
2. Системи, які використовують ***шифрування ПЗ***, що захищається. Вони є більш стійкими. Для дезактивації таких захистів необхідним є визначення ключа дешифрації ПЗ.

3. ***Комбіновані системи***, які найбільш стійкі до атак.

За принципом функціонування СЗ можна розділити на:

- пакувальники/шифрувальники;
- СЗПЗ від несанкціонованого копіювання (НСК);
- СЗПЗ від несанкціонованого доступу (НСД).

1.2.1 Пакувальники/шифратори

Спочатку основною метою пакувальників/шифраторів було зменшення на диску обсягу модуля, що виконується, без нанесення збитків для функціональності програми, але пізніше на перший план вийшла мета захисту ПЗ від аналізу його алгоритмів та несанкціонованої модифікації. Для досягнення цього використовуються:

- алгоритми компресії даних;
- прийоми, пов'язані з використанням недокументованих особливостей операційних систем (ОС) і процесорів;
- шифрування даних, алгоритми мутації, заплутування логіки програми, приведення ОС у нестабільний стан на час роботи програми і ін.

Використання пакувальників та шифраторів має як позитивні, так і негативні сторони.

Позитивні сторони.

1. У рамках періоду безпечного використання дані системи забезпечують високий рівень захисту ПЗ від аналізу його алгоритмів.
2. Методи пакування/шифрування набагато збільшують стійкість систем захисту інших типів.

Негативні сторони.

1. Практично всі методи захисту, що застосовуються при цьому, уповільнюють виконання коду ПЗ.
2. Шифрування/пакування коду ПЗ викликає утруднення при відновленні (update) і виправленні помилок (bugfix, servicepack).
3. Можливе підвищення апаратно-програмних вимог ПЗ.
4. У чистому вигляді дані системи не застосовуються для авторизації використання ПЗ.
5. Дані системи застосовні лише до продуктів невеликого обсягу (до 1Мбайта).
6. Даний клас систем уразливий, оскільки програмний код все ж таки повинен бути розпакований або розшифрований для його виконання.
7. Мають невеликий термін безпечного використання (через п.4).
8. Пакування і шифрування виконуваного коду у сучасних ОС вступає в конфлікт із забороною коду, що самомодифікується.

1.2.2 Системи захисту від несанкціонованого копіювання

Системи захисту від несанкціонованого копіювання здійснюють прив'язку ПЗ до дистрибутивного носія (гнучкого диску, CD, ...). Даний тип захистів ґрунтується на глибокому вивченні роботи контролерів накопичувачів, їх фізичних показників, нестандартних режимів розбиття при форматуванні, зчитування/запису і т.п. При цьому на фізичному рівні створюється дистрибутивний носій, що, наприклад, володіє неповторними властивостями (зазвичай це досягається за допомогою нестандартної розмітки носія інформації і запису на нього додаткової інформації, пароля або мітки), а на програмному рівні створюється модуль (контролююча частина програми), налаштований на ідентифікацію й аутентифікацію носія за його унікальними властивостями. При цьому можливе застосування прийомів, що їх використовують пакувальники/шифратори.

Позитивні фактори.

1. Утруднення нелегального копіювання і поширення ПЗ.
2. Захист прав користувача на придбане ПЗ.

Негативні фактори.

1. Велика трудомісткість реалізації системи захисту.
2. Уповільнення продажів через необхідність фізичної передачі дистрибутивного носія інформації.
3. Підвищення системних вимог через введений захист (наприклад, обов’язкова наявність накопичувача).
4. Зниження стійкості до відмов ПЗ (зростає можливість помилок).
5. Несумісність захисту і апаратури користувача (накопичувач, контролер).
6. На час роботи системи захисту накопичувач або інший пристрій виявляються зайнятими, що неможливо при використанні деяких програмних продуктів.
7. Існує загроза крадіжки захищеного носія.

1.2.3 Системи захисту від несанкціонованого доступу

СЗПЗ від НСД здійснюють попередню чи періодичну аутентифікацію користувача ПЗ або його комп'ютерної системи шляхом опитування додаткової інформації. До цього типу СЗ можна віднести:

- системи парольного захисту ПЗ;
- системи прив'язки ПЗ до комп'ютера користувача;
- системи з ключовими дисками;
- апаратно-програмні системи з електронними ключами.

У першому випадку ключову інформацію вводить користувач, у другому – вона міститься в унікальних параметрах комп'ютерної системи

користувача, у третьому – вона зберігається на диску, а у четвертому випадку ключова інформація зчитується з мікросхем електронного ключа.

Парольні захисти

Цей клас СЗПЗ на сьогоднішній день є найпоширенішим. Основний принцип роботи даних систем полягає в ідентифікації та аутентифікації користувача ПЗ шляхом запиту додаткових даних (це можуть бути назва фірми і/або ім'я і прізвище користувача, його пароль або реєстраційний код). Ця інформація може запитуватися в різних ситуаціях, наприклад, при старті програми, після закінчення терміну безкоштовного використання ПЗ, під час виклику процедури реєстрації, в процесі установлення на ПК користувача.

Процедури парольного захисту прості в реалізації і, тому вони дуже часто застосовуються виробниками ПЗ. Більшість парольних СЗПЗ використовують логічні механізми, що зводяться до перевірки правильності пароля (або реєстраційного коду) і запуску або незапуску ПЗ, у залежності від результатів перевірки.

Існують також системи, які шифрують ПЗ, що захищається, і використовують пароль або похідну від нього величину як ключ дешифрації. Більшість таких систем використовує слабкі або найпростіші алгоритми шифрування, нестійкі до спрямованих атак. Це відбувається через складність коректної реалізації стійких криптоалгоритмів і недоцільності їх застосування для захисту недорогих умовно безкоштовних програмних продуктів, які складають більшість ПЗ, що використовує парольні захисти. Але останнім часом розроблені парольні СЗПЗ, що реалізують стійкі криптоалгоритми типу DES і RSA, вони реалізовані у вигляді захисного модуля і допоміжних бібліотек і встановлюються на вже скомпільовані модулі ПЗ.

Слабкою ланкою парольних захистів є блок перевірки правильності введеного пароля/коду. Для такої перевірки можна порівнювати введений пароль із записаним у коді програми правильним або з правильно сгенерованим із введених додаткових даних паролем. Крім того, є можливість порівняння похідних величин від введеного і правильного паролів, наприклад, їх хеш-функцій, у такому випадку в коді можна зберігати тільки похідну величину, що підвищує стійкість захисту. Шляхом аналізу процедур перевірки можна знайти реальний пароль, записаний у коді програми, знайти правильно сгенерований пароль із введених даних або створити програму для перебору паролів для визначення пароля з потрібною хеш-сумою. Крім того, якщо система захисту не використовує шифрування, досить примусово змінити логіку перевірки для одержання безперешкодного доступу до ПЗ.

Системи, що шифрують, більш стійкі до атак, але при використанні найпростіших або некоректно реалізованих криптоалгоритмів є небезпека дешифрації ПЗ.

Для всіх парольних систем існує загроза перехоплення пароля при його введенні авторизованим користувачем. Крім того, у більшості СЗПЗ даних типів процедура перевірки використовується лише одноразово, як правило, при реєстрації чи установленні ПЗ, потім система захисту просто відключається, що створює реальну загрозу для несанкціонованого доступу (НСК) при незаконному копіюванні ПЗ.

Позитивні сторони.

1. Надійний захист від зловмисника-непрофесіонала.
2. Мінімальні незручності для користувача.
3. Можливість передачі пароля/коду по мережі.
4. Відсутність конфліктів із системним і прикладним ПЗ й апаратним забезпеченням.
5. Простота реалізації і застосування.
6. Низька вартість.

Негативні сторони.

1. Низька стійкість більшості систем захисту даного типу.
2. Користувачу необхідно запам'ятовувати пароль/код.

Системи прив'язки ПЗ

Системи цього типу при установленні ПЗ на персональний комп’ютер користувача здійснюють пошук унікальних ознак комп'ютерної системи або ці унікальні ознаки встановлюються самою системою захисту. Після того цей модуль захисту в самому ПЗ налаштовується на пошук і ідентифікацію даних ознак, за якими надалі визначається авторизоване чи неавторизоване використання ПЗ. При цьому можливе застосування методик оцінювання швидкісних і інших показників процесора, материнської плати, додаткових пристроїв, ОС, зчитування/запис у мікросхеми енергонезалежної пам'яті, запис прихованих файлів, настроювання на найбільше використовувану карту ОЗП і т.п.

Слабкою ланкою таких захистів є той факт, що на комп’ютері користувача ПЗ завжди запускається на виконання, що приводить до можливості примусового збереження ПЗ після відпрацьовування системи захисту, дослідження самого захисту і виявлення даних, використовуваних СЗПЗ для аутентифікації ПК користувача.

Позитивні фактори.

1. Не потрібно додаткових апаратних засобів для роботи захисту.
2. Утруднення несанкціонованого доступу до скопійованого ПЗ.
3. Простота застосування.
4. "Невидимість" СЗПЗ для користувача.

Негативні фактори.

1. Помилкові спрацьовування СЗПЗ при будь-яких змінах у параметрах комп’ютера.
2. Низька стійкість при доступі зловмисника до ПК користувача.
3. Можливість конфліктів із системним ПЗ.

Програмно-апаратні засоби захисту ПЗ з електронними ключами

Цей клас СЗПЗ останнім часом здобуває все більшу популярність серед виробників програмного забезпечення. Під програмно-апаратними засобами захисту у даному випадку маються на увазі засоби, засновані на використанні так званих апаратних (електронних) ключів. *Електронний ключ* – це апаратна частина системи захисту, що являє собою плату з мікросхемами пам'яті і, у деяких випадках, мікропроцесором, розміщену в корпусі і призначену для установлення в один із стандартних портів ПК (COM, LPT, USB ...) чи слот розширення материнської плати. Також в якості такого пристрою можуть використовуватися СМАРТ-карти. За результатами проведеного аналізу, програмно-апаратні засоби захисту в даний момент є одними із найстійкіших систем захисту ПЗ від НСД.

Електронні ключі за архітектурою можна розділити на *ключі з пам'яттю* (без мікропроцесора) і *ключі з мікропроцесором* (і пам'яттю).

Найменш стійкими (у залежності від типу програмної частини) є системи з апаратною частиною першого типу. У таких системах критична інформація (ключ дешифрації, таблиця переходів) зберігається в пам'яті електронного ключа. Для дезактивації таких захистів у більшості випадків необхідна наявність у зловмисника апаратної частини системи захисту (основна методика: перехоплення діалогу між програмною й апаратною частинами для доступу до критичної інформації).

Самими стійкими є системи з апаратною частиною другого типу. Такі комплекси містять в апаратній частині не тільки ключ дешифрації, але і блоки шифрації/дешифрації даних. У такий спосіб при роботі захисту в електронний ключ передаються блоки зашифрованої інформації, а приймаються звідти розшифровані дані. У системах цього типу досить складно перехопити ключ дешифрації, оскільки всі процедури виконуються апаратною частиною, але залишається можливість примусового збереження захищеної програми у відкритому вигляді після відпрацьовування системи захисту. Крім того, до них можуть бути застосовані методи криптоаналізу.

Позитивні фактори.

1. Значне утруднення нелегального поширення і використання ПЗ.
2. Звільнення виробника ПЗ від розробки власної системи захисту.
3. Висока автоматизація процесу захисту ПЗ.
4. Наявність API системи для більш глибокого захисту.
5. Можливість легкого створення демо-версій.

6. Досить великий вибір таких систем на ринку.

Негативні фактори.

1. Утруднення розробки і налагодження програми через обмеження з боку СЗ.
2. Додаткові витрати на придбання системи захисту і навчання персоналу.
3. Уповільнення продажів через необхідність фізичної передачі апаратної частини.
4. Підвищення системних вимог через захист (сумісність, драйвери).
5. Зниження відмовостійкості програмного забезпечення.
6. Несумісність систем захисту і системного або прикладного ПЗ користувача.
7. Несумісність захисту й апаратури користувача.
8. Обмеження через несумісність електронних ключів різних фірм.
9. Зниження розширюваності комп'ютерної системи.
10. Утруднення або неможливість використання захищеного ПЗ в переносних і блокнотних ПК.
11. Розмір і вага апаратної частини (для COMM/LPT вони становлять 5×3×2см ~ 50гр).
12. Загроза крадіжки апаратного ключа

Засоби захисту ПЗ з ключовими дисками

В даний момент цей тип систем захисту малорозповсюджений через його моральне старіння. СЗПЗ цього типу багато в чому аналогічні системам з електронними ключами, але тут критична інформація зберігається на спеціальному ключовому носії. Так само багато спільного є і з системами захисту від копіювання, оскільки використовуються ті ж методи роботи з ключовим носієм.

Основною загрозою для таких СЗПЗ є перехоплення зчитування критичної інформації, а також незаконне копіювання ключового носія.

Позитивні і негативні сторони даного типу СЗПЗ практично цілком збігаються з такими в систем з електронними ключами:

Позитивні фактори.

1. Значне утруднення нелегального поширення і використання ПЗ.
2. Звільнення виробника програмного забезпечення від необхідності розробки власної системи захисту.
3. Висока автоматизація процесу захисту ПЗ.
4. Можливість легкого створення демо-версій.

Негативні фактори.

1. Утруднення розробки і налагодження програм через обмеження з боку СЗ.

2. Додаткові витрати на придбання системи захисту і навчання персоналу.
3. Уповільнення продажів через необхідність фізичної передачі носія.
4. Підвищення системних вимог через захист.
5. Зниження відмовостійкості програмного забезпечення.
6. Несумісність систем захисту і системного або прикладного програмного забезпечення користувача.
7. Несумісність захисту і апаратури користувача.
8. Зниження розширюваності комп'ютерної системи.
9. Утруднення чи неможливість використання захищеного ПЗ в переносних і блокнотних ПК.
10. Загроза крадіжки ключового носія.

Необхідно відзначити, що користувачами явно відчуваються лише негативні сторони систем захистів. А виробники ПЗ розглядають лише "плюси" та "мінуси" систем даних захисту і практично не розглядають фактори, що відносяться до кінцевого споживача.

1.3 Основні алгоритми захисту програмного забезпечення

Для захисту ПЗ використовується такі методи та алгоритми:

- *алгоритми заплутування* – використовуються хаотичні переходи в різні частини коду, впровадження помилкових процедур-"пустишок", холості цикли, спотворення кількості реальних параметрів процедур ПЗ, розкидання ділянок коду по різних областях оперативного запам’ятовуючого пристрою (ОЗП) і т.п.;
- *алгоритми мутації* – створюються таблиці відповідності операндів-синонімів та їх взаємозамінність при кожному запуску програми за визначеною схемою або випадковим чином;
- *алгоритми компресії даних* – програма упаковується, а потім розпаковується по ходу виконання;
- *алгоритми шифрування даних* – програма шифрується, а потім розшифровується по ходу виконання;
- *обчислення складних математичних виразів в процесі відпрацьовування механізму захисту* – елементи логіки захисту залежать від результату обчислення значення будь-якої формули або групи формул;
- *методи утруднення дизасемблювання* – використовуються різні прийоми, спрямовані на запобігання дизасемблювання в пакетному режимі;
- *методи утруднення налагодження* – використовуються різні прийоми, спрямовані на ускладнення налагодження програми;
- *емуляція процесорів і операційних систем* – створюється віртуальний процесор і/або операційна система (не обов'язково реально існуючі) і програма-перекладач із системи команд IBM у систему команд створеного процесора або ОС. Після такого перекладу ПЗ може

виконуватися тільки за допомогою емулятора, що різко утруднює дослідження алгоритму ПЗ

- *нестандартні методи роботи з апаратним забезпеченням* – модулі системи захисту звертаються до апаратури ЕОМ, минаючи процедури операційної системи, і використовують маловідомі або недокументовані її можливості.

У свою чергу, зловмисники так само застосовують ряд методів і засобів для порушення систем захисту (рис.1.1).

Рисунок 1.1 – Схема протистояння технічних методів і засобів захисту ПЗ

Ситуація протистояння розробників СЗПЗ і зловмисників постійно змінюється за рахунок комбінування вже відомих методів захисту і нападу, а також за рахунок створення і використання нових методів.

1.4 Показники застосовності та критерії оцінювання СЗПЗ

За результатами дослідження був розроблений набір показників застосовності і критеріїв оцінювання СЗПЗ.

Показники застосовності.

Технічні. Відповідність СЗПЗ функціональним вимогам виробника ПЗ і вимогах до стійкості, системні вимоги ПЗ і системні вимоги СЗПЗ, обсяг ПЗ й обсяг СЗПЗ, функціональна спрямованість ПЗ, наявність і тип СЗ в аналогах ПЗ - конкурентів.

Економічні. Співвідношення втрат від піратства і загального обсягу прибутку, співвідношення втрат від піратства і вартості СЗПЗ та її впровадження, співвідношення вартості ПЗ і вартості СЗПЗ, відповідність вартості СЗПЗ і її впровадження поставленим цілям.

Організаційні. Поширеність і популярність ПЗ, умови поширення і використання, унікальність, наявність загроз, імовірність перетворення користувача в зловмисника, роль документації і підтримки ПЗ.

Критерії оцінювання.

Захист як такий. Утруднення нелегального копіювання, утруднення нелегального доступу, захист від моніторингу, відсутність логічних проломів і помилок у реалізації системи.

Стійкість до дослідження/зламу. Застосування стандартних механізмів, нові/нестандартні механізми.

Відмовостійкість (надійність). Імовірність відмови захисту (НСД), час напрацювання на відмову, імовірність відмови програми захисту (крах), частота помилкових спрацьовувань.

Незалежність від конкретних реалізацій ОС. Використання недокументованих можливостей, "вірусних" технологій і "дір" ОС.

Сумісність. Відсутність конфліктів із системним ПЗ, відсутність конфліктів із прикладним ПЗ, відсутність конфліктів з існуючим апаратним забезпеченням, максимальна сумісність з майбутніми апаратним та програмним забезпеченням.

Незручності для кінцевого користувача ПЗ. Необхідність і складність додаткового настроювання системи захисту, доступність документації, доступність інформації про відновлення модулів системи захисту через помилки/несумісності/нестійкості, доступність сервісних пакетів,

безпека мережної передачі пароля/ключа, затримка через фізичну передачу пароля/ключа, порушення прав споживача.

Побічні ефекти. Перевантаження трафіка, відмова в обслуговуванні, уповільнення роботи зали захищеного ПЗ, уповільнення роботи ОС, захоплення системних ресурсів, перевантаження ОЗУ, порушення стабільності ОС.

Вартість. Вартість/ефективність, вартість/ціна ПЗ, що захищається, вартість/ліквідовані збитки.

Доброякісність. Правдива реклама, доступність результатів незалежної експертизи, доступність інформації про побічні ефекти, повна інформація про СЗ для кінцевого користувача.

В цілому загальна картина взаємодії агентів ринку програмного забезпечення може бути представлена на схемі, наведеній на рис. 1.2.

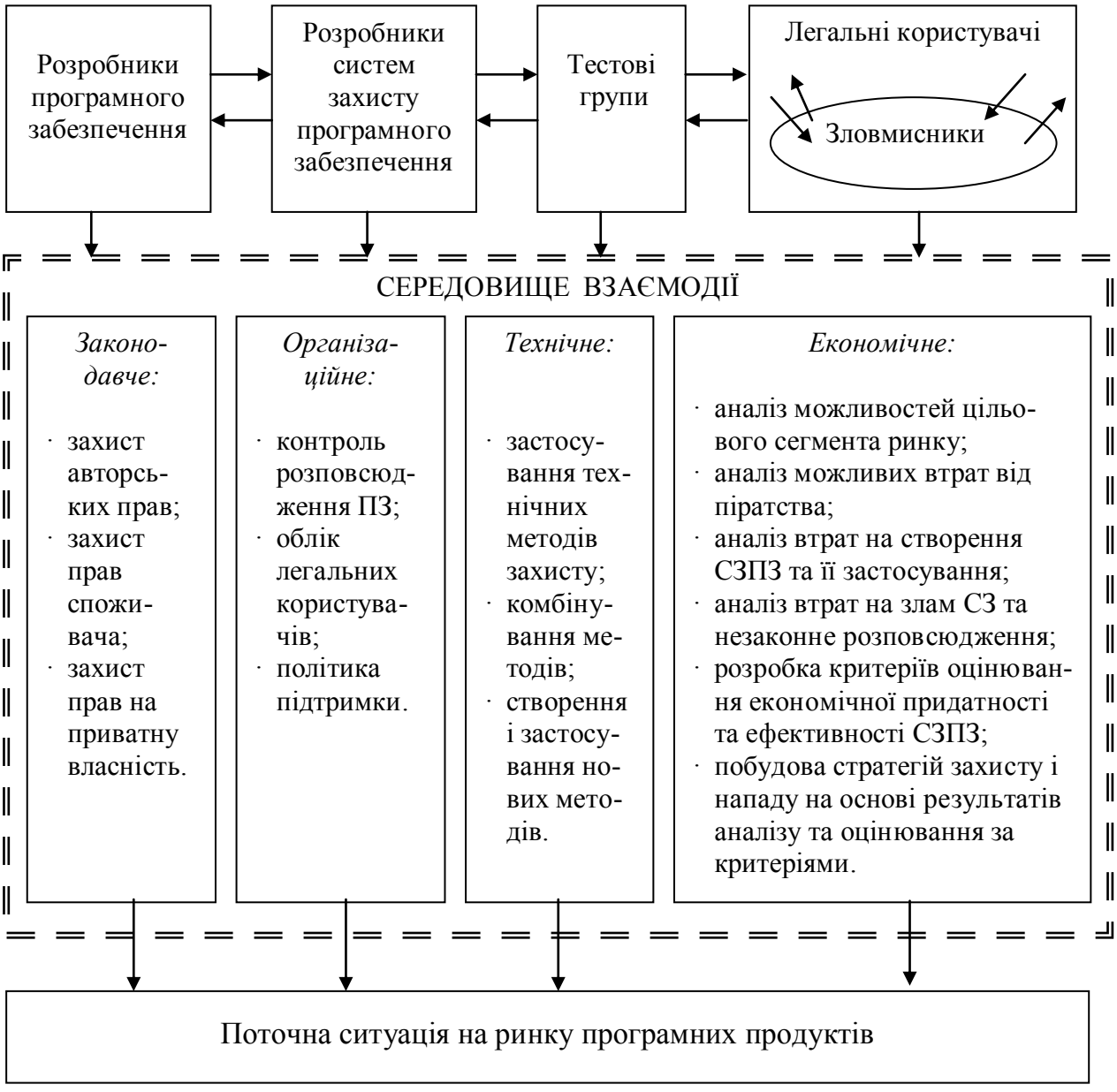


Рисунок 1.2 – Схема взаємодії учасників створення та розповсюдження ПЗ

З чотирьох зазначених у наведеній схемі видів середовища взаємодії стороні, що захищається, підконтрольні (чи частково підконтрольні) три види – організаційне, технічне й економічне середовище. Очевидно, що найважливішим середовищем взаємодії є економічне середовище, оскільки економічна взаємодія у даному випадку є першопричиною і метою усієї взаємодії. При розробці та аналізі захисту програмного забезпечення необхідно враховувати існуючу законодавчу базу, при цьому потрібно проводити докладний економічний аналіз ситуації, застосовуючи різні критерії оцінювання, а потім створювати стратегію захисту, що включає застосування технічних і організаційних заходів для захисту ПЗ.

1.5 Основні вимоги до розробки систем захисту ПЗ

На жаль, Україна залишається однією з головних споживачів піратської продукції, оскільки в нашій країні існує солідний попит на продукцію софтверних компаній, а рівень доходів населення не дозволяє купувати ліцензії за загальносвітовими цінами. Звідси і високий попит на піратську продукцію, що з'являється незважаючи на системи захисту від копіювання, які виробники встановлюють на свої розробки. Розкриваються і зламуються всі, ну чи майже всі захисти!

На початку 90-х років практично кожна з компаній-виробників ПЗ розробляла свою систему захисту від копіювання з тим чи іншим ступенем надійності. Також у середині 90-х років багатьма компаніями застосовувалася тактика фізичного впливу на торговців продукцією, але і ці заходи не набули належного ефекту.

Таким чином, існує необхідність самотужки захищати програмне забезпечення за допомогою тих програмних засобів, що будуть протидіяти піратам заполоняти ринок контрафактною продукцією.

Вимоги, які можна висунути до системи захисту, щоб вона змогла стати популярною, такі:

- захист повинен бути з *великим запасом міцності*, тобто враховувати високий рівень піратів взагалі, і вітчизняних зокрема, бути здатною протистояти їх натиску довгий час;
- використовувати для захисту *недорогі додаткові апаратні пристосування*, що підвищують вартість самого захисту, а отже, і кінцевого продукту;
- *не дуже прив'язуватися до апаратної конфігурації комп'ютера*, оскільки персональний комп'ютер не є "річ у собі", і його окремі компоненти можуть і повинні бути замінними в міру старіння;
- система захисту повинна бути заснована на *оригінальних принципах захисту від зламу*. Показником критерію може служити той факт, що захист ще не зламаний або, якщо і зламаний, то не всіма можливими способами;

- система захисту *не повинна перешкоджати вільному копіюванню* захищених даних (повинна забороняти тільки несанкціонований запуск, тобто копіюється копіювальником, але не виконується);
- *несуттєво збільшує вартість* кінцевого продукту.

Практично всі захисти уже зламані, не зважаючи на хитрування, що застосовуються компаніями-захисниками. Тут діють кілька золотих правил: по-перше, те, що зробила одна людина, інша може завжди зламати, а, по-друге, тираж і популярність продукту стимулюють хакерів на дуже активні дії. Отже, сподіватися на 100% надійний захист від зламу – тільки тішити себе ілюзіями! Зате можна, розробивши ефективну систему захисту, максимально ускладнити життя піратам, а собі гарантувати повернення інвестицій. Говорячи простою мовою, якщо захист розкривається 4-5 місяців, то компанія може продати достатню кількість копій свого продукту, щоб покрити усі витрати і дістати прибуток. А якщо нова версія продукту виходить через 7-8 місяців, то комерційний успіх можна повторювати до нескінченності. Але це можливо в тих випадках, коли продукт користується великим попитом, а захист постійно модифікується.

У найбільш вигідному світі тут представлений тільки захист StarForce, що використовує ряд цікавих рішень, які роблять її злам практично неможливим або важкоздійснюваним. З інформації, яку вдалося одержати з офіційних джерел, розробники StarForce застосували спосіб динамічного шифрування коду, і навіть не стільки шифрування, у загальному розумінні цього слова, а в тому, що для цих цілей розроблена спеціальна віртуальна машина зі своєю мовою програмування. Цей підхід дозволяє не просто шифрувати блоки коду, а транслювати на іншу мову програмування і створити тим самим віртуальну машину.

1.6 Розповсюджені типи захистів та їх недоліки

На сьогоднішній день існує досить велика кількість способів захисту програмного забезпечення. На деяких типах захисту не будемо зупинятися, оскільки вони не одержали широкого розповсюдження на ринку ПЗ. Природно, що всі перераховані нижче типи останнім часом комбінуються, що ускладнює їх злам, але все-таки не робить його неможливим.

1.6.1 Демо-версії програмного забезпечення

Проаналізувавши досить велику кількість таких “демок”, можна зробити висновок, що це непоганий спосіб захисту від несанкціонованого використання ПЗ.

До *недоліків* цього виду програм слід віднести те, що якщо ми пишемо повноцінний програмний продукт, то слід не забувати написати окрему демо-версію цього продукту, у якому дійсно не буде якихось істотних

функцій. Багато авторів програмних розробок поступають так: пишуть повноцінну програму, а потім з неї “роблять демо-версію”, вставляючи декілька “хитрих переходів” перед викликом ряду функцій. Ці самі “переходи” неначебто виключають ці функції, і у користувача складається враження, що він користується демо-версією програми. Насправді ж варто виправити 2-3 байти в коді програми і одержати повноцінний продукт. Як приклад, можна привести досить відому програму “Астра-Д”. У демо-версії її, наприклад, відключена така опція, як друкування, але код, відповідальний за друкування, в програмі присутній, і простою заміною 3-х байтів проблема легко розв’язується.

Отже, при розробці захисту даного типу *рекомендується* при написанні демо-версій програм “обрізати” цілком функції, які ми хочемо виключити з “демки”.

1.6.2 Захист обмеженням часу роботи програми

Часові обмеження (наприклад, 30 днів) або обмеження за кількістю запусків – не зовсім вдалий спосіб захисту і піддається зламу навіть “крекеру”-початківцю. Цей досить давній спосіб захисту, і тому найбільш уразливий, “ламається” новачками, що лише вчора прочитали документацію до Soft-Ice і пару статей на reversing.net. Тут немає нових рішень. Реалізують або програмний лічильник днів (запусків), або при першому запуску програми прописується дата запуску до реєстру Windows і потім при кожному запуску порівнюється з тим, скільки пройшло днів (запусків), або роблять те саме, тільки використовують не реєстр, а файл, що приховується на твердому диску. Єдине, що є позитивним у цьому захисті, це те, що цей спосіб захисту, як правило, самостійно майже не використовується, а комбінується з якимись іншими способами захисту.

Отже, для покращення захисту даного типу *рекомендується н*іколи не користуватися одним лише цим типом захисту, а обов'язково комбінувати його з іншими типами захистів.

1.6.3 Захист, заснований на генерації серійного номера або на використанні зовнішнього “ключового файлу”

Захист на основі генерації серійного номера в залежності від імені (компанії) користувача – також досить слабкий спосіб, хоча іноді і зустрічаються досить вдалі реалізації цього типу захистів. У цю же групу можна включити захисти, засновані на генерації серійного номера в залежності від відбитка, створеного програмою при першому її запуску. Причому відбиток генерується програмою в залежності від того, які параметри має “залізо” комп'ютера. Тут набагато більше вдалих рішень, але сам спосіб не зручний для “юзерів”, що купили таку програму, – адже при модернізації

“заліза” йому знову прийдеться звертатися до автора, а це не завжди можливо і зручно.

Захисти, що базуються на так званих “key-file” захисту, – це коли програма при кожному запуску шукає такий файл у своїй директорії, і якщо знаходить, то зчитує зашифровану інформацію з нього, розшифровує її, порівнює контрольну суму, отриману при розшифровці. Якщо все гаразд, то програма працює як зареєстрована, якщо ж такого файлу немає або інформація при порівнянні є критичною, то програма не запускається або працює в демо-режимі. Досить непоганий спосіб, який починаючі “крекери”, не можуть швидко зламати, якщо цей захист грамотно реалізований.

Тут *недоліком* або основною помилкою, є або використання занадто простих алгоритмів генерації серійного номера, що приводить до написання “кейгена”, або навпаки – алгоритм досить складний, але залишено занадто багато лазівок для реалізації “бітхака”.

Рекомендацією з покращення захисту даного типу може бути те, що при використанні подібних захистів слід використовувати кілька алгоритмів генерації з можливо більшою довжиною ключа. Щоб захиститися від “бітхака” при неможливості написання генератора ключів, треба включити більше перевірок правильності серійного номера й інтегрувати ці перевірки в основні функції програми. При цьому бажаним є використання шифрування коду програми.

1.6.4 Використання унікальних ідентифікаторів носія програмного продукту

Способів є тут не так вже й багато. Це нанесення лазерних міток на визначеній частині диска й одержання унікальних характеристик, властивих носію. І все це без нанесення фізичних міток. Подібне поки удалося зробити тільки таким системам захисту, як StarForce, CD-Cops і TAGES, що разюче відрізняє їх від конкурентів. По-перше, не треба мати складне устаткування для нанесення міток, адже, як ми знаємо, що будь-яке подорожчання виробничого циклу б'є по кишенях покупців, підвищуючи вартість продукту. І тут ще один аспект: чим вища вартість продукту, тим більша увага до нього з боку піратів, тим скоріше захист буде зламаний, а спосіб зламу розтиражований.

Отже, приходиться констатувати той факт, що фізичні прив'язки помалу відходять у минуле, а на зміну їм приходять подібного роду способи захисту. І ще слід відмітити, захисти StarForce, CD-Cops і TAGES не ламуються (поки що) побітовими копіювальниками і емуляторами, оскільки фізичні характеристики носіїв хакери ще не навчилися відтворювати програмно (програми-емулятори), і їх неможливо копіювати (програми-копіювальники).

При використанні цього типу захисту виникає необхідність розробки ще одного атрибуту захисту – розробка *резидентної програмної частини, здатної на 100% ідентифікувати мітку або характеристику носія*, з якого вона була запущена.

1.6.5 Навісні захисти

До них відносяться системи захисту типу Vbox, AsProtect, AsPack і т.д. За цим типом захистів велике майбутнє. Досить непогані рішення, але знову ж переважна більшість подібних захистів слабка й у мережі Internet існує величезна кількість утиліт, що знімають автоматично такі захисти. Але це найбільш перспективний напрямок. Пояснимо чому. Більшість таких захистів, включають всі розглянуті вище типи відразу, “в одному флаконі”, плюс до всього використовуються алгоритми стискання модулів, що виконуються, і шифрування коду програми. Найбільш яскравий приклад цієї групи – AsProtect – найбільш вдалий на сьогоднішній день. Хоча просунуті “крекери” при наявності певного досвіду і певних засобів, успішно справляються і з цим продуктом.

Окрему групу складають дуже відомі захисти, застосовувані у світі іграшок. Такі продукти, як LaserLock, SecuRom, SafeDisc, StarForce і ін., як правило, об’єднують захисти п'ятого і третього типу: програма ніби прив'язується до диска, на який вона записана, тобто вона зашифрована за допомогою відбитка оригінального диска. Очевидно, що тут виникають деякі проблеми при аналізі. Найбільш грамотною в цьому плані можна назвати програму StarForce (саме вона використана для захисту гри “Рандеву з незнайомкою”).

Серед *недоліків* назвемо лише ті недоліки, які зустрічаються дуже часто, а саме:

- нереалізовані антиналагоджувальні прийоми або реалізовані ті з них, які нейтралізуються спеціальними утилітами типу FrogIce. Це в решті решт приводить до розпаковки\розшифровки і зняття захисту;
- слабкі алгоритми шифровки коду, що дозволяє зняти захист навіть не розшифровуючи код програми (прямо під налагоджувачем);
- нереалізована або реалізована невдало перевірка контрольної суми і цілісності коду;
- навісний захист використовує реєстр. Наведемо такий приклад. Програма захищена AsProtect і включена функція обмеження роботи (30 днів). Запустивши утиліту RegMon, ми без труднощів знайдемо ключик у реєстрі, видаливши який, одержимо можливість користуватися програмою ще 30 днів і це можна робити як завгодно довго;
- найбільша помилка: автори навісних захистів не стежать за сайтами, на яких представлені утиліти для автоматичного зняття їх захистів. Порада: при виявленні подібних програм досліджувати роботу такої

утиліти, робити висновки і виправляти знайдені помилки, а іноді і заново переписати алгоритм захисту.

Для покращення реалізації захисту даного типу можна *рекомендувати* таке:

- використовувати всі можливі антиналагоджувальні прийоми;
- намагатися не використовувати реєстр Windows для збереження даних про запуск програми;
- використовувати при шифруванні коду програми алгоритми з великою довжиною ключа;
- бажано, щоб під час запуску програми вона не розшифровувалась відразу в пам'ять комп'ютера цілком, а робити це “східчасто”, тобто покроково, у міру необхідності;
- робити більше перевірок контрольної суми і цілісності коду, інтегрувати їх у найбільш важливі функції програми, що захищається.

1.6.6 Захисти за допомогою електронних ключів

До них відносять так звані “dongle”, що найважче піддаються зламу. Якби не дорожнеча даного типу захистів, то, напевно, всі ПЗ захищалося за допомогою “донглів”. Матеріалу про ці засоби захисту у мережі Internet набагато менше, ніж про інші типи захисту. Цей тип захисту успішно знімають тільки “просунуті” і досвідчені “крекери”.

До *недоліків* даного типу захисту відносять таке:

- використання можливостей електронного ключа не на повну потужність;
- перевірка на наявність ключа реалізована 1 раз, зазвичай лише при запуску програми;
- не використана можливість шифрування програмного коду за допомогою так званого “конверта” (envelope).

Рекомендації з покращення захисту даного типу можуть бути такі. Тут найважливіше використовувати всі можливості ключа. Обов'язково робити множинну перевірку на наявність ключа. Обов'язкове використання “конверта”. Також непогано включати перевірку контрольної суми в найбільш важливі функції програми. Використовувати для шифрування даної програми дані з ключа (це є в багатьох сучасних електронних ключах). Також хотілося б відзначити доцільність застосування блокових шифрувальників, вбудованих у більшість існуючих ключів.

1.6.7 Надання розробникам ПЗ засобів SDK

Даний метод захисту не є окремим типом захисту ПЗ, а є лише ще одним способом посилення захисту – це надання розробникам ПЗ так званих SDK (Software Development Kit), за допомогою яких, розробники мо-

жуть ефективно використовувати елементи захисту на етапі розробки ПЗ. Мета захисту – протидіяти всім методам зламу, причому “просунутий” захист повинен захищати не тільки себе, але і модулі того файлу, до якого його приєднали. Останнє можливо в тому випадку, якщо захист має відкритий розширений SDK, що дозволяє розробникам додатків вбудовувати захист у свій модуль вже на етапі розробки власного додатка. Така система може істотно ускладнити життя хакерам. Це життєво необхідно, оскільки один з найефективніших методів захисту – це не просто перевірка інформації окремим модулем, а неявне “розмазування” самої системи захисту на весь програмний продукт. За допомогою визначених механізмів порушується цілісність і правильність виконання програмного продукту (можна згадати, як ще в часи DOS був зламаний пакет програми анімації 3-D Studio 4.0, що ніяк не хотів працювати правильно – результат подібного захисту). Досягти подібного ефекту “розмазування” захисту можна як самотужки, так і силами компанії, що реалізує захист. Останнє більш прийнятне і потрібне, оскільки тільки так можна одержати комплексний підхід у захисті. Тому послуга надання SDK надалі розглядається як особливо важлива для побудови бар'єрів на шляху хакерів.

1.7 Аналіз сучасних програмних продуктів для захисту програмного забезпечення

Наведемо характеристику лише деяких відомих сучасних продуктів із захисту програмного забезпечення.

Тут і надалі будуть вживатися деякі терміни.

Під терміном "*крек*" розуміється зовнішня програма, здатна дезавуювати захист. При даному способі в код захищеної програми вносяться певні зміни.

Емуляція – даний вид програм емулює лазерні мітки. При даному підході в код програми, що розкривається, не вносяться зміни.

Побітове копіювання – найбільш розповсюджений спосіб копіювання, зміст якого полягає у використанні спеціальних побітових копіювальників, на зразок CloneCD. Даний тип захисту може працювати як сам по собі, так і разом з "креком".

Сильним захистом можна назвати той, який зовсім не розкритий (поки не розкритий). *Перспективним захистом* можна вважати той, який розкритий, але якимось одним способом. Такий захист має перспективу стати таким, що розкривається важко, якщо його розробники зможуть підсилити той чи інший слабкий блок. І *слабким захистом* будемо вважати той, який зламаний всіма трьома відомими способами, що говорить про надзвичайно низьку захисну функцію.

CD-COPS

Фірма - виробник: Link Data Security.

Тип захисту: Вимірювання фізичних характеристик без нанесення особливих міток на носій (!).

Спосіб подолання захисту: "Крек".

Апаратна сумісність (cd/dvd різних виробників): Середня (сумісна тільки з популярними приводами).

Наявність особливої апаратури для захисту серії: НЕ потрібно.

Надання користувачу засобів SDK: ТАК.

Захист дрібних партій (CD/R/RW): НЕМАЄ.

Комерційні продукти, що використовують даний вид захисту: Interactive English/De Agostini, Nationalencyklopedin, Agostini Atlas 99, Agostini Basetera, BMM, DK Kort, Lademanns'99.

Особливості захисту: Дана програма користується найефективнішою системою захисту від копіювання, заснованою не на нанесенні фізичних міток, а на способі вимірювання специфічних характеристик CD\DVD-ROM. За словами виробника, система аналізує фізичний кут між минулим і поточними логічними блоками на компакт-диску.

Слабке місце захисту: Сам код, що аналізує кути. Фірмі поки не вдалося знайти ефективного способу протидії дизасемблерам і налагоджувачам.

StarForce

Фірма - виробник: ProtectionTechnology

Тип захисту: Вимірювання фізичних характеристик без нанесення особливих міток на носій (!).

*Спосіб подолання захисту :*Спосіб не знайдений. Справедливості заради варто відзначити, що прецедент розкриття є, але став він можливий тільки завдяки тому, що пірати одержали доступ до незахищеного коду додатка, після чого був зроблений “крек”.

Апаратна сумісність (cd/dvd різних виробників): Висока.

Наявність особливої апаратури для захисту серії: НЕ потрібно.

Надання SDK: ТАК.

Захист дрібних партій (CD/R/RW): ТАК.

Комерційні продукти, що використовують даний вид захисту: 1С (ігри), Нивал, Softmax Co, Q-puncture Inc, Scholastic, Hypnosys World та деякі русифіковані ігри.

Особливості захисту: Дана програма так само, як і CD-COPS, користується самою ефективною системою захисту від копіювання, заснованою на способі вимірювання специфічних характеристик CD\DVD-ROM. Оскільки захист не розкритий, то не відомий і спосіб, яким виробникам вдається ідентифікувати різні диски.

Заявлена 100% сумісність з будь-якою апаратурою і 100% ідентифікація дисків підтверджується незалежними сайтами.

Слабке місце захисту: Слабке місце в захисті знайти не вдалося. Захист ефективно протидіє налагоджувачам і дизасемблерам. Ефективність захисту підтверджує той факт, що за час існування не знайшовся спосіб її нейтралізації. Додатковий диск захисту додає можливість захисту дрібних партій дисків (CD-R\RW) і наявність SDK, за допомогою якого розробники ігор можуть шифрувати окремі сегменти своїх витворів.

Існує декілька варіантів даного програмного продукту.

StarForce Professional 3.0 призначається для видавців і розробників програмного забезпечення і являє собою могутню багаторівневу систему захисту від копіювання програмного забезпечення і файлів даних, розповсюджуваних на дисках CD-ROM і DVD-ROM. StarForce Professional 3.0 Вперше дозволяє реалізувати повнофункціональний захист додатків – захищає не тільки виконувані файли, але і файли використовуваних даних.

StarForce Basic Edition призначена для видавців і розробників програмного забезпечення і є оптимальним інструментом для захисту CD-ROM дисків від домашнього копіювання. StarForce Basic Edition 1.0 є ефективним рішенням для недорогих програмних продуктів, таких як програми для дітей і навчальні програми, що традиційно залишаються без захисту через їх низьку вартість.

StarForce CD-R 3.0 призначається для видавців і розробників ПЗ і є могутньою багаторівневою системою захисту від копіювання програмного забезпечення і файлів даних, розповсюджуваних на дисках, що поставляються компанією Protection Technology. StarForce CD-R 3.0 рекомендується для захисту початкових версій популярних ігор, тестових версій і програмного забезпечення, виробленого невеликими чи поодинокими тиражами. Захищені від копіювання диски з ПЗ записуються за допомогою звичайного пишучого приводу CD-RW або дублікатора CD-R.

StarForce PDF-PRO 1.1 – це система захисту від копіювання і несанкціонованого використання електронних документів у форматі Adobe PDF (Portable Document Format), розповсюджуваних на CD-ROM і DVD-ROM дисках. Захищені документи можуть бути відкриті кінцевим користувачем тільки за умови наявності в CD/DVD-ROM приводі ліцензійного диску. Захищені документи недоступні для перегляду і використання за допомогою інших додатків, що дозволяють працювати з файлами у форматі PDF, крім Adobe Acrobat і Acrobat Reader.

StarForce PDF-CDR 1.1 – це система захисту від копіювання і несанкціонованого використання електронних документів у форматі Adobe PDF, розповсюджуваних на CD-R дисках. StarForce PDF-CDR 1.1 рекомендується для захисту PDF документів, розповсюджуваних невеликими чи поодинокими тиражами. Диски з захищеними документами

записуються за допомогою звичайного пишучого приводу CD-RW чи дублікатора CD-R.

LaserLock

Фірма - виробник: MLS LaserLock International.
Тип захисту: Фізичне нанесення мітки на носій.
Спосіб подолання захисту: Копіювання (BlindRead), "крек", емуляція (D-Tools).
Апаратна сумісність (cd/dvd різних виробників): Низька.
Наявність особливої апаратури для захисту серії: ТАК, потрібно.
Надання SDK: ТАК.
Захист дрібних партій (CD/R/RW): НЕМАЄ
Комерційні продукти, що використовують даний вид захисту: Asghard, Fallout 2, Icewind Dale, Jagged Alliance 2, Messiah, Metro Police, Outcast, Shogo, SpecOps.
Особливості захисту: Використовує унікальне маркування. Кожен додаток на CD має унікальне блокування параметра, що відповідає за кінцевий захист від копіювання.
Слабке місце захисту: Продукт уже розкритий усіма можливими методами. Розробники системи явно не встигають вносити зміни в код захисту для протидії хитрощам піратів.

SafeDisk

Фірма - виробник: Macrovision Corporation.
Тип захисту: Фізичне нанесення мітки на носій
Спосіб подолання захисту: Копіювання (CloneCD), "крек", емуляція (D-Tools).
Апаратна сумісність (cd/dvd різних виробників): Гарна.
Наявність особливої апаратури для захисту серії: ТАК, потрібно.
Надання SDK: ТАК.
Захист дрібних партій (CD/R/RW): НЕМАЄ.
Комерційні продукти, що використовують даний вид захисту: Практично всі ігри після 01-01-2001.
Особливості захисту: Тут також застосовується метод нанесення фізичних міток, для чого потрібно додаткове устаткування. Виробник стверджує про велику ефективність системи, не розкриваючи методів, що застосовуються в захисті. Тільки піратів це не зупинило. Усі програми, захищені даною системою, вже розкриті. Способи протидії захисту також знайдені. Незважаючи на надання SDK компаніям-виробникам ігор, не вдалося забезпечити надійну протидію.

SecuRom

Фірма - виробник: Sony.
Тип захисту: Фізичне нанесення мітки на носій.
Спосіб подолання захисту: "Крек", емуляція (D-Tools).
*Апаратна сумісність (cd/dvd різних виробників):.*Низька.
Наявність особливої апаратури для захисту серії: ТАК, потрібно.
Надання SDK: НЕМАЄ.
Захист дрібних партій (CD/R/RW): НЕМАЄ.
Комерційні продукти, що використовують даний вид захисту: Diablo 2, SimCity 3000, Decent FreeSpace, FIFA 99, Panzer Commander, S.A.G.A: Rage of the Vikings.
Особливості захисту: Використовуються все ті ж мітки. Цікавою відмінністю даного захисту є те, що вони дійсно не копіюються. Принаймні дотепер не знайдено програм, здатних побітово копіювати захищені диски. В іншому ж так само, як і з багатьма іншими системами, – є методи обходу захисту, і це незважаючи на відоме ім'я корпорації, що його виробляє. Справедливості заради варто відзначити, що над удосконалюванням захисту ведуться постійні роботи. Хто знає, може вдасться стати абсолютно непроникними. У всякому разі, мітка, що не копіюється, вже є.

TAGES

Фірма - виробник: Thomson & MPO.
Тип захисту: Вимірювання фізичних характеристик без нанесення особливих міток на носій (!).
Спосіб подолання захисту: Емуляція, "Крек".
Апаратна сумісність (cd/dvd різних виробників): Гарна.
Наявність особливої апаратури для захисту серії: НЕ потрібно.
Надання SDK: НЕМАЄ.
Захист дрібних партій (CD/R/RW): НЕМАЄ.
Комерційні продукти, що використовують даний вид захисту: Moto Racer
Особливості захисту: Захист заснований на методі багаторазового зчитування одного й того самого сектора з наступним порівнянням результатів. Цілком можливо, що тут відбувається аналіз фізичних характеристик носія.
Слабке місце: Його програмний модуль, що вже розкритий.

Dallas Lock

Призначення: Програмно-апаратний комплекс Dallas Lock призначений для захисту від несанкціонованого доступу до ресурсів персонального комп'ютера і локальної обчислювальної мережі, розмежування прав зареєстрованих користувачів щодо доступу до

ресурсів ПК, автоматизованого контролю і ведення протоколу дій щодо доступу до комп'ютера.

Фірма-виробник: Dallas Semiconductor Corp. (США).

Особливості захисту: Як основний засіб ідентифікації користувачів використовуються персональні електронні ідентифікатори iButton (донедавна Touch Memory) і карти proximity, що в сукупності з використанням паролів гарантує підвищену надійність механізму захисту. Високий рівень захищеності гарантований унікальністю коду кожного ключа і надійністю функціонування пристрою.

Контрольні питання

1. Вказати мету і довести доцільність використання СЗПЗ.
2. Дати класифікацію систем захисту ПЗ за методом їх установлення та механізмом захисту, який вони використовують.
3. Дати загальну характеристику основним методам та алгоритмам захисту програмного забезпечення.
4. Охарактеризувати програми-пакувальники та програми-шифратори, навести їх сильні та слабкі сторони.
5. Дати загальну характеристику позитивних та негативних факторів систем захисту від несанкціонованого копіювання.
6. Навести перелік показників, які використовуються для розробки систем захисту програмного забезпечення.
7. Навести та дати пояснення основним критеріям, за якими оцінюються системи захисту програмного забезпечення.
8. Навести основні вимоги до розробки систем захисту ПЗ.
9. Охарактеризувати такі типи захистів, як демо-версії та обмеження часу роботи програми. Навести їх недоліки та рекомендації щодо покращення даних типів захисту.
10. Дати характеристику недоліків захистів за допомогою генерації серійного номера, використання зовнішнього ключового файлу та захисту на основі використання унікальних ідентифікаторів носія.
11. Навести основні характеристики навісних захистів, їх переваг, недоліків та способів покращення захистів цього типу.
12. Охарактеризувати основні недоліки захисту ПЗ за допомогою електронних ключів та надання розробникам SDK і способи їх подолання.
13. Дати порівняльну характеристику сучасним продуктам із захисту ПЗ з точки зору типу захисту, особливостей захисту.

2 СУЧАСНИЙ СТАН ЗАСОБІВ ПОДОЛАННЯ СИСТЕМ ЗАХИСТУ

2.1 Проблема існування засобів зламу захистів ПЗ

Проблеми захисту авторських прав на програмне забезпечення в області контролю над його використанням і подальшим поширенням у даний час прийнято вирішувати за допомогою систем захисту ПЗ. У той же час для обходу і відключення подібних систем захисту існує безліч інструментальних засобів. Виникає задача зіставити можливості засобів захисту ПЗ (СЗПЗ) з можливостями засобів для їх подолання. Результати такого аналізу будуть корисні для оцінювання ризиків при виробництві програмних продуктів, а так само плануванні й оцінюванні рівня стійкості систем захисту ПЗ.

У рамках дослідження проблем захисту програмного забезпечення можна розглядати три таких глобальних питання.

1. *Що захищати?* Це питання пов'язане з класифікацією ПЗ й оцінюванням можливостей захисту ПЗ.
2. *Як захищати?* Це питання пов'язане з аналізом і класифікацією способів і засобів захисту ПЗ.
3. *Від чого захищати?* Це питання пов'язане з аналізом і класифікацією загроз ПЗ і засобів їхньої реалізації.

Перші два питання на даний час освітлені хоча б частково, а третє питання є серйозна "біла пляма" в області досліджень у сфері захисту ПЗ, хоча дослідження з цієї теми проводяться досить активно. Без чіткого розуміння загроз безпеці ПЗ і можливостей засобів реалізації подібних загроз складно взагалі говорити про ефективний захист програмного забезпечення, що підтверджується існуючою практикою в області захисту інформаційних систем. Отже, для серйозного дослідження питань захисту ПЗ необхідно здійснити аналіз і класифікацію засобів реалізації атак на програмне забезпечення.

Слід зазначити, що лише мала частина з розглянутих типів програмного забезпечення є специфічними програмними засобами, призначеними для несанкціонованого відключення систем захисту ПЗ. Більшість же зазначених засобів відноситься до системного програмного забезпечення і розглядається як "інструментарій зловмисника" у силу подвійності технологій. Тобто, те, що звичайні користувачі та програмісти використовують для ефективної роботи, хакери використовують для того, щоб подолати захист або нелегально скористатись програмними продуктами.

В цілому, всі захисти можна розділити на два великі класи:

а) ті, в яких захисні процедури впроваджуються в програми на етапі її розробки;

б) навісні захисти – коли програма упаковується в так званий "конверт" - спеціальну захисну програму.

Крім того зустрічається комбінований метод – коли в самій програмі є захисні процедури, і для того, щоб було складніше залізти усередину, застосовують ще й зовнішні програми. Практично завжди в цих програмах присутні антиналагоджувальні прийоми, іноді вони також упаковують програму, що захищається. Таких програм безліч: MegaShield, Anti-Lame, Pcrypt, Trap, Gardian Angel, SCRAM!, LockProg, ExeLock, USCC, MSCC, UnPackStop, HackStop, CrackStop, ProtEXE, Mask, Mess, XcomOR, Scrypt!, Cn, Spirit, AsPack, AsProtect, Petite, Shrinker, ...

У залежності від типу захисту існують і різні методи зламу захищених програм. Програми з комбінованим захистом зручніше перед вивченням розпакувати для того, щоб можна було внести в програму необхідні зміни. Якщо розпакування за якихось причин неможливе, то можна скористатися спеціальною програмою – run-time patcher'ом – у захищеній програмі за допомогою налагоджувача знаходиться ділянка, яку необхідно відкорегувати, знайдені дані вносяться в патчер, потім з його допомогою запускається вихідна програма і патчер виправляє в запущеній і розпакованій програмі потрібні байти.

2.2 Класифікація засобів подолання СЗПЗ

Наведена нижче класифікація є неформальною класифікацією засобів подолання СЗПЗ, вона ґрунтується на традиційно сформованому поділі програмних засобів у предметній області, і в силу цього є, можливо, не зовсім повною і частково суперечливою. Пропонується класифікація за функціональними ознаками, і в її рамках засоби подолання захисту упорядковані за ступенем складності і стадії аналізу досліджуваного ПЗ.

2.2.1 Програмні оболонки операційних систем

Програми-каталогізатори або файлові оболонки ОС

У стандартні можливості таких програм входять функції перегляду атрибутів файлів (тип, дата створення/модифікації, розмір, прапорці доступу й ін.), підрахунку їхньої кількості і загального обсягу в каталозі додатку, перегляду файлів і т.п. За допомогою цього типу програмних засобів, як правило, реалізується *попередній аналіз захищених продуктів і первинна локалізація СЗПЗ.*

Прикладом подібного використання може бути порівняння дат створення усіх файлів у каталозі встановленого додатка (або у системному каталозі). У випадку використання системою захисту якихось динамічних бібліотек різниця в датах їх створення дозволить легко локалізувати

файли, що відносяться до СЗПЗ (як правило, дати створення "робочих" файлів пакета збігаються, дата створення модулів СЗПЗ відрізняється від них, оскільки СЗПЗ часто поставляються окремо як зовнішня бібліотека).

Аналогічним же чином локалізуються і файли, що зберігають лічильники кількості запусків ПЗ, дати цих файлів постійно обновляються. А за допомогою звичайного текстового перегляду об'єктного модуля можна досить легко визначити тип і виробника СЗПЗ, оскільки зазвичай ця інформація включається в тіло захищеного модуля самої СЗПЗ.

Програми пошуку файлів і текстових та двійкових послідовностей у текстових та двійкових файлах.

Даний тип програм дозволяє здійснювати пошук заданої послідовності (маски пошуку) в одному чи відразу декількох файлах з видачею результатів у вигляді списку зсувів відносно початку файлу, за якими був знайдений шуканий фрагмент; а також усіх файлів, що задовольняють визначеному критерію або містять вищеописану послідовність. За допомогою зазначених засобів *реалізується вторинний аналіз СЗПЗ і локалізація ключових фрагментів СЗПЗ*.

Звичайно засоби файлового пошуку використовуються для:

- пошуку відомих сигнатур СЗПЗ в об'єктних модулях;
- пошуку рядків з повідомленнями СЗПЗ (наприклад, "Програма не зареєстрована!" чи "Спасибі за реєстрацію!"),
- пошуку файлів СЗПЗ з відомими іменами/сигнатурами.

Перший і останній види використання розглянутого типу ПЗ орієнтовані на пошук стандартних елементів СЗПЗ, досліджених раніше й адаптації "типових рішень" до досліджуваної версії СЗПЗ. Другий вид використання пошукових програм орієнтований на локалізацію процедур СЗПЗ, що відповідають за ідентифікацію й аутентифікацію легального користувача ПЗ.

Більшість СЗПЗ реалізує в процесі своєї роботи діалог з користувачем (як мінімум на рівні повідомлень про помилки), локалізація елементів цього діалогу дозволяє досить легко локалізувати "ядро" СЗПЗ, а іноді навіть визначити пароль легального користувача.

2.2.2 Програми-монітори

Програми-монітори файлової системи (File Monitors)

Даний тип програмних продуктів дозволяє відслідковувати зміни, що відбуваються у файловій системі під час запуску певних програм. У більшості таких програм передбачена система фільтрів для формування

протоколів роботи окремих додатків. За допомогою даного типу засобів реалізується *аналіз роботи СЗПЗ з файлами.*

Наприклад, подібні програми дозволяють з'ясувати, що саме і де змінюють розпізнані на етапах первинного і вторинного аналізу модулі СЗПЗ, або визначити модуль, що робить зміни у певному файлі. Ця інформація дозволяє точно локалізувати лічильники кількості запусків ПЗ, приховані файли систем "прив'язки" ПЗ, "ключові файли", файли з інформацією про функції ПЗ, дозволені для використання в рамках даної ліцензії на продукт і т.п., а також модулі і конкретні процедури СЗПЗ, що працюють з цими даними.

Програми-монітори системних файлів ОС (Registry Monitors)

Програмні засоби цього типу призначені для відстеження змін, внесених додатками в конфігураційні файли ОС. У розглянутому контексті дані програми дозволяють реалізувати *аналіз роботи СЗПЗ із системними файлами* (більш специфічно для ОС сімейства Windows).

Розглянуті засоби дозволяють визначати, чи працює система захисту з файлами конфігурації ОС, які зміни вона туди вносить і які дані використовує. У результаті подібного аналізу стає можливим знайти приховані лічильники кількості запусків ПЗ, збережені дати першої установлення ПЗ на ЕОМ користувача, записи з ліцензійними обмеженнями функціональності ПЗ і т.п. Такий аналіз дає результати, подібні до результатів аналізу роботи СЗПЗ з файлами.

Програми-монітори викликів підпрограм ОС (API Monitors)

Програмне забезпечення цього типу призначено для відстеження виклику системних функцій одним чи декількома додатками з можливістю фільтрації/виділення груп системних функцій, що відслідковуються, або додатків. Застосування таких програм дозволяє проводити *аналіз використання системами захисту ПЗ системних функцій.*

З огляду на те, що всі дії ПЗ (і СЗПЗ), зв'язані з роботою з файловою системою, роботою з конфігурацією ОС, реалізацією діалогу з користувачем, роботою з мережею та іншим, реалізуються за допомогою викликів функцій ОС, аналіз використання СЗПЗ системних функцій дозволяє досить докладно вивчити механізми роботи систем захисту, знайти в них слабкі місця і розробити шляхи обходу вживаного захисту.

Наприклад, практично всі сучасні системи захисту від копіювання оптичних дисків базуються на досить невеликому наборі системних функцій при роботі з даним видом накопичувачів інформації, відстеження цих функцій дозволяє знайти і нейтралізувати механізми перевірки типу носія усередині СЗПЗ.

Програми-монітори обміну даними із системними пристроями (портами) (Port Monitors)

У сучасній архітектурі ОС доступ до всіх системних пристроїв (їх контролерів) здійснюється через так звані "порти введення/виведення". Усі сучасні ОС віртуалізують ці порти, організовуючи в такий спосіб спільний доступ декількох додатків до одного й того самого порту (використовуючи механізм черг), а також здійснюючи контроль доступу до портів з метою забезпечення безпеки ОС. Використання цього типу програмних засобів дозволяє проводити *аналіз взаємодії СЗПЗ із системними пристроями.* Контролюючи доступ і обмін даними через порти введення/виведення програмної й апаратної частин СЗПЗ, можна аналізувати і переборювати механізми таких типів захистів, як СЗПЗ з електронними ключами, СЗПЗ з ключовими дисками і СЗПЗ "прив'язки" до ЕОМ користувача.

Програми-монітори мережного обміну даними (Network Traffic Monitors)

Даний тип програм призначений для відстеження мережної активності додатків у рамках ОС. Як правило, такі програми дозволяють фільтрувати/виділяти додатки чи мережні з'єднання за введеними критеріями. Таким чином, використання мережних моніторів дозволяє проводити *аналіз мережного обміну СЗПЗ.*

Цілий ряд сучасних програмних продуктів реалізує перевірку аутентичності користувача шляхом запиту даних про стан ліцензії для даної робочої станції з "сервера ліцензій" у локальних обчислювальних мережах. Також останнім часом з'явилися програмні продукти, що перевіряють аутентичність користувача або термін свого використання через Internet. Крім зазначених видів ПЗ, існують також умовно безкоштовні програмні продукти, у яких тимчасові чи функціональні обмеження замінені обов'язковим переглядом рекламної інформації, одержуваної через Internet. Відслідковуючи мережний обмін подібних програмних продуктів, можна аналізувати механізми систем їх захисту.

Програми-монітори активних задач, процесів, потоків і вікон (Process/Windows Managers)

Зазначений тип програмних засобів призначений для відстеження і керування об'єктами ОС (задачами, процесами, потоками, вікнами й ін.). Подібні програми звичайно надають можливості пошуку необхідного об'єкта ОС, переключення на нього керування, зміни його пріоритету, знищення об'єкта, збереження його параметрів (а іноді і вмісту) на диску.

Застосування моніторів задач дає можливість здійснювати *аналіз модульної структури СЗПЗ*. Подібний аналіз дозволяє з'ясувати подробиці організації СЗПЗ під час її роботи. Список бібліотек, що завантажуються динамічно процесом, дані про кількість створюваних і знищуваних додатком потоків і вікон, поводження програмних продуктів при спробі примусово завершити процес, що містить СЗПЗ, дозволяють істотно доповнити картину аналізу функціонування системи захисту.

Програми-монітори конвеєрів даних, системних повідомлень і високорівневих міжпрограмних взаємодій (Message/COM hooks)

Засоби даного типу призначені для відстеження повідомлень, даних і викликів підпрограм, якими обмінюються об'єкти ОС. Застосування моніторів повідомлень дозволяє здійснювати *аналіз міжмодульної взаємодії в рамках СЗПЗ* (більш специфічно для ОС сімейства Windows).

У результаті такого аналізу виходить інформація про протоколи обміну даними між різними частинами системи захисту, умовах її спрацьовування і відключення, динаміці функціонування захисту.

Програми перехоплення і протоколювання клавіатурного введення (Keyboard Loggers)

Даний тип програмних засобів призначений для збереження інформації, введеної в ЕОМ із клавіатури в спеціальні файли протоколу, можлива фільтрація даних, що зберігаються, за критеріями, що додатково вводяться. "Клавіатурні шпигуни" ніяк не пов'язані з аналізом СЗПЗ, але надають можливість здійснити *незаконне одержання ключа реєстрації/ пароля до програмного продукту, захищеного парольною СЗПЗ.*

2.2.3 Сервісні програми операційних систем

Прикладом цих програм може слугувати Norton Utilites, яку хакери використовують для перегляду/редагування жорсткого диска, для перегляду файлів у різних форматах та для редагування.

Програми копіювання областей ОЗП у ЗЗП (Memory Dumpers)

Зазначений тип програмних засобів призначений для збереження областей оперативної пам'яті, у тому числі пам'яті виконуваних програм, на диск. У рамках дослідження СЗПЗ дані засоби дозволяють здійснити *примусове збереження образу пам'яті захищеного додатка*.

У випадку використання механізмів шифрування/пакування об'єктного коду ПЗ збереження пам'яті активного процесу ОС дозволяє одержати

копію (незначно модифікованого завантажувачем ОС) коду захищуваного ПЗ у "відкритому вигляді". В результаті таких дій можливо або відразу одержати екземпляр незахищеного програмного продукту, або одержати інформацію, важливу для подальшого аналізу СЗПЗ.

Дуже велике число "захищених" програмних продуктів – це упаковані і/або зашифровані об'єктні модулі без якого-небудь серйозного внутрішнього алгоритмічного захисту ПЗ.

Останнім часом з'явилася безліч програм, що знімають навісний захист методом зняття з пам'яті (Cup386, Intruder, IceDump, ProcDump). Принцип тут дуже простий. Після того як захист відпрацював і дав "добро" на виконання, у пам'яті знаходиться початкова програма в такому ж вигляді, начебто її запустили звичайним чином, без усяких захистів. Якщо в цей момент вміст ОЗП записати на диск, то з отриманого зліпка пам’яті можна витягти первісну програму. У кращому випадку виходить працездатний EXE-файл практично ідентичний первісній програмі (правда, для цього процес одержання зліпка пам’яті потрібно буде повторити два рази, для знаходження елементів таблиці переміщень – Relocation Table). Якщо ж так не поталанить, то все рівно буде файл – образ ОЗП, який потрібно лише розмістити в пам'ять на те ж саме місце.

Програми відновлення вилучених файлів (Unerase/Undelete Utilities)

Подібні програмні засоби призначені для відновлення файлів, що були (помилково) вилучені з доступної користувачу області файлової системи ОС і не були ще перезаписані новими даними. Застосування програм такого типу до СЗПЗ дозволяє примусово відновлювати тимчасові файли СЗПЗ, використані ними в процесі роботи; відновлювати в повному обсязі розпаковані і частково вилучені дистрибутиви ПЗ і тимчасові файли ОС. Так реалізується *повторне використання об'єктів СЗПЗ*. Наприклад, ряд СЗПЗ робить розпаковку/дешифрацію об'єктних модулів ПЗ в спеціальні тимчасові файли, що потім запускаються на виконання, а після відпрацювання стираються. Відновлення таких файлів дозволяє перебороти захист.

Засоби ОС за контролем доступу до програм і даних (Access Rights Managers)

Засоби забезпечення контролю і розподілу доступу до даних і додатків є однією з основних частин системи безпеки ОС. Даний тип програмних засобів, як правило, ґрунтується на так званій "матриці доступу", створюваній адміністратором системи. Ця матриця містить права на доступ до системних ресурсів різних категорій користувачів і прикладних програм (тобто користувач трактується як один із процесів ОС).

Застосування подібних засобів до захищеного ПЗ реалізує *системний моніторинг СЗПЗ*. Зокрема, в ОС Windows NT, наприклад, можливе блокування доступу до файлів з даними СЗПЗ або доступу до файлів системної конфігурації (ключів реєстру), блокування створених СЗПЗ тимчасових файлів і т.п.

Програми побайтового копіювання гнучких, жорстких магнітних та оптичних дисків

До них можна віднести такі копіювальники:

- Floppy Disk Analizer by "Мединком";
- CopyWrite by Quaid Software – obsolete;
- CopyIIPC by Central Point Software – obsolete;
- AT Copymaster by Serge S. Pachkovsky;
- Teledisk by Sydex - нескладний, але швидкий копіювальник;
- Byte Copiers, CD Rippers – для копіювання оптичних дисків.

Даний тип програмних засобів призначений для створення максимально точних копій фізичної структури носіїв даних без обліку їх логічної структури. Звичайно подібні програми надають можливість збереження таких зліпків у вигляді файлів на диску. За допомогою наведеного типу засобів реалізується *подолання СЗПЗ від копіювання, СЗПЗ з ключовими дисками і СЗПЗ з прив'язкою до комп'ютерної системи користувача* (до жорсткого диска). В даний момент надзвичайно популярними є СЗПЗ від копіювання для комп'ютерних ігор, розповсюджуваних на оптичних дисках формату CD і Sony PS. Не меншою популярністю користуються і засоби побайтового копіювання оптичних дисків (для створення "піратських" дисків) і засобу збереження вмісту оптичних дисків у вигляді файлів на жорстких дисках (для одержання можливості використання ПЗ, не займаючи накопичувач).

В результаті використання програм-копіювальників хакери здійснюють *копіювання ключових дискет* – один з найшвидших способів одержати нелегальну копію захищеної програми. Тільки надзвичайно складна структура доріжки (мало шансів зустріти таку) може дещо затягти копіювання. Хоча і існують способи дуже ускладнити цю задачу хакерові.

Крім того, хакери використовують ці програмні продукти для *зняття програм з вінчестера*. Припустимо, є пакет, який можна інсталювати на жорсткий диск і який можна потім зняти з вінчестера зі збільшенням лічильника інсталяцій. У такому випадку діє простий *алгоритм одержання нелегальної копії:*

- установити програму на вінчестер;
- зробити копію вінчестера (всього – на стриммер або ж тільки зайнятих секторів), включаючи FAT, кореневий каталог та інші службові області, якщо потрібно;

- зняти програму з вінчестера з відновленням лічильника інсталяцій;
- відновити області вінчестера, що запам’ятовувались.

Ось і все! Тепер при незміненому лічильнику інсталяцій на вінчестері є працездатна копія програми.

Засоби пакетної обробки команд (Batch Processors/Script Engines)

Даний тип програмних засобів дозволяє виконувати (послідовно або паралельно) відразу цілий набір команд, попередньо заданий користувачем. У рамках пакетів завдань підтримуються оператори циклу і розгалуження. Подібні засоби дозволяють здійснювати *створення віртуального оточення на час роботи СЗПЗ*.

Умовно безкоштовні програмні продукти доступні для використання ще до їх придбання як такого. Зазвичай такі продукти містять обмеження за часом свого використання, за кількістю запусків або функціональному наповненню. За допомогою засобів пакетної обробки команд можливе створення необхідного програмного оточення (установлення системної дати, зміна файлів даних СЗПЗ, зміна параметрів ОС і ін.) до запуску захищеного ПЗ з можливістю повернення до попереднього стану середовища після закінчення роботи програмного продукту.

2.2.4 Програми статичного та динамічного аналізу

Засоби дизасембювання об'єктних модулів ПЗ (Disassemblers)

Програми цього типу (IDA, WinDasm, Sourcer, hiew) призначені для "детрансляції" об'єктних модулів з машинного коду в мнемокод асемблера. При застосуванні засобів дизасембювання здійснюється *статичний аналіз алгоритмів СЗПЗ за мнемокодом.*

Одержання доступу до мнемокоду СЗПЗ дає чудову можливість детального аналізу програмного й алгоритмічного виконання процедур СЗПЗ, а також перебування конкретних шляхів обходу або модифікації ключових фрагментів СЗПЗ. Іноді з'являється можливість використання елементів СЗПЗ у наново створюваних засобах їх подолання.

Засоби декомпіляції об'єктних модулів ПЗ (Decompilers)

Програми-декомпілятори подібні дизасемблерам і навіть іноді їх використовують у якості підпрограм до власних програм. Задачею, що стоїть перед даним типом програмних засобів, є "детрансляція" об'єктних модулів з машинного коду у вихідний код мовою високого рівня. Застосування декомпіляторів до досліджуваного програмного забезпечення реалізує *статичний аналіз алгоритмів СЗПЗ за початковим кодом.*

Більшість існуючих сучасних декомпіляторів орієнтована на обробку об'єктних модулів, написаних мовами інтерпретуючого типу (FoxPro, Clipper, Visual Basic, Java), декомпілятори для мов компілювального типу зустрічаються вкрай рідко і мають обмежені можливості в силу технічних особливостей процесу компіляції.

Декомпіляція ПЗ дає доступ до його вихідного коду (чи його еквіваленту) і дозволяє цілком розпоряджатися програмним продуктом, включаючи внесення в нього функціональних змін і повторну компіляцію.

Засоби налагодження об'єктних модулів (Debuggers)

До складу стандартних функцій налагоджувачів входять можливості покрокового виконання об'єктного коду, встановлення точок зупину (у тому числі тих, що спрацьовують при виконанні певної умови), перегляду об'єктного коду ПЗ в дизасембльованому вигляді, зміни послідовності виконання об'єктного коду, редагування пам'яті налагоджуваного процесу, відстеження зміни даних процесу та ін. (наприклад, програми SoftIce, TD, Deglucker).

У рамках дослідження СЗПЗ використання налагоджувачів реалізує *динамічний аналіз алгоритмів СЗПЗ.* Подолання практично будь-якої СЗПЗ в більшості випадків неможливе без використання налагоджувальних засобів. При цьому велика частина налагоджувальних функцій реалізується в архітектурі центрального процесора ЕОМ.

На додаток до звичайних налагоджувачів існує особливий клас програм, названих *емулюючі налагоджувачі.* Вони не намагаються коректно виконувати трасування працюючої програми, що, до того ж, активно цьому “пручається”, а самі інтерпретують і виконують її машинні інструкції (наприклад, замість MOV AX, 56 вони привласнюють змінній, що відповідає регістру AX - скажімо, Reg_AX - число 56). Існують також налагоджувачі з неповною емуляцією, що емулюють тільки "небезпечні" команди, а інші виконують на реальному процесорі.

Такі налагоджувачі нейтралізують практично всі методи протидії налагодженню (блокування переривань і пристроїв, робота з контролерами через порти, підрахунок контрольних сум) для виявлення контрольних точок, контролю стека, а також, методи, засновані на особливостях процесора і ДОС. Плюс до всього, ці налагоджувачі мають просто фантастичні можливості для налагодження: можна поставити контрольну точку для запису/зчитуванню або ще для будь-чого на будь-яку область пам'яті чи ряд портів введення/виведення, на невідповідність конвеєра команд і ОЗП (!), і т.д. і т.п.

Єдине, що (поки що) можна їм протиставити – це часові вимірювання, “паралельні процеси” і т.д., але і це доти, поки не з'явиться емулятор мікросхеми таймера - 8254 - здатний синхронно працювати з

емулятором процесора. При додаванні емуляторів інших пристроїв вийде віртуальна машина, на якій повного проходження захисту, скоріше усього, уникнути не удасться, і можна буде тільки тією чи іншою мірою утруднити процес усвідомлення отриманих результатів людиною.

2.2.5 Програми статичної та динамічної модифікації

Засоби пошуку і заміни текстових і двійкових послідовностей у текстових та двійкових файлах (Patchers/Hex-editors)

Функціонально такі програмні засоби призначені для оперативного і простого внесення бажаних змін в один файл або групу файлів. Багато подібних засобів можуть здійснювати пошук за заданою маскою.

За допомогою таких засобів пошуку і заміни здійснюється *статична модифікація коду СЗПЗ*. Як правило, модифікація СЗПЗ з метою позбавлення її функціональності полягає у заміні декількох байтів об'єктного коду. У той же час існують СЗПЗ, для дезактивації яких потрібна модифікація великої кількості (іноді непостійних) послідовностей байтів, що стає можливим при використанні цього типу програм.

Засоби редагування "ресурсів" об'єктних модулів (Resource Editors)

Подібні програми використовуються для редагування текстових, діалогових, графічних, аудіо-, відео- і інших ресурсів, що містяться в області даних об'єктних модулів ПЗ. У рамках дослідження СЗПЗ подібні засоби дозволяють робити *редагування ресурсів СЗПЗ.*

Як правило, вміст усіх пунктів меню інтерфейсу програми, всі текстові повідомлення і діалоги, видавані програмою, графічні елементи інтерфейсу й ін. містяться в секції ресурсів області даних об'єктного модуля. Модифікація цих ресурсів дозволяє змінити інтерфейс програми, у тому числі активувати відключені в рамках даної ліцензії на ПЗ пункти меню, запобігти видачі додатком попереджувальних повідомлень про необхідність придбання ПЗ, змінити діалоги і т.п. Іноді в текстових ресурсах містяться паролі/серійні номери захищеного ПЗ.

Засоби завантаження об'єктних модулів і/або їх динамічної модифікації в ОЗП (Loaders/In-memory Patchers)

Цей тип програмних засобів призначений для модифікації пам'яті процесу ОС під час його виконання в ОЗУ. За винятком особливостей модифікації об'єктного коду в оперативній пам'яті, дані засоби функціональ-

но подібні засобам пошуку і заміни у файлах. За допомогою таких програм здійснюється *динамічна модифікація коду СЗПЗ.*

Зазвичай засоби динамічної модифікації коду використовуються при високій складності або нераціональності розпакування/дешифрації об'єктних модулів захищеного ПЗ. Модифікація коду модуля захисту з метою *обійти перевірку чи спотворити її результати* – один з найбільш розповсюджених методів, що дає найзручніші у використанні програми. Але він також і є самим трудомістким і тривалим.

Засоби завантаження і/або модифікації контексту об'єктних модулів у регістрах ЦП

Цей тип програмних засобів призначений для зміни стану регістрів центрального процесора ЕОМ під час виконання об'єктного модуля ПЗ. Дані засоби реалізують *зміну контексту процесу виконуваної СЗПЗ*.

За допомогою подібних засобів можливо впливати на процес виконання об'єктного коду, не вносячи в нього змін. З юридичної точки зору, подібне керування центральним процесором не порушує (та й не може порушити) ніяких законодавчих норм, оскільки воно ніяк не стосується об'єктного коду захищеного ПЗ, у той же час зовсім аналогічні дії є складовою частиною роботи ОС.

2.2.6 Програми-емулятори

Програми емуляції апаратних засобів (ГМД, ОД, електронних ключів, локальних мереж) (FDD/ CD/LPT/Network Emulators)

Програми емуляції апаратних засобів призначені для створення віртуальних пристроїв, необхідних для функціонування ПЗ типів, а також забезпечення доступу до них як до реальної апаратури.

Застосування подібного ПЗ до захищених програмних продуктів уможливлює *подолання: СЗПЗ від копіювання, СЗПЗ з електронними ключами, з ключовими дисками, з прив'язкою до ЕОМ користувача, з ключовими файлами, парольних СЗПЗ з авторизацією через мережу*. Використання даного типу ПЗ також є цілком легальним.

Засоби емуляції центрального процесора і підпрограм ОС (CPU/API Emulators)

Програми зазначеного типу (TRW, Cup386) роблять віртуалізацію центрального процесора ЕОМ і/або певних функцій ОС на час виконання одного або групи додатків. Дані засоби реалізують *зміну процесу діючої СЗПЗ*. Симулятори процесора або сервісів ОС роблять попередній аналіз

інструкцій процесора або викликів функцій операційної системи і потім обробляють і виконують (чи ігнорують) їх відповідно до закладених заздалегідь правил.

Засоби емуляції операційних систем чи ЕОМ цілком (OS/PC/Mac/... Emulators)

ПЗ цього типу призначено для забезпечення виконання додатків, створених для однієї програмної/апаратної платформи, на іншій платформі. Велика частина подібних програм надає також і налагоджувальні можливості (порівнянні з можливостями апаратного налагодження, а іноді і переважаючи їх за функціональністю).

Застосування зазначеного типу програмних засобів до захищеного ПЗ дозволяє здійснювати *подолання СЗПЗ довільного типу*.

Емулятори ОС здійснюють динамічний аналіз викликів системних функцій працюючого додатка, їх конверсію у виклики поточної ОС і зворотну конверсію кодів повернення в коди емулюємої ОС. Емулятори процесорів роблять те саме, але на рівні машинного коду процесорів.

Незважаючи на зовсім "мирні" цілі, що полягають у забезпеченні переносимості додатків між різними платформами, нестандартне використання засобів цього типу дозволяє переборювати не тільки системи захисту програмного забезпечення, але і схеми "керування цифровими правами" (Digital Rights Management) на доступ до авторських витворів, заснованих на прив'язці до ЕОМ користувача.

2.2.7 Апаратні засоби, що полегшують задачу зламу СЗПЗ

Щодо апаратних засобів, то, в принципі, кожну (!) дискету з неушкодженою поверхнею можна скопіювати з використанням *апаратури для аналогового копіювання дискет*. Хоча така апаратура, звичайно, екзотика. Непогані характеристики для цього має плата Copy II PC Option Board Deluxe. Що ж стосується налагодження, то існують апаратні налагоджувачі, і навіть програмні можуть використовувати додаткове hardware (наприклад, це може Periscope).

Для зламу захисту використовують і метод *спарювання ПЕВМ.* Для цього використовують апаратні заглушки на COM-порти, через які до ПЕВМ можна підключити другу ПЕОМ, для відстеження переданих у заглушку сигналів.

2.3 Програми розпакування, дешифрування та криптоаналізу

Програми-розпакувальники/дешифратори (Unpackers/Decryptors)

Засоби розпакування/дешифрування об'єктних модулів дозволяють одержувати копії зазначених модулів у тому вигляді (або близькому до такого), у якому вони були до їх упакування/шифруання. За функціональним змістом ці програми близькі до засобів збереження областей ОЗП на диску, але даний тип програмних засобів, по-перше, відрізняється високою спеціалізацією (тобто спрямованістю на якийсь один тип або клас засобів пакування/шифрування), а, по-друге, не завжди вимагає завантаження об'єктного модуля в ОЗП як процесу ОС.

При використанні подібних програм реалізується *подолання СЗПЗ пакувального чи шифрувальног типу*. Як правило, розпакування/дешифрування захищених об'єктних модулів ПЗ здійснюється для одержання можливості більш глибокого подальшого аналізу СЗПЗ.

Засоби криптоаналізу (Password Crackers/Bruteforcers)

Зазначений тип програмних засобів призначений для аналізу і подолання систем криптографічного закриття інформації. Звичайно в них реалізується кілька видів атак на шифри: атака з використанням відомого відкритого тексту, що вичерпує перебір, спрямований перебір з евристикою, перебір за словником. Використовуючи подібні засоби, можна здійснити *криптоаналіз СЗПЗ із шифруванням та парольних СЗПЗ.*

Оскільки об'єктні модулі ПЗ складаються з інструкцій машинного коду і послідовність таких інструкцій іноді можна вгадати, у ряді випадків можливо зробити атаку по відомому відкритому тексті, а також ряд інших атак для подолання СЗПЗ, що використовують методи шифрування.

Засоби генерації паролів і серійних ключів (Key Generators)

Засоби подібного типу використовуються для генерації ключових послідовностей, що задовольняють критеріям використовуваних у СЗПЗ криптоалгоритмів. Зазначений тип засобів реалізує *подолання парольних СЗПЗ, а також СЗПЗ з електронними ключами і ключовими файлами*.

Програми-генератори різних ключів, кодів повернення і т.п., як правило, є результатом попередньо проведеного криптоаналізу СЗПЗ і дозволяють одержувати "придатні" до СЗПЗ значення ключів для "легального" відключення СЗПЗ.

2.4 Висновки

Як уже було зазначено вище, практично всі перераховані програмні засоби відносяться до звичайного системного або програмного забезпечення користувачів. До "незаконних" засобів можна частково віднести лише засоби протоколювання клавіатурного введення, засоби статичної модифікації файлів і засоби генерації серійних номерів ПЗ. У той же час навіть ці типи програмних засобів здатні використовуватися (і реально використовуються) в областях, що ніяк не відносяться до дослідження і подолання СЗПЗ і порушення авторських прав. Засоби протоколювання клавіатурного введення використовуються для обробки натискань комбінацій клавіш у рамках функціонування інтерфейсів для користувачів ПЗ, а також систем комп'ютерного навчання. Крім того, вони можуть використовуватися для архівації всієї інформації, набраної з клавіатури, з метою відновлення загубленої під час збоїв інформації.

Засоби модифікації файлів використовуються у ряді областей, наприклад, для оперативної модифікації власних програмних проектів, службових файлів і ін.

Засоби ж генерації ключових послідовностей можуть легально використовуватися для відновлення загубленої легальним користувачем ключової інформації (у випадку відмовлення з боку власника авторських прав) або в освітніх цілях.

Таким чином, можна стверджувати, що необдумана заборона використання перерахованих типів ПЗ (за аналогією зі шкідливими програмами) буде неефективною мірою, оскільки спричинить за собою серйозні труднощі або повну неможливість використання ПЗ, необхідного для нормального функціонування комп'ютерних систем. Природно, подібна заборона не буде дотримуватися на практиці через його нездійсненність.

Можлива законодавча заборона "нецільового використання" приведених типів ПЗ, але на законодавчому рівні практично неможливо регламентувати "цільові" і "нецільові" види використання ПЗ, що веде до практичної незастосовності (чи високій складності і неоднозначності застосування) подібних законодавчих норм.

З усього перерахованого вище можна зробити висновок, що одні тільки технічні засоби захисту ПЗ, навіть з урахуванням їхньої законодавчої підтримки, не здатні забезпечити належний рівень безпеки програмних продуктів, що захищаються. Отже, необхідний більш комплексний підхід до захисту ПЗ, з обліком багатьох інших аспектів поширення, реалізації і використання програмного забезпечення.

Таким чином, використовуючи ті або інші засоби зняття захистів, будь-яку програму можна зламати. Як кажуть, "If it runs, it can be defeated" - "Якщо програма запускається, то її можна зламати".

Контрольні питання

1. Охарактеризувати загальний стан засобів подолання систем захисту програмного забезпечення.
2. Дати основні характеристики сервісним системним програмам, що можуть бути використані для зняття захистів.
3. Вказати мету використання і основні характеристики файлових оболонок, що використовуються для подолання захистів програм.
4. Охарактеризувати програми-монітори для дослідження систем захисту програмного забезпечення.
5. Охарактеризувати сутність засобів для емуляції та стимуляторів програмного середовища для зняття захисту програмних продуктів.
6. Дати характеристику засобів для статичної та динамічної модифікації програмного середовища.
7. Охарактеризувати засоби, що використовуються для статичного та динамічного дослідження алгоритмів програм.
8. Засоби для розпакування, дешифрування та криптоаналізу систем захисту програмного забезпечення.
9. Навести приклади апаратних засобів для зламу систем захисту.

3 ЗАХИСТ ВІД НЕСАНКЦІОНОВАНОГО КОПІЮВАННЯ

3.1 Методи розповсюдження програмного забезпечення

У світовій практиці існують такі способи поширення програм:

CommercialWare – комерційне програмне забезпечення, яке створене для отримання прибутку. Для нього застосовується принцип "гроші – вперед", тобто користувач отримує програму лише після оплати;

FreeWare – версії програм для вільного розповсюдження зі збереженням прав за автором;

ShareWare – умовно безкоштовне програмне забезпечення, тобто версії програм, які можна 2-4 тижні випробовувати, а потім або не використовувати, або оплатити;

TrialWare – програмне забезпечення, яке можна використовувати на протязі певного терміну, потім продукт стає недієздатним;

CriptWare – програмний продукт має дві версії: демо-версія плюс зашифрована робоча версія;

DemoWare – коли в програмі присутні функціональні обмеження. Наприклад, можна обробляти файли, що не перевищують певного розміру, не можна виконувати їх збереження і т.д. Іноді ці програми ще називають *CrippleWare* – "урізане" програмне забезпечення;

Nagware – користувачу постійно пропонуються нагадування про те, що дана версія програми не є повноцінною комерційною версією (у вигляді діалогового вікна, що з'являється з деякою періодичністю під час роботи, або додаткові надписи, що виводяться на екран, на принтер і т.д.);

AdWare – розробник отримував плату за використання програми не від кінцевого споживача, якому програма надавалася безкоштовно, а від рекламодавців, а користувач повинен був переглядати надані через Інтернет рекламні картинки.

Іноді автори програм з якихось причин не хочуть розповсюджувати їх як комерційні продукти, але не заперечують проти отримання якогось зиску, не враховуючи морального задоволення. І тоді вибирається один з таких методів розповсюдження:

CardWare – кожний користувач програми, що бажає зареєструватися, повинен надіслати автору програми поштову листівку з виглядом місцевості, де він проживає;

MailWare – більш сучасний варіант CardWare, що передбачає відсилання автору електронного листа. Як правило, у відповідь автор надсилає реєстраційний код, що дає можливість працювати з програмою;

DonationWare – коли автор не вимагає ніякої оплати, але пропонує всім, кому сподобалась програма, пожертвувати певну суму для підтримки даної програмної розробки;

GifWare – майже те саме, що і попередній вид розповсюдження, але автор готовий прийняти не лише грошові пожертвування, але й інші подарунки;

BeerWare – подяка за програму приймається у вигляді пива;

VegeWare – автор збирає з користувачів плату за програму у формі рецептів вегетеріанських блюд;

MemorialWare – людина на ім’я Гарі Кремблітт присвятив свою програму пам’яті свого батька. Користувачам цієї програми пропонується допомоги меморіальному фонду Кремблітта-батька.

Більшість програм поширюється за принципом “AS IS” (як є), що є загальноприйнятим у міжнародній комп'ютерній практиці. Це означає, що за проблеми, які виникають у процесі експлуатації програми, розробник і розповсюджувач відповідальності не несуть.

Захист програмного продукту від несанкціонованого копіювання – актуальна задача у зв'язку зі збереженням комерційних і авторських прав фірм і розробників. За висновками закордонних фахівців, економічний збиток від "піратського" копіювання програмного забезпечення складає мільярди доларів. Точні втрати установити неможливо через відсутність повних відомостей про число "піратських" копій, але вважається, що з кожної програми їх робиться від 2 до 15. У Росії та в Україні близько 95% використовуваного софта "піратське", а решта 5% - FreeWare.

3.2 Технології захистів програмного забезпечення

З погляду професійного програміста термін "захист від копіювання" для IBM PC, що працює під керуванням MS DOS/Windows, досить умовний, оскільки практично завжди можна переписати інформацію, що знаходиться на дискеті або на жорсткому диску. Інша справа, що після цього програма може не виконуватися. Таким чином, без санкції розробника чи фірми-розповсюджувача неможливо одержати працездатний програмний продукт. Тобто, фактично, "захист від копіювання" – це створення засобів, що здійснюють можливість "захисту від несанкціонованого виконання".

На сучасному етапі найбільшого розповсюдження набули такі технології захисту від копіювання.

1. Однією з розповсюджених технологій захисту від копіювання є *створення особливо обумовлених дискет*. Їх особливість полягає в тому, що на дискеті створюється спеціально організована мітка, яка використовується як ознака її дистрибутивності. Функцію контролю мітки виконує спеціальна частина програми, що захищається. Після копіювання засобами ОС диска, що захищається, буде скопійована вся інформація, за винятком мітки. Під час виконання програми її контролююча частина встановить, що диск не дистрибутивний, і припинить виконання програми. Тим самим програма мовби

"прив'язується" до своєї дискети. Для створення мітки застосовуються програмні й апаратні засоби, а також можливе їх комбінування.

2. Інший спосіб запобігання незаконного використання програм і даних полягає в *збереженні інформації в кодованому, зашифрованому вигляді.* У цьому випадку без знання ключа робота з інформацією неможлива.
3. Третій спосіб полягає у *використанні ключів*, які підключаються до COM, LPT чи USB портів.

Системи захисту від несанкціонованого копіювання (НСК) можна розділити на такі групи:

- прив’язка до дискети або компакт-диску;
- прив’язка до комп’ютера;
- прив’язка до ключа;
- опитування довідників;
- обмеження використання ПЗ.

Перші три групи методів захисту від копіювання пов’язані з роботою з дисками на фізичному рівні, а для цього необхідно добре знати будову дисків та можливості керування ними.

3.3 Основні поняття ОС, необхідні для створення систем захисту програмного забезпечення

3.3.1 Склад операційної системи

Функції ОС реалізовані у вигляді набору системних файлів, які виконують такі функції:

- *програма початкового завантаження*, яка автоматично зчитується при запуску ЕОМ і керує завантаженням решти ОС;
- *програма файлової служби*, яка підтримує файлову структуру системи і виконує цілий ряд спеціальних функцій;
- *програма керування базовим введенням-виведенням* всіх пристроїв;
- *процесор консольних команд,* який обробляє введені з клавіатури системні команди.
- *драйвери пристроїв* – спеціальні програми, які доповнюють систему введення-виведення і забезпечують обслуговування нових пристроїв або нестандартне виконання наявних пристроїв.

3.3.2 BIOS: Базова система введення-виведення

Апаратні засоби ЕОМ повинні працювати з програмним забезпеченням, тому для них потрібний інтерфейс. BIOS дає ЕОМ невеликий вбудований стартовий набір для виконання іншого програмного забезпечення на гнучких дисках (FDD) і жорстких дисках (HDD). BIOS відповідає за

завантаження ЕОМ, забезпечуючи базовий набір команд. BIOS виконує всі задачі, що повинні виконуватися під час запуску: POST – Power-On-Self-Test, і завантаження системи з FDD чи HDD. Крім того, BIOS забезпечує інтерфейс ОС з використовуваним обладнанням у формі бібліотеки підпрограм обробки переривань. Наприклад, кожний раз при натисканні клавіші CPU виконує переривання для читання її коду. Подібне відбувається і для інших пристроїв введення/виведення (послідовні і паралельні порти, відеокарти, звукові карти, контролери жорсткого диска і т.п.).

3.3.3 CMOS: Complementary Metal Oxide Semiconductor

Для виконання своїх задач BIOS необхідно "знати" різні параметри (конфігурацію апаратних засобів). Ці параметри постійно зберігаються у невеликому фрагменті (64 байти) CMOS-ОЗП. Живлення ОЗП забезпечується невеликою батареєю, тому вміст ОЗП не втрачається після вимикання РС. Отже, на системній платі є батарея і невелика пам'ять, що ніколи (ніколи не повинна!) не втрачає інформацію. Раніше пам'ять розташовувалася у вигляді частини мікросхеми системного годинника, а в даний час вона – частина інтегральної схеми (ІС).

CMOS – назва технології, за якою виготовляються ІС із вкрай малим споживанням енергії, тому батарея в комп'ютері майже "не використовується". Фактично на нових системних платах установлена не батарея, а акумулятор (найчастіше – нікель-кадмійовий). Він підзаряджується щоразу при включенні комп'ютера. Якщо CMOS у комп'ютері живиться від зовнішньої батареї, то слід упевнитись, що вона знаходиться у робочому стані – інакше системна плата може бути ушкоджена. В іншому випадку CMOS може раптово "забути" конфігурацію, а ми будемо шукати помилку зовсім в іншому місці.

3.3.4 Переривання, їх роль і процедура звертання в програмах

Центром обчислювальної системи є процесор, який виконує команди, що входять до складу програми для ЕОМ. Процесор керує основними операціями за допомогою передачі і прийому керуючих сигналів, адрес пам'яті і даних через сукупність ліній зв'язку, які називаються *шинами*. Вздовж шини розміщені вхідні і вихідні порти, які з'єднують з нею різні електронні мікросхеми. Дані від процесора проходять через ці порти до інших частин ЕОМ і навпаки.

Всередині процесора знаходяться регістри, які утворюють робочу область для передачі і обробки даних. Ці внутрішні регістри забезпечують тимчасове зберігання даних, адрес пам'яті, покажчики команд і прапорці стану і керування. За їх допомогою процесор може отримувати доступ до пам'яті об'ємом понад 1 млн. байтів і до всіх портів введення-виведення.

ЕОМ має реагувати на дії, які здійснюються за межами її мікропроцесора (натискання клавіш на клавіатурі, на миші, ...). Є два способи, як це можна зробити.

1. *Постійно опитувати всі частини ЕОМ*. Такий спосіб досить нераціональний – потребує багато процесорного часу.
2. Дозволити процесору виконувати ту чи іншу роботу до тих пір, поки що-небудь зовні не приверне його увагу. Такий підхід дістав назву *роботи з перериваннями*.

Механізм переривання працює таким чином. Кожному виду переривань відповідає номер. У пам'яті ЕОМ зберігається таблиця, що визначає програми, які мають активізуватися при виникненні того чи іншого переривання. Отже, якщо виникає переривання X, то значення покажчика наступної команди запам'ятовується і замінюється значенням, яке визначає першу команду програми обробки переривання X. Коли програма обробки переривання (ПОП) закінчує свою роботу, то вона за допомогою спеціальної команди відновлює значення покажчика наступної команди, тобто повертає керування початковій програмі.

Поняття переривання було створене як спосіб організації взаємодії процесора з зовнішнім світом. Поряд з перериваннями через апаратуру існують так звані логічні і програмні переривання.

Логічні переривання генеруються самим мікропроцесором, коли виникають якісь надзвичайні ситуації (наприклад, спроба ділення на нуль).

Програмні переривання – більш цікавий вид переривань. Коли певна програма потребує використання іншої програми або підпрограми, то вона має передати керування процесором цій програмі. Переривання дозволяє це зробити і є альтернативою використанню команди CALL. Існують дві причини, які пояснюють, чому використання переривань краще за безпосередню адресацію:

- такий спосіб дає можливість змінювати програму, що викликається. Зміна може вплинути на розмір і розміщення в пам'яті; хоча, оскільки ця програма викликається перериванням, то немає необхідності в її зміні;
- можливість заміни програми обробки переривань. Для цього необхідно замінити лише відповідний елемент таблиці переривань.

Звертання до функцій операційної системи і BIOS здійснюється за допомогою програмних переривань (команд INT). Початок оперативної пам'яті, починаючи від адреси 0000h, відводиться під вектори переривань, що являють собою чотирибайтові області, в яких зберігаються адреси ПОП (рис.3.1). В два старших байти кожного вектора записується сегментна адреса ПОП, а в два молодші – відносна адреса точки входу в ПОП у сегменті. Вектори, як і переривання, що їм відповідають, мають номери, які називаються типами. Вектор з номером 0 (вектори типу 0) займає байти пам'яті, починаючи з адреси 0, а вектор з номером N, відповідно, займає

байти пам’яті, починаючи з адреси N×4 до N×4+3. Всього у виділеній під вектори області пам’яті розміщується 256 векторів.

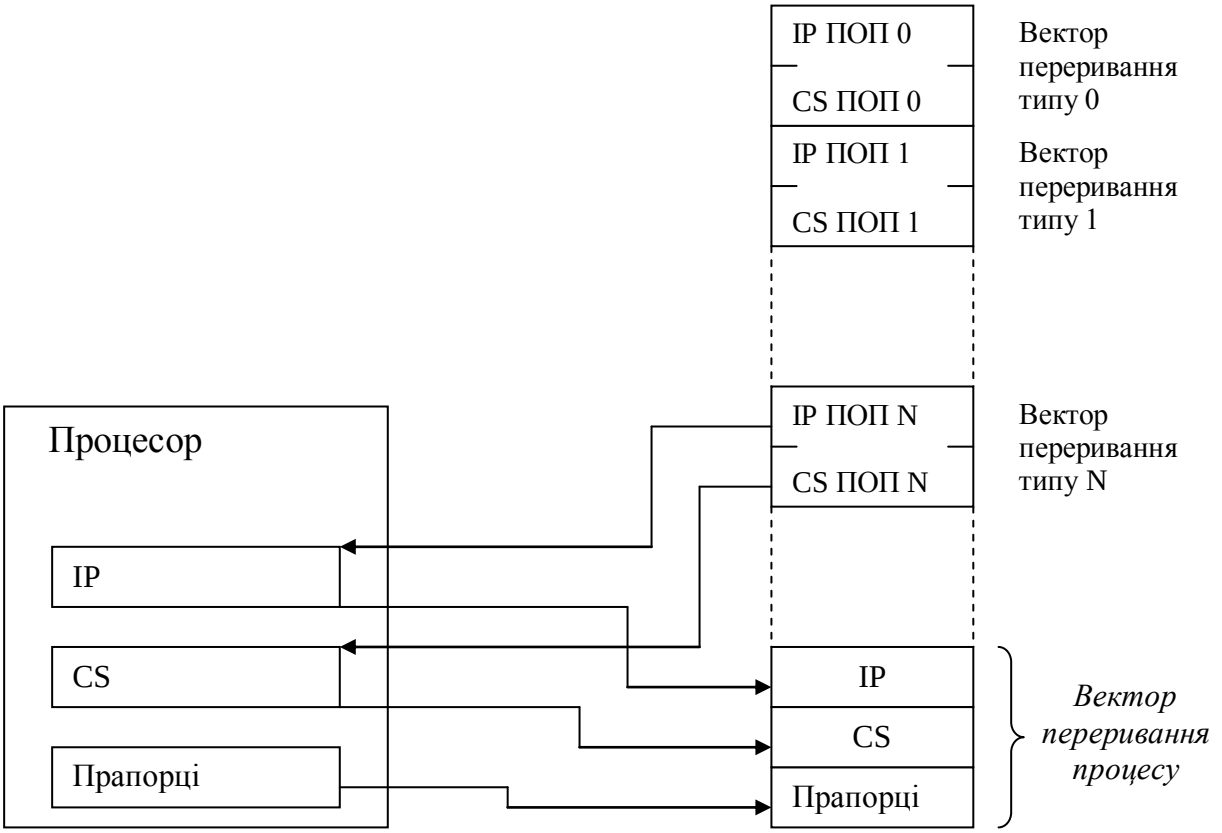


Рисунок 3.1 - Схема процедури переривання

Отримавши сигнал на виконання переривання з певним номером, процесор зберігає в стеку програми, що виконується, слово прапорців, а також сегментну і відносну адреси програмного сегмента (вміст CS та IP) і завантажує CS та IP із відповідного вектора переривань, здійснюючи тим самим перехід на ПОП. ПОП зазвичай закінчується командою повернення із переривання IRET, яка виконує протилежні дії – завантаження CS та IP і регістру прапорців із стека, що призводить до повернення в основну програму у точку, де вона була перервана.

Велика частина векторів переривань зарезервована для виконання певних дій; частина з них автоматично заповнюється адресами системних програм при завантаженні системи.

Запити на виконання процедури переривання можуть мати різну природу, їх умовно можна розділити на такі групи:

- вектори апаратних переривань (08h-0Fh, 70h-77h);
- драйвери BIOS (10h, 13h, 16h, …);
- програми DOS (21h, 22h, 23, …);
- адреси системних таблиць (1Dh, 1Eh, 43h, …).

Системні програми, адреси яких зберігаються у векторах переривань, в більшості своїй є лише диспетчерами, які відкривають доступ до великих груп програм, що реалізують системні функції.

Звертання з прикладної програми до системних функцій здійснюється однаково:

- в регістр AH засилається номер функції (не тип переривання!);
- в інші регістри засилаються початкові дані для виконання конкретної системної функції;
- після цього виконується команда INT з числовим аргументом, який вказує тип (номер) переривання (наприклад, INT 21h).

Більшість цих функцій ставлять у прапорці переносу CF код закінчення. Якщо функція виконана успішно, то CF=0, у разі помилки CF=1.

3.4 Робота з дисками на фізичному рівні

У розпорядженні програміста є засоби різного рівня, призначені для роботи з дисковою системою в середовищі операційної системи.

По-перше, будь-яка програма може *звертатися безпосередньо до апаратури контролера* жорсткого диска або контролера накопичувача на гнучких магнітних дисках. Це найнижчий рівень, застосування якого виправдано далеко не завжди. Більш того, настійно рекомендується не працювати з контролером на рівні портів, якщо ми точно не знаємо, навіщо вам це потрібно. Більшість задач розв'язуються з застосуванням функцій ОС чи BIOS (у тому числі такі нетрадиційні задачі, як захист від несанкціонованого копіювання). Однак іноді програміст буває змушений використовувати самий низький рівень, ризикуючи втратити сумісність з численними типами дискових контролерів.

По-друге, програма може працювати з дисками *за допомогою переривання BIOS INT 13h*, призначеного для виконання операцій зчитування, запису і форматування. Використання функцій переривання INT 13h набагато зручніше, ніж безпосереднє програмування контролера, оскільки BIOS приховує особливості апаратної реалізації контролера.

По-третє, програма може звертатися до *спеціально призначених перериванъ* ОС для доступу до дискової системи. Сервіс, наданий цими перериваннями також можна розділити на низько- і високорівневий, тому програміст завжди має вибір.

Вибираючи засоби звертання до дискової системи, необхідно спочатку розглянути можливість використання найбільш високорівневих документованих засобів таких, як переривання ОС. І лише потім є сенс звернутися до функцій BIOS чи до програмування портів контролера.

Незважаючи на все сказане вище, ми розглянемо засоби роботи з дисковою системою, дотримуючи зворотного порядку. Спочатку розглянемо використання найбільш низькорівневих засобів, і лише потім

перейдемо до опису переривань BIOS і ОС. Це дозволить нам підійти до вивчення високорівневих засобів, маючи повне уявлення про те, як влаштована дискова система.

3.4.1 Дискові накопичувачі та їх будова

Перші персональні комп'ютери типу IBM PC не мали жорсткого диска – "вінчестера" (за вітчизняною термінологією, накопичувача на жорсткому магнітному диску – НМД). Вони були обладнані двома флопі-дисками (накопичувачами на гнучкому магнітному диску – НГМД), які і являли собою дискову систему. У таких комп'ютерах встановлені зазвичай два накопичувачі для флопі-дисків (дискет).

Ці накопичувачі підключені до контролера – спеціального пристрою, що виконує функції керування. Контролер звичайно виконаний у вигляді плати і вставлений у роз’єм системної шини, що знаходиться на основній платі комп'ютера.

У комп'ютері IBM PC використовували дискети діаметром 5,25". Зараз все більше користувачів відмовляються від громіздких дискет діаметром 5,25", віддаючи перевагу більш компактним дискетам діаметром 3,5". Комп'ютер IBM XT мав один чи два НГМД для дискет діаметром 5,25" і, як правило, один НМД ємністю 20 Мбайтів. Всі дисководи підключалися до одного загального контролера. Модель IBM AT і комп'ютери більш високого класу можуть мати кілька дискових контролерів, два НГМД із різним діаметром (3,5" і 5,25") і декілька НМД.

Ємність сучасних НМД складає сотні Мбайтів і навіть Гбайтів. Зараз, коли вартість 1 Мбайта дискової пам'яті не така вже висока, багато користувачів встановлюють у своєму комп'ютері диски ємністю понад 40-60 Гбайтів. Проте, і такого обсягу пам'яті часто виявляється недостатньо.

Мабуть, самий цікавий різновид сучасних дискових накопичувачів – *оптичні* чи лазерні. Тільки оптичні накопичувачі здатні цілком вирішити проблему збереження величезних обсягів інформації.

В даний час існує кілька типів оптичних дискових накопичувачів. Це пристрій зчитування компакт-дисків CD-ROM, пристрій запису CD Recordable, накопичувачі WORM і магнітооптичні накопичувачі.

Диски CD-ROM (Compact-Disk, Read-Only Memory) – це диски, що за своїм форматом і технологією запису інформації нагадують звукові компакт-диски. Вони мають діаметр 120 мм і можуть містити приблизно 650 Мбайтів інформації. Ця інформація записується один раз і згодом може тільки читатися, як з постійного запам'ятовувального пристрою.

Диски CD Recordable зовні дуже схожі на диски CD-ROM, однак за допомогою спеціального пристрою користувач може записати на нього свою інформацію. При необхідності можна на той самий диск

дозаписувати нові дані. Диск CD Recordable можна прочитати за допомогою звичайного пристрою зчитування компакт-дисків CD-ROM.

WORM - диски (Write Once, Read Many) призначені для одноразового запису і багаторазового зчитування даних. Ці диски, як і CD Recordable, щонайкраще підходять для архівного збереження інформації, наприклад, вмісту великих баз даних.

Магнітооптичні накопичувачі можуть багаторазово записувати інформацію на той самий носій (як звичайні магнітні диски). Це самі дорогі дискові накопичувачі. Їх продуктивність порівнянна з продуктивністю звичайних жорстких дисків (наприклад, один з накопичувачів MaxOptix має ємність 1,3 Гбайтів при середньому часі доступу 19 мс).

Основний недолік лазерних накопичувачів (крім магнітооптичних) – відносно невисока швидкодія в порівнянні з традиційними накопичувачами на жорстких дисках. Однак цей недолік поступово переборюється і, очевидно, оптичні накопичувачі незабаром одержать широке поширення, особливо для роботи з великими архівами і базами даних.

Що ж, власне, представляє із себе диск?

Дискета (флопі-диск) – це кругла пластинка в квадратному конверті, покрита з двох сторін магнітним матеріалом. Цей матеріал схожий на той, що використовується в магнітних стрічках звичайних побутових магнітофонів, але відрізняється за деякими характеристиками (наприклад, за формою і шириною петлі гістерезису). Ближче до центра в диску знаходиться маленький отвір, призначений для синхронізації. Коли дискета вставляється в дисковод, по обидва боки (зверху і знизу) до неї притискаються магнітні головки. При цьому немає ніякого зазору між головками і поверхнею дискети. За допомогою спеціального крокового двигуна головки можуть переміщатися стрибкоподібно уздовж радіуса диска, як би прорисовуючи при обертанні диска концентричні кола. Ці кола називаються *доріжками, треками, циліндрами* – у літературі можна зустріти різні назви.

Жорсткий диск складається з декількох твердих круглих пластинок, покритих магнітним матеріалом. Пластинки обертаються з величезною швидкістю (порядку 3600-7000 об/хв і більше) у герметичному корпусі. Біля кожної сторони пластинки розташовується по одній магнітній головці, але ці головки не дотикаються до диска, а плавають на повітряній подушці в безпосередній близькості від його поверхні. Подаючи команди дисковому контролеру, програма може переміщати блок головок уздовж радіуса диска, переходячи в такий спосіб від одного циліндра до іншого.

Переміщаючись уздовж окружності доріжки, магнітна головка може записувати чи зчитувати інформацію приблизно так, як це відбувається в побутовому магнітофоні. Запис виконується бітами, при цьому додається різна службова інформація й інформація для контролю даних.

Дані записуються не суцільним потоком, а блоками визначеного розміру. Ці блоки називаються *секторами*. Сектор являє собою найменший обсяг даних, що записується чи прочитується контролером.

Для кожного сектора виконується контроль запису або зчитування. При записі сектора обчислюється контрольна сума всіх байтів, що знаходяться в секторі, і ця контрольна сума записується на диск у службову область, розташовану після сектора. При зчитуванні ця контрольна сума обчислюється заново і порівнюється з контрольною сумою, прочитаною зі службової області. При розбіжності контролер повідомляє про помилку.

Доріжки нумеруються починаючи *від нульової*, *головки* – теж починаючи *від нульової*, а от *сектори* – починаючи *з першого*. Чому так було зроблено – сказати важко, але саме така нумерація використовується при роботі з контролером диска і функціями переривання BIOS, що обслуговують дискову підсистему. В операціях зчитування або запису на фізичному рівні необхідно вказувати номер доріжки (0, 1, ...), головки (0, 1, ...), номер сектору (1, 2, ...).

Для правильної роботи з дисками на фізичному рівні програма повинна мати у своєму розпорядженні істотно більшу інформацію про диски, ніж просто номер потрібної доріжки чи головки. Наприклад, вона повинна знати, скільки головок і скільки доріжок має той чи інший дисковий пристрій, скільки байтів міститься в одному секторі і багато чого іншого, тобто слід визначити конфігурацію дискової системи й основні параметри встановлених дискових накопичувачів.

3.4.2 Характеристики дискових накопичувачів

Перш ніж почати роботу з дисками на фізичному рівні, необхідно з'ясувати конфігурацію дискової системи – скільки дисководів і якого типу підключено до комп'ютера, скільки доріжок і головок є на кожному з дисководів і т.д. Спосіб, яким визначається конфігурація дискової системи, залежить від моделі комп'ютера (PC, XT, AT), тому спочатку треба визначити тип комп'ютера.

Визначення типу комп'ютера

ПЗП базової системи введення/виведення BIOS містить за адресою F000h:FFFEh *байт конфігурації*, значення якого можна використовувати для ідентифікації типу комп'ютера (табл.3.1). Проаналізувавши значення байта конфігурації, можна зробити попередній висновок про склад дискової системи комп'ютера. Для комп'ютерів IBM PC і IBM PC/XT конфігурація дискової системи визначається установкою перемикачів на основній платі, зокрема, перемикачами встановлюється кількість

підключених до системи НГМД (для визначення байта конфігурації можна скористатися перериванням INT 11h).

Таблиця 3.1 – Значення байта конфігурації

Значення	Тип комп'ютера	Наявність НМД
FFh	Оригінальний IBM PC	Немає НМД
FEh	IBM PC/XT, Portable PC	Може бути НМД
FDh	PCjr	Немає НМД
FCh	IBM PC/AT (нас цікавить саме цей тип комп'ютерів)	Конфігурація дискової системи повинна визначатися, виходячи з вмісту CMOS-пам'яті
FBh	IBM PC/XT з пам'яттю 640 Кбайтів на материнській платі	Може бути НМД
F9h	Convertible PC	Немає НМД

Комп'ютери IBM PC/AT (і більш високого класу) мають на основній платі CMOS-пам'ять з малим енергоспоживанням, що живиться від акумулятора і в якій зберігається інформація про конфігурацію дискової системи. У процесі ініціалізації BIOS зчитує цю інформацію і записує її у свою внутрішню область даних.

Зауважимо, що нові моделі комп'ютерів можуть мати й інші, не перераховані вище коди ідентифікації.

Аналіз вмісту CMOS-пам'яті

Програма не може безпосередньо адресувати CMOS-пам'ять, як звичайну оперативну пам'ять. Для роботи з CMOS-пам'яттю необхідно використовувати порти введення/виведення з адресами **70h** і **71h**, причому процедура запису або зчитування складається з двох кроків.

На першому кроці операції зчитування-запису програма повинна записати в порт 70h номер потрібної комірки CMOS-пам'яті (0...3Fh). На другому кроці програма повинна звернутися до порту 71h для виконання запису в зазначену комірку пам'яті чи зчитування з неї.

Приведемо фрагмент програми, складеної мовою асемблера, що зчитує байт із CMOS-пам'яті з адресою 12h:

```
mov     al,12h
out     70h,al   ; задаємо адресу в CMOS-пам'яті
jmp     $+2      ; невелика затримка
in      al,71h   ; записуємо в AL прочитане значення
```

Мовою C++ це можна зробити таким чином:

```
outp(0x70, 0x12);              // задаємо потрібну адресу
char cfg_byte = inp(0x71);     // зчитуємо значення байта
```

Записування у CMOS-пам'ять виконується аналогічно.

При аналізі конфігурації дискової системи для нас найбільший інтерес становлять комірки CMOS-пам'яті з такими адресами:

14h *– байт конфігурації.*

Значення цього байта можна знайти двома шляхами.

Перший шлях – за допомогою переривання INT 11h, яке повертає в регістрі AX байт конфігурації системи, який можна використовувати для визначення кількості НГМД і наявності НМД. Це переривання найкраще використовувати для IBM PC/XT і IBM PC. Для IBM PC/AT необхідно докладніше досліджувати вміст CMOS-пам'яті. Тому краще визначати байт конфігурації через порти введення-виведення.

Самий молодший біт байта конфігурації (біт 0) - ознака наявності в системі НМД. Значення біта 0, рівне нулю, говорить про те, що в системі немає жодного НГМД. Якщо цей біт встановлений у 1, то комп'ютер обладнаний НМД, інакше дискова система складається тільки з накопичувачів на гнучких магнітних дисках. Біти 7 і 6 містять інформацію про кількість НГМД:

Вміст бітів 7 і 6	00	01	10	11
Кількість НГМД	1	2	3	4

10h *– тип НГМД.*

Молодша і старша тетради цього байта описують, відповідно, другий і перший НГМД (табл.3.2):

Таблиця 3.2 – Значення байта типу НГМД

Значення	Ємність, Кбайт	Діаметр	Кількість секторів на одну доріжку	Кількість доріжок
0000	НГМД немає	-	-	-
0001	360	5,25"	9	40
0010	1200	5,25"	15	80
0011	720	3,5"	9	40
0100	1440	3,5"	18	80

12h – тип НМД C: і D: .

Цей байт розділений на дві тетради аналогічно байта, що описує НГМД. Але у тетраді можна закодувати лише 16 значень, а типів НМД значно більше. Тому тип 15 використовується спеціальним чином – якщо тип НМД у молодшій тетраді (диск C:) дорівнює 15, то правильне значення типу знаходиться в CMOS-пам'яті за адресою **19h**. Аналогічно

для диска D: цей тип можна взяти з байта за адресою **1Ah** (якщо вміст старшої тетради байта з адресою 12h дорівнює 15).

Якщо у комп'ютері встановлений НМД з інтерфейсом ESDI , SCSI чи іншим спеціалізованим інтерфейсом, то, як правило, для роботи з ними використовується спеціальна "дискова" базова система введення/виведення. Відповідна мікросхема ПЗП може бути розташована безпосередньо в контролері. При цьому в CMOS-пам'яті в комірки 12h для типу диска може бути зазначене нульове значення, незважаючи на те, що диск встановлений. Переривання INT 11h, проте, скаже вам, що в системі є НМД.

Якщо використовується "дискова" базова система введення/виведення, то вона сама ініціалізує таблицю параметрів диска (буде описана пізніше) і виконує обробку переривання INT 13h. Оскільки MS-DOS при звертанні до дисків використовує саме це переривання, то не виникає ніяких проблем, зв'язаних з відсутністю типу диска в CMOS-пам'яті. Інші операційні системи такі, як Windows NT і OS/2 використовують для роботи з дисками спеціальні драйвери.

Скорочений перелік параметрів для стандартних типів НМД (залежить від версії BIOS) наведено у додатку А. Для всіх приведених у таблиці типів дисків на одній доріжці розташовується 17 секторів. Стандартний комп'ютер IBM PC/XT комплектується звичайно НМД із типом 1, тип 2 використовується в стандартному комп'ютері IBM PC/AT. Інші типи НМД використовуються головним чином старими версіями BIOS.

Таблиці параметрів НМД і НГМД

Для роботи з диском на фізичному рівні необхідно знати такі його характеристики, як кількість головок, секторів і ін. Ці характеристики можна визначити з таблиць параметрів НГМД і НМД, заповнюваних BIOS у процесі ініціалізації системи.

Аналізуючи вміст CMOS-пам'яті в комп'ютерах IBM PC/AT (або установлення перемикачів конфігурації на основній платі в комп'ютерах IBM PC і IBM PC/XT), BIOS у процесі ініціалізації створює таблицю параметрів дискети DPT (Diskette Parameter Table), а також одну чи дві таблиці параметрів жорсткого диска HDPT (Hard Disk Parameter Table). Якщо є спеціальна "дискова" система введення/виведення, то вона сама створює таблиці HDPT.

Таблиця параметрів дискети DPT має довжину 10 байтів, її адреса розташовується в області даних BIOS за адресою 0000h:0078h, що відповідає вектору переривання INT 1Eh. Основні параметри DPT наведені у табл.3.3. Всі часові змінні залежать від частоти тактового генератора контролера НГМД.

Адреси *таблиць параметрів жорстких дисків HDPT* розташовані за адресами: 0000h:0104h, що відповідає вектору переривання INT 41h (для першого фізичного диска) і 0000h:0118h, що відповідає вектору переривання INT 46h (для другого фізичного диска). Ці таблиці мають формат, наведений у табл.3.4.

Таблиця 3.3 – Параметри таблиці параметрів дискети - DPT

Зсув, байт	Розмір, байт	Ім'я поля	Опис
0	1	srt_hut	Біти 0...3:SRT (Step Rate Time) - затримка для переключення головок, у межах 1 - 16 мс (0Fh - 1 мс, 0Eh - 2 мс, 0Dh - 3 мс, ...). Біти 4...7: Затримка розвантаження головки, лежить у межах 16 - 240 мс і задається з інтервалом 16 мс (1 - 16 мс, 2 - 32 мс, ..., 0Fh - 240 мс)
1	1	dma_hlt	Біт 0: Значення цього біта, рівне 1, говорить про те, що використовується прямий доступ до пам'яті; Біти 2...7: Час завантаження головок HLT - інтервал між сигналом завантаження головок і початком операції зчитування-запису, лежить у межах 2 - 254 мс і задається з інтервалом 2 мс (1 - 2 мс, 2 – 4 мс, ..., 0FFh - 254 мс)
2	1	motor_w	Затримка перед вимиканням двигуна
3	1	sec_size	Код розміру сектора в байтах: 0 - 128; 1 - 256; 2 - 512; 3 - 1024
4	1	eot	Номер останнього сектора на доріжці
5	1	gap_rw	Довжина міжсекторного проміжку для зчитування чи записування
6	1	dtl	Максимальна довжина переданих даних. Використовується коли не задана довжина сектору
7	1	gap_f	Довжина міжсекторного проміжку для операції форматування
8	1	fill_char	Байт-заповнювач для форматування, звичайно використовують символ F6h
9	1	hst	Час установлення головки в мс
10	1	mot_start	Час запуску двигуна в 1/8 частках секунди

Найбільш корисна інформація, яку можна витягти з таблиці параметрів дискети – це код розміру сектору. Якщо доведеться працювати з нестандартним розміром сектору (відмінним від 512 байтів), то не обійтися без цієї таблиці.

Таблиця параметрів жорсткого диска містить такі найважливіші значення, як максимальна кількість доріжок і максимальна кількість головок. Якщо не вдалося визначити тип диска, то таблиця HDPT - єдине надійне місце, звідки можна одержати інформацію про кількість доріжок і головок.

Таблиця 3.4 – Параметри таблиці параметрів твердих дисків - HDPT

Зсув, байт	Розмір байт	Ім'я поля	Опис
0	2	max_cyl	Максимальна кількість доріжок на диску
2	1	max_head	Максимальна кількість магнітних головок
3	2	srwcc	Початкова доріжка для попереднього запису (Starting reduced-write current cylinder)
5	2	swpc	Початкова доріжка для попередньої компенсації при записі (Starting write precompensation cylinder)
7	1	max_ecc	Максимальна довжина блока корекції помилок ECC
8	1	dstopt	Параметри пристрою: біт 7 - заборона відновлення; біт 6 - заборона відновлення по блоку корекції помилок ECC (Error Correction Code); біти 2-0 - додаткові параметри пристрою
9	1	st_del	Стандартна величина затримки
10	1	fm_del	Величина затримки для форматування диска
11	1	chk_del	Величина затримки для перевірки диска
12	4	reserve	Зарезервовано

3.4.3 Програмування контролера НГМД

Більшість операцій можна виконати на рівні функцій BIOS. Це найпростіший і надійний спосіб роботи з диском на фізичному рівні. Однак в окремих випадках може знадобитися безпосередній доступ до контролера НГМД, а особливо при розробці систем захисту даних від несанкціонованого копіювання.

Інформація, необхідна для роботи з контролерами НГМД, описана в спеціальній літературі і орієнтована насамперед не на виконання операцій зчитування-записування (які краще виконувати за допомогою функцій BIOS), а на керування контролером і одержання стану контролера. Саме ці операції використовуються для організації захисту даних від НСК.

3.4.4 Функції BIOS для роботи з дисками

Найкращий і самий безпечний спосіб роботи з дисками на фізичному рівні в середовищі ОС – використання функцій базової системи введення/виведення BIOS. Ці функції враховують всі особливості апаратури і надають досить широкий набір засобів доступу до дисків на фізичному рівні.

Вся дискова підсистема обслуговується перериванням **INT 13h**. Це переривання виконує безліч функцій. Для виклику визначеної функції програма повинна занести її код у регістр AH. При цьому в інші регістри варто записати параметри – номер НМД чи НГМД, номер циліндра, головки, адреси таблиць параметрів НМД чи НГМД і т.д.

У табл.3.5 наведено короткий перелік функцій переривання INT 13h. Детальний опис цих функцій можна знайти у спеціальній літературі.

Таблиця 3.5 – Функції переривання INT 13h

Номер функції	Опис
00h	Скидання дискової системи
01h	Визначення стану дискової системи
02h	Зчитування сектора
03h	Запис сектора
04h	Перевірка сектора
05h	Форматування доріжки
06h	Форматування доріжки НМД
07h	Форматування НМД
08h	Одержання поточних параметрів НГМД чи НМД
09h	Ініціалізація таблиць параметрів НМД
0Ah	Зчитування довге (тільки для НМД)
0Bh	Запис довгий (тільки для НМД)
0Ch	Пошук циліндра (тільки для НМД)
0Dh	Альтернативне скидання НМД
0Eh	Зчитування буфера сектора (тільки для НМД)
0Fh	Запис буфера сектора (тільки для НМД)
10h	Перевірка готовності НМД
11h	Рекалібровка НМД
12h	Перевірка пам'яті контролера НМД
13h	Перевірка НМД
14h	Перевірка контролера НМД
15h	Одержання типу НМД чи НГМД
16h	Перевірка заміни диска
17h	Установлення типу дискети
18h	Установлення середовища носія даних для форматування
19h	Паркування головок (тільки для НМД)
1Ah	Форматування НМД з інтерфейсом ESDI

3.4.5 Функція _bios_disk

Стандартна бібліотека Borland C++ містить спеціальну функцію _bios_disk(), що полегшує роботу з диском на рівні BIOS. Ця функція описана у заготовочному файлі <bios.h>, і прототип її такий:

```
unsigned _bios_disk (unsigned funct, struct diskinfo_t *diskinfo).
```

Параметр diskinfo – це покажчик структури, що описує необхідні параметри, такі як номер доріжки, номер головки і т.д.:

```
struct diskinfo_t
{   unsigned drive;        // номер дисковода
    unsigned head;         // номер головки
    unsigned track;        // номер доріжки
    unsigned sector;       // номер першого сектора
    unsigned nsectors;     // кількість секторів
    void far *buffer;      // адреса буфера в пам'яті
}.
```

Параметр funct задає ідентифікатор виконуваної функції, можливі значення його наведені у табл.3.6. Перед використанням функції _bios_disk програма повинна заповнити поля структури diskinfo і вказати відповідний параметр funct.

Таблиця 3.6 – Значення параметра funct для функції_bios_disk

Значення	Опис
_DISK_FORMAT (05h)	Форматування доріжки, описаної в структурі diskinfo. Програма повинна задати для цієї функції в структурі diskinfo номер НГМД чи НМД, для якого виконується форматування, номер головки і номер форматованої доріжки. Крім цього, програма повинна встановити покажчик buffer на підготовлений буфер формату. Варто також виконати всі підготовчі дії, пов'язані з настроюванням контролера НГМД і таблиці параметрів дискети
_DISK_READ (02h)	Зчитування одного чи декількох секторів. Ця функція аналогічна функції 2 переривання INT 13h. Якщо при зчитуванні секторів відбулася помилка, її код буде повернутий функцією _bios_disk у старшому байті. При успішному завершенні операції функція повертає 0
_DISK_WRITE (03h)	Запис одного чи декількох секторів на диск. Функція аналогічна попередньої, за винятком того, що дані з буфера записуються на диск
_DISK_RESET (00h)	Скидання контролера НГМД. Для цієї функції не треба заповнювати структуру diskinfo, оскільки її вміст ігнорується. Скидання контролера виконують після того, як відбулася помилка при виконанні іншої операції, наприклад, зчитування або запису. Після скидання можна спробувати повторити виконання операції
_DISK_STATUS (01h)	Одержання стану НГМД після виконання останньої операції. Старший байт значення, яке повертається функцією _bios_disk, містить байт стану
_DISK_VERIFY (04h)	Перевірка диска. За допомогою цієї функції можна переконатися в тому, що зазначені сектори існують і можуть бути прочитані в пам'ять. Додатково виконується перевірка CRC. Функція перевірки диска використовує всі поля структури diskinfo. При помилці старші 8 бітів значення, яке повертається функцією, містять байт стану.

3.5 Форматування дискети

Перед нанесенням інформації на дискету (або оптичний диск), її заздалегідь форматують. Розглянемо процес форматування носія на прикладі дискети.

В процесі розмітки диска кожна поверхня диска розбивається на 80 концентричних доріжок (треків). Таким чином, на дискеті 80×2=160 доріжок. Оскільки трек має форму кільця, то початок треку – це лінія, що виходить з центру і проходить через центр індексного отвору.

Після форматування кожна доріжка розбивається на певну кількість полів і має структуру, наведену на рис.3.2.

	Фізична адресна мітка (індексний отвір)	Початок кожної доріжки
GAP4A	Передіндексний синхронізуючий проміжок	
GAP5	Доіндексний проміжок	
IAM	Індексна адресна мітка	
GAP1	Після індексний проміжок	
CCCCC	Поле синхронізації	1-й сектор
IDAM CHRN CRC	Заголовок сектора (ідентифікатор), або секторний ID-маркер	
GAP2	Проміжок після ID-маркера	
DATA_IDA …DATA ... CRC	Місце для розташування даних	
GAP3	Проміжок після даних	
…	… … … …	…
CCCCC	Поле синхронізації	(N-1)-й сектор
IDAM CHRN CRC	Секторний ID-маркер	
GAP2	Проміжок після ID-маркера	
DATA_IDA …DATA ... CRC	Місце для розташування даних	
GAP3	Проміжок після даних	
CCCCC	Поле синхронізації	N-й (останній) сектор
IDAM CHRN CRC	Секторний ID-маркер	
GAP2	Проміжок після ID-маркера	
DATA_IDA …DATA ... CRC	Місце для розташування даних	
GAP4	Завершуючий проміжок	

Рисунок 3.2 – Структура доріжки відформатованої дискети

Заголовок сектора (секторний ID-маркер) містить такі байти:

IDAM (1 байт) – символи FEh – означають, що це заголовок сектора;

C – номер циліндра;

H – номер головки;

R – номер сектора;

N – код розміру сектора (розмір сектора обчислюється за формулою 128×2^N);

CRC (2 байти) – контрольна сума ID-маркера (C×N).

Поле даних сектора містить:

DATA_IDA (1 байт) – символи FBh – означають, що починаються дані;

DATA - безпосередньо дані або символи заповнення, якщо даних немає;

CRC (2 байти) – контрольна сума, що розраховується і поміщається сюди при записуванні інформації у сектор.

Проміжки. В них при форматуванні записується код 4Eh. Розміри проміжків становлять 50÷80 байтів.

Поле синхронізації (синхробайти) – необхідні для забезпечення достовірності інформації. Вони забезпечують затримку, під час якої система виконує підрахунок контрольної суми.

3.6 Захист від НСК методом прив'язки до дискети

Цей метод захисту не вимагає великих фінансових витрат при впровадженні, однак має низьку стійкість до зламу. Внаслідок цього застосування такого захисту виправдано тільки для ПЗ нижньої цінової категорії. Для подібних програм важливі популярність і великі тиражі (іноді і за рахунок піратських копій). Використання більш могутньої і дорогої системи захисту в даному випадку не буде мати сенсу.

Серед методів захисту, пов’язаних з прив’язкою до дискет, можна виділити такі:

- нестандартні параметри форматування (використання додаткових доріжок, “зайві сектори”, нестандартні номери і розміри секторів, нестандартний Interleave, нестандартні заголовки секторів і таке інше);
- інформація в проміжках;
- перевищення об’єму доріжки;
- сектори з Bad CRC;
- вимірювання довжини доріжки;
- фізичні дефекти на носії;
- неоднозначні біти;
- збій синхронізації;
- пошкоджені або відсутні адресні маркери;
- різношвидкісні доріжки.

3.6.1 Використання додаткової доріжки

При цьому способі захисту використовується додаткова, 81-а доріжка на дистрибутивній дискеті. Зауважимо, що на гнучкому диску є місце для розміщення ще трьох доріжок, понад прийнятих 80. Дисковод дозволяє вести на них запис даних. Тому на додатковій 81-й доріжці на дистрибутивній дискеті зберігається інформація для контролюючої частини захищуваної програми.

При копіюванні на стандартно відформатовану дискету додатково введена 81-а доріжка сприйнята не буде, що і виявляється контролюючою частиною захищеної програми.

3.6.2 Перестановка в нумерації секторів

При підготовці нової дискети до роботи вона форматується, тобто визначається кількість доріжок, довжина сектора, кількість секторів на доріжці, нумерація секторів, тобто формується секторний ID-маркер (див. рис. 3.2) і виконуються інші операції. Якщо ці дії здійснюються з установленням параметрів (довжини сектора, нумерації секторів, величини міжсекторного проміжку й ін.), відмінних від прийнятих за замовчуванням, то такий процес будемо називати *нестандартним форматуванням дискети*.

Один з методів захисту від копіювання ґрунтується на перестановці номерів секторів на доріжці, тобто замість звичайної послідовності 1,2,3,4,5,6,7,8,9 вводиться, наприклад, така: 1,5,3,7,9,8,6,2,4.

При виконанні програми на скопійованій дискеті її контролююча частина визначає порядок проходження секторів на заданій доріжці. Оскільки дискета, на яку здійснене копіювання, була відформатована звичайними засобами, то нумерація секторів встановлена зазвичай послідовна.

Надалі контролююча частина програми здійснює порівняння реально існуючого порядку проходження секторів із установленим на дистрибутивній дискеті. І оскільки порядок номерів не збігатиметься, то виконання програми припиниться.

3.6.3 Введення однакових номерів секторів на доріжці

Іншою схемою захисту, що базується на ідеї нестандартного форматування, є спосіб, при якому частина секторів на визначеній доріжці нумерується однаково (наприклад, 1,2,3,3,3,6,7,8,9). Далі у ці сектори записуються деякі різні дані. Контролююча частина захищеної програми повинна визначити, чи є на дискеті кілька секторів з однаковим номером. Для цього вона надсилає запит на зчитування даних із сектора, номер якого повторюється (у нашому прикладі із сектора 3). У цьому випадку

контролер НГМД при виведенні головки зчитування/запису на сектор 3 може зчитати будь-який сектор з даним номером. Цей процес повторюється задану кількість разів підряд, і чергові зчитані дані порівнюються з отриманими раніше. У випадку виявлення розходження робиться висновок про наявність на дискеті секторів з однаковим номером. Якщо ж за задану кількість повторень циклу N розходження в даних не знайдено, то робиться висновок про одиничність сектора з номером 3.

Оскільки дискета, на яку здійснене копіювання, була відформатована звичайними засобами, то нумерація секторів на всіх доріжках встановлюється послідовною. Контролююча частина захищеної програми організує перевірку на наявність декількох секторів з однаковими номерами. І оскільки кожен номер сектора на заданій доріжці присутній тільки один раз, то виконання програми припиняється.

3.6.4 Введення міжсекторних зв'язків

В одному з методів захисту від копіювання використовується модифікація викладеного вище способу, що використовує організацію декількох секторів з однаковим номером на заданій доріжці.

Сутність методу. На виділеній доріжці дистрибутивної дискети шляхом застосування спеціальної програми організується кілька секторів з однаковим номером і в них записуються деякі різні дані. Нехай, наприклад, номери на 79-й доріжці задані як 2,2,1,4,5,6,7,8,9 і в першому секторі з номером 2 записано "С", а в другому секторі 2 - "В". Береться ще одна доріжка (наприклад, 78) та сектор на ній (наприклад, 1-ий) і в нього записується деяка інформація, нехай "А". У контролюючій частині захищуваної програми організовується зчитування сектора 1 на 78-й доріжці і відразу ж запит зчитування сектора 2 на 79-й доріжці. Тим самим завжди забезпечується перехід до зчитування інформації з другого сектора з номером 2 на доріжці 79, тобто прочитаємо біграму "АВ" з дистрибутивної дискети.

Нехай було здійснено "піратське" копіювання дискети з програмою, що захищається, на звичайним чином відформатовану дискету. Контролююча частина програми зчитує 1-ий сектор з 78 доріжки. Далі йде запит на зчитування сектора 2 на 79 доріжці. Оскільки сектори на "піратській" дискеті пронумеровані стандартно (1,2,3,4,5,6,7,8,9), то після запиту на зчитування сектора 2 буде отримане значення "С". Отже, буде прочитана біграма "АС". Таким чином, контролююча частина захищуваної програми встановлює, що у випадку, коли дискета не дистрибутивна, зчитується біграма "АС", що означає, що зчитано сектор 2 з 79 доріжки, що знаходиться на іншій фізичній відстані від початку доріжки в порівнянні з відстанню для сектора 2 на дистрибутивній дискеті, коли зчитується біграма "АВ". Програма переривається.

3.6.5 Зміна довжини секторів

Ще одна схема захисту, заснована на методі нестандартного форматування, використовує зміну довжини сектора. Нагадаємо, що стандартна довжина сектора, з яким працює MS DOS за замовчуванням, – 512 байтів, при цьому на доріжці розміщається 9 секторів. У процесі спеціального форматування дискети на заданій доріжці довжина секторів може бути встановленою або 128, або 256 байтів.

Введена зміна довжини секторів на заданій доріжці визначається контролюючою частиною захищуваної програми. Це може здійснюватись шляхом звертання до сектора з максимальним номером або перевіркою того, що довжина секторів на зазначеній доріжці 128, або 256 байтів.

Стандартно відформатована дискета, на яку проведене "піратське" копіювання захищуваної програми, має довжину секторів на всіх доріжках 512 байт (по 18 секторів на доріжці). У ході виконання програми її контролююча частина здійснює перевірку довжини секторів на заданій доріжці. Якщо контролююча частина використовує спосіб звертання до сектора з максимальним номером (більшим за 18), то такий сектор знайдений не буде (оскільки їх тільки 18) і виконання програми припиняється.

Аналогічно працює контролююча частина й у способі, у якому здійснюється перевірка, що довжина секторів 128 (чи 256) байтів.

3.6.6 Зміна міжсекторних проміжків

Таблиця параметрів дисків (див. табл.3.3), за допомогою якої частково керується робота НГМД, містить параметр, що визначає величину міжсекторного проміжку. Вибирається доріжка на дискеті, що спеціальною програмою форматується заново зі зміненим значенням міжсекторного проміжку в таблиці параметрів дисків, що веде до нестандартного розташування секторів. Контролююча частина захищуваної програми при звертанні до обраної доріжки здійснює її аналіз і робить висновок, чи є дискета дистрибутивною.

3.6.7 Інформація в проміжках

Простий спосіб одержання доступу до міжсекторних проміжків полягає в заданні розміру сектора в заголовку (CHRN) більшого ніж його реальний розмір, зазначений при форматуванні.

При зчитуванні такого сектора спочатку прочитається його поле даних (див. рис.3.2), потім межсекторный проміжок (GAP3), заголовок наступного сектора і т.д. Останній сектор може таким чином контролювати довжину доріжки. При спробі записати такий сектор він затре заголовок наступного сектора, і на доріжці одночасно будуть сектори різних

розмірів. Це використовується в деяких системах захисту (наприклад, ECP); якщо ж це небажано, необхідно перервати запис після запису потрібного числа байтів.

Між іншим, переривання поточної операції – широко використовуваний прийом, що дозволяє одержувати сектори з “поганим” CRC, "слабкі" біти і т.д.

Якщо інформацію з проміжків достатньо лише зчитувати, то це можна зробити, використовуючи команду контролера READ_TRACK. При цьому на доріжці цілком зберігається стандартний формат (так робиться, наприклад, у системі CONVOY).

3.6.8 Перевищення обсягу доріжки

Перевищення обсягу доріжки буває фізичне і логічне.

При *логічному перевищенні* сектори насправді містяться на доріжці, але кінець останнього сектора (тобто байт, що відстоїть від початку на $128*2^N$) попадає на початок доріжки. При записі такого сектора затреться початок першого сектора (звичайно, якщо вчасно не перервати операцію). Тому такі сектори не копіює COPYIIPC. Широко поширене використання таких секторів для одержання доступу до межсекторних проміжків, контролю довжини доріжки, а також збоїв синхронізації.

Фізичне перевищення обсягу доріжки можна назвати ще “багатооборотним” форматуванням. Воно має місце, коли сума реальних розмірів секторів (тобто зазначених у заголовку команди форматування) плюс, можливо, службові поля, перевищує обсяг доріжки. Тоді форматування не закінчиться після одного оберту дискети, і спочатку буде зіпсований маркер початку доріжки (це допустимо, оскільки він не використовується ні при яких дискових операціях), потім заголовок і дані першого сектора і т.д. Після формування всіх секторів до кінця доріжки запишеться поле GAP4B, що знищить сектори, сформовані на першому оберті, якщо, звичайно, не перервати форматування. Копіювальники, не враховуючу цю можливість, не зможуть скопіювати таку доріжку. Крім того, метод можна використовувати для досягнення особливо складного розташування секторів на доріжці.

3.6.9 Введення логічних дефектів у заданий сектор

Цей метод називають ще методом організації секторів *з Bad CRC*.

Спочатку згадаємо фізичну і логічну будову доріжки на гнучкому диску. Як відомо, початок доріжок на дискеті відзначено індексним отвором. Кожен сектор на доріжці має дві частини: секторний ID-маркер і власне дані. ID-маркер має шість байтів і додаткові байти, що ідентифікують

його для контролера накопичувача на гнучких магнітних дисках (НГМД) як секторний маркер, а не як дані. Ці шість байтів такі (див. рис.3.2):

1-й. Номер доріжки з 0 по 79 включно.

2-й. Номер сторони (головки). Верхня сторона позначається 0, нижня – 1.

3-й. Номер сектора на треку з 1 по 18 включно.

4-й. Код розміру сектора в байтах: 0,1,2 чи 3 означають відповідно, 128, 256, 512 чи 1024 байта в секторі. За замовчуванням формат DOS має 512 байт у секторі.

5-й і 6-й. CRC байти для контролю за допомогою циклічного надлишкового коду можливих перекручувань.

Чотири перших байти секторного ID-маркера позначаються як CHRN. CRC-байти фактично містять 16-бітну контрольну суму CHRN-байт, що обчислюється, коли сектор записується. Коли ж сектор читається, контрольна сума переобчислюється для прочитаних з CHRN даних і порівнюється з даними з байтів CRC. Будь-яка розбіжність викликає CRC-помилку ID-маркера.

Наступними за ID маркером йдуть декілька міточних байтів маркера початку даних (BOD) і 512 байтів власне даних. Ще два перевірочних байти CRC слідують за даними. Вони містять контрольну суму 512 байтів даних, обчислену при записі сектора. Розбіжність між контрольними сумами, записаною в CRC байтах і обчисленою при зчитуванні, визначається як CRC-помилка даних. Після CRC-байтів першого сектора є міжсекторний проміжок, потім йдуть інші сектори і міжсекторні проміжки.

Сам метод полягає в тому, щоб мати на доріжці сектор з неправильною контрольною сумою. Створити такий сектор можна коротким записом, шляхом "відсічення" CRC (тобто перервати запис після запису сектора, але до запису другого байта CRC). Якщо ж дисковод не дозволяє зробити "відсічення" з точністю до байта, то можна відформатувати сектор з більшим кодом довжини (див. інформацію в проміжках), записати сектор, а потім відновити заголовок (форматуючи з обривом у GAP2).

Очевидно, обидва способи дають сектор, заповнений даними, але з помилковою CRC. Причому перевіряти, звичайно, потрібно і те, і інше.

Тепер зрозуміла ідея методу захисту, заснованого на використанні CRC-помилки даних. Її виникнення досягається в такий спосіб. Стандартна операція запису даних у сектор включає шість кроків:

- увімкнення мотора НГМД і коротке очікування його розгону;
- виконання операції пошуку заданого сектора і очікування переривання, що вказує на завершення даної операції;
- ініціалізація мікросхеми прямого доступу в пам'ять (DMA) 8237 для пересилання даних з пам'яті;
- посилання команди запису мікросхемі контролера НГМД 765 і очікування переривання, що вказує на закінчення пересилання даних;
- одержання інформації про статус контролера НГМД;

- вимкнення мотора.

На дистрибутивній дискеті при виконанні операції запису у визначений сектор під час четвертого кроку викладеної вище процедури здійснюється короткочасне вимкнення мотора НГМД, що приводить до спотворення записуваних даних. Контрольна ж сума даних у CRC-байтах буде мати значення, що було отримано за вихідними переданими даними. Так що при зчитуванні даних з цього сектора і підрахунку їх контрольної суми отримане значення буде відрізнятися від того, що міститься в CRC-байтах. Таким чином, контролююча частина захищуваної програми при спробі зчитування заданого сектора знайде CRC-помилку даних і визначить дискету як дистрибутивну.

При копіюванні раніше спотворені дані на заданому секторі сприймаються як вірні і за ними підраховується відповідна контрольна сума. Отже, при зчитуванні даного сектора на скопійованій дискеті CRC-помилка даних не виникає, і КЧЗП дискету відкидає.

3.6.10 Контроль довжини доріжки

Даний метод базується на збої синхронізації між полями GAP4B і GAP4A при форматуванні (рис.3.2). Довжина доріжки вимірюється з точністю до півбіта!

При формуванні доріжки поле GAP4A починає записуватися відразу після проходження індексного отвору. Після формування всіх секторів контроллер заповнює частину доріжки, що залишилася, полем GAP4B. Форматування припиняється негайно при досягненні індексного отвору. Це може відбутися в середині записуваного байта. Якщо після цього прочитати кінець доріжки, як суцільний масив даних, то поле GAP4A буде заповнено не 4Eh, а якимсь іншим символом, що представляє собою 4Eh, зрушений на деяке число півбітів.

Крім того, при проходженні індексного отвору, у момент закінчення форматування, контролер скидає на диск вміст внутрішнього буфера. Це унеможливлює визначення довжини доріжки по границі GAP4B/GAP4A (хоча ці дані, взагалі кажучи, випадкові, можна використовувати для ідентифікації дискети, як робиться, наприклад, у ARMOUR II і CERBERUS), тому для вимірювань використовується границя GAP4A/SYNC.

3.6.11 Зміна параметрів дисковода

Однією зі схем створення захисту від копіювання є зміна швидкості обертання диска. Стандартна швидкість обертання – 300 об/хв. Якщо її зменшити хоча б до 280 об/хв, у той час як дані для запису передаються з попередньою швідкістю, то це зменшення збільшує щільність збереження

даних на дискеті, і на кожній доріжці утвориться місце для додаткового десятого сектора. Ця обставина є основою захисту, оскільки робота з інформацією, записаною на дискеті, що обертається з новою швидкістю, можлива тільки при відповідній модифікації параметрів дисковода.

3.6.12 Технологія "ослаблених" бітів

Способом запису інформації на дискету є її подання у вигляді 0 чи 1. Даний метод захисту використовує запис деякої ділянки дистрибутивної дискети з невизначеним рівнем сигналу. Таким чином, з'являється ділянка "ослаблених" бітів. Дані з цієї ділянки при їх зчитуванні кілька разів підряд будуть сприйматися по-різному через те, що сигнал може перетворюватися тільки в два дискретних значення – 0 чи 1.

Неоднозначні (ослаблені) біти – це біти, записані з половинною напруженістю магнітного поля. Контролер не може їх чітко розпізнати як 0 чи 1, і, читаючи їх декілька разів підряд, можна одержати різні результати. Саме так і відрізняють вихідну дискету від її копії.

Одержувати слабкі біти можна за допомогою спеціальної апаратури, але, в принципі, досить і звичайного дисковода. Для цього можна маніпулювати швидкістю передачі даних або вибором дисковода.

При копіюванні дистрибутивної дискети ділянка "невизначених" даних прийме якесь фіксоване значення. Таким чином, контролююча частина захищеної програми забезпечує зчитування зазначеної ділянки кілька разів підряд. Якщо виходять однакові дані при всіх запитах зчитування, це означатиме, що дискета не дистрибутивна, і програма переривається.

Досить "інтелектуальні" копіювальники здатні виявляти і копіювати області ослаблених бітів. Тому рекомендується додатково контролювати розмір слабкої області і значення байтів, що попадають у неї частково.

3.6.13 Фізичне маркування дискети

Ідея цього методу захисту полягає в тому, що ушкоджену ділянку магнітного шару дискети неможливо скопіювати ніяким (!) способом. Послідовність дій повинна бути такою: спочатку псується поверхня дискети (для цього досить просто подряпати магнітний шар голкою, чи випалити ділянку шару лазером, що набагато безпечніше для головок дисковода), потім визначаються ушкоджені сектори і надалі перевіряється їх наявність.

Однією з найбільш відомих технологій захисту від копіювання є фізична позначка дискети лазерним променем шляхом її пропалювання. Такий дефект призводить до спотворення даних у секторі і виникненню CRC помилки даних при його зчитуванні. При копіюванні пробивання не

відтворюється на приймальній дискеті, хоча дані переписуються. Таким чином, CRC-помилка даних при зчитуванні сектору з копії не виникає.

Контролююча частина захищуваної програми, виконуючи зчитування відповідного сектору, встановлює нелегальність копії і виконує дії, передбачені розробником (переривання програми, знищення і т.п.).

Перевірка наявності пробивання може здійснюватися контролюючою частиною програми й іншим, більш надійним способом. Вона робить спробу запису в місце, де повинен знаходитися отвір. Невдача – явна ознака того, що дискета дистрибутивна. Якщо ж запис здійснюється успішно, це означатиме, що поверхня в даному місці не ушкоджена і дискета не дистрибутивна.

Розглянута технологія застосовується фірмою Vault Corporation у продукті PROLOCK і його подальших модифікаціях.

Основні помилки, що допускаються при реалізації цього методу:

- перевіряється тільки помилка при зчитуванні сектора. Дискета легко копіюється, якщо ушкоджені сектори замінити секторами з “Bad CRC”. Необхідно записати ушкоджений сектор, а потім знову прочитати;
- наявність помилки перевіряється за допомогою використання функції BIOS. Дуже легко створюється резидентна "заглушка" на 13h переривання, що при зчитуванні потрібного сектора повертає помилку, будучи прозорою для інших викликів BIOS. Щоб цього уникнути, потрібно або працювати безпосередньо з контролером (але тоді з'являється можливість імітації помилок у V86), або викликати BIOS, використовуючи точки входу в ПЗУ.

3.6.14 Збій синхронізації

Методи, засновані на збої синхронізації, не зважаючи на простоту реалізації, є найбільш захищеними від копіювання. Справа в тому, що всі поля даних і заголовки секторів синхронізуються окремо, і можуть бути зсунуті один відносно одного на нецiле число біт. Говорять, що на границі двох таких полів має місце збій синхронізації. Те ж саме може відбутися між кінцем і початком доріжки (див. вимірювання довжини доріжки).

Таким чином, якщо зчитати доріжку (чи будь-яку область зі збоєм синхронізації), як єдиний масив даних, то отриманий "зліпок" буде залежати від індивідуальних особливостей дисковода, на якому виконувався запис або форматування, і його буде практично неможливо відтворити на іншому дисководі.

Але цей метод може дати неоднозначні результати. Дійсно, якщо відбулося зсування на нецiле число півбітів, то результат зчитування буде мати нестабільний, імовірносний характер. У такий спосіб можливе упізнання оригінальної дискети як копії і навпаки. Варіації цього методу використовують, наприклад, CERBERUS і CopyLock.

3.6.15 Збійні чи відсутні адресні маркери

У неушкодженого сектора є два адресних маркери - IDAM (адресний маркер заголовка сектора) і DATA_AM (адресний маркер поля даних). З метою захисту можна створювати сектори без будь-якого з них.

Сектори без DATA_AM породжуються коротким форматуванням з перериванням у полі GAP2. Перевіряється наявність такого сектора спробою зчитування, яка повинна завершитися помилкою "Bad DATA AM".

Сектори без IDAM можна створити, якщо будь-який сектор на доріжці має довжину більше ніж вказувалося при форматуванні, тоді при записуванні такого сектора буде перезаписаний IDAM наступного сектора. Щоб при цьому не зіпсувати його DATA_AM, треба перервати записування у GAP2. Сектори без IDAM не можуть бути виявлені ніякою дисковою операцією. Для виявлення такого сектора необхідне коротке форматування, щоб відновити IDAM.

Сектори зі збійним IDAM хоча і мають IDAM, але з неправильним CRC. Такий сектор не виявляється командою READ_ID. Здійснити зчитування такого сектора можна лише командою ЗЧИТУВАННЯ ДОРІЖКИ чи ЗЧИТУВАННЯ СЕКТОРА, якщо відомі його CHRN.

Якщо сектор має збійний IDAM і не має поля даних (DATA_AM), то на ньому відбудеться збій команди ЗЧИТУВАННЯ ДОРІЖКИ, тобто такий сектор можна використовувати на початку доріжки, щоб утруднити її аналіз.

Цікавим є створення доріжки з єдиним сектором, що, до того ж має збійний заголовок (IDAM) і не має поля даних. Тоді цей сектор не буде виявлятися READ_ID, через збійний заголовок, а ЗЧИТУВАННЯ ДОРІЖКИ не буде працювати, оскільки на доріжці немає жодного поля даних. Таким чином, жоден з методів дослідження доріжки не знайде на ній інформації. Але якщо, знаючи CHRN сектора, виконати ЗЧИТУВАННЯ СЕКТОРА, то буде отримана помилка "Bad CRC", що означає, що дискета є оригіналом.

3.6.16 Різношвидкісні доріжки

Ідея полягає в тому, щоб розділити доріжку на частини, записані при різній швидкості передачі даних. Це простіше зробити не переключенням швидкості під час форматування, а декількома короткими форматуваннями на різних швидкостях.

3.6.17 Застосування фізичного захисного пристрою

Велика кількість апаратно-програмних способів захисту засновані на тому, що в комп'ютер додається спеціальний фізичний захисний пристрій. (Він під’єднується, наприклад, до порту введення-виведення.) При запуску

захищуваної програми її контролююча частина звертається до цього додаткового пристрою, перевіряючи його наявність. Якщо його не знайдено (деякі пристрої ще і формують код відповіді, який потім аналізується), то здійснюється зупинка програми, або якісь інші дії (наприклад, знищення інформації на дискеті чи вінчестері).

Загальний принцип роботи комп'ютера в цьому випадку такий. Після запиту на виконання захищуванї програми відбувається її завантаження в оперативну пам'ять і ініціалізація контролюючої частини. На фізичний пристрій захисту, приєднаний до комп'ютера через порт введення-виведення (або іншим способом), посилається запит. У відповідь формується код, що посилається через мікропроцесор в оперативну пам'ять для розпізнавання контролюючою частиною програми. У залежності від правильності коду відповіді програма або переривається, або виконується.

Один з найбільш розповсюджених захисних пристроїв є HASP by "Аладін". Це невеликий пристрій, що підключається до паралельного порту комп'ютера. Існує кілька різновидів ключів з пам'яттю, з годинником. Звертання до HASP відбувається винятково через API ключа і, до того ж, через ту саму точку.

Контрольні питання

1. Навести характеристику сучасних технологій захисту програмного забезпечення від копіювання.
2. Що таке операційна система і які основні її функції?
3. Яка роль BIOS та CMOS в операційних системах?.
4. Дати поняття переривання та охарактеризувати їх роль в ОС і алгоритм їх обробки.
5. Охарактеризувати рівні роботи з дисковою системою комп’ютерів.
6. Навести перелік відомих видів дискових накопичувачів та охарактеризувати їх.
7. Яким чином здійснюється доступ до вмісту CMOS-пам’яті (байт конфігурації, типи НГМД та НМД).
8. Охарактеризувати вміст таблиць параметрів НГМД та НМД і можливості доступу до них у програмах.
9. Навести перелік відомих функцій BIOS для роботи з дисками на фізичному рівні.
10. Охарактеризувати функції стандартної бібліотеки Borland C++ для роботи з дисками.
11. Якою є структура відформатованої доріжки на гнучкому диску.
12. Дати перелік методів захисту від НСК шляхом прив’язки до дистрибутивного носія.
13. Що означає поняття нестандартного форматування і яким чином воно використовується для захисту від НСК?

3.7 Прив'язка до компакт-дисків

Незважаючи на зовнішню несхожість дискет та компакт-дисків, більшість методів створення некопійованих магнітних носіїв були успішно перенесені на компакт-диски.

Більшість використовуваних компакт-дисків не допускає перезапису, а ті, що допускають, не дозволяють змінювати інформацію у довільному місці – можна лише дописати нові дані або знищити все, що було раніше записано. І тому на компакт-дисках не використовують захисти з використанням лічильників установок. Однак, існує багато способів створити диски, які не копіюються стандартними засобами або для яких існують надійні способи відрізнити копію від оригіналу. Перед тим, як розглянути деякі з цих способів, зауважимо, що тримати доступ до вмісту компакт-дисків можна на декількох рівнях.

Самий високий рівень – це *рівень файлової системи*. Дані записуються на диск у певному форматі (наприклад, ISO-9660), і драйвер файлової системи компакт-диска (CD-ROM File System – CDFS) відповідає за те, щоб представити вміст диска у вигляді дерева каталогів і файлів. На цьому рівні доступні такі операції, як отримання списку файлів, відкриття файла за його ім'ям і зчитування з нього інформації.

Наступний рівень – *рівень секторів.* На цьому рівні диск представляється як послідовність секторів, що містять корисні дані, і таблиця, що описує вміст диска (Table Of Content – TOC). Доступні операції зчитування TOC і секторів за їх номерами.

Найнижчий рівень – *рівень команд контролера*. Різні приводи CD-ROM можуть відрізняться за набором команд, але лише на цьому рівні можна отримати найповнішу інформацію, яку видає привод відносно встановленого диска. Використання цього рівня вимагає розробки драйвера.

3.7.1 Найпростіші захисти

Якщо користувач намагається зробити копію диска на рівні файлової системи, він спочатку копіює все дерево каталогів і файлів на вінчестер, а потім за допомогою будь-якої програми для створення компакт-дисків записує диск, що містить скопійовані файли.

1. Програма, прив'язана до компакт-диска, може *перевіряти мітку тома* диска, яка не зберігається при копіюванні файлів на вінчестер. Правда, цю мітку можна створити вручну при створенні образу нового диска. Отже, краще перевіряти серійний номер, який вибирається випадковим чином при створенні диска і не може задаватися користувачем.

2. Можна *зробити декілька посилань на будь-який файл* з різних директорій. При цьому можна домогтися того, щоб сумарний розмір файлів, скопійованих на жорсткий диск, перевищував розмір компакт-диска.

3. Для будь-якого файлу в директорії компакт-диска можна встановити *дуже великий розмір*, що не дозволить прочитати цей файл, оскільки його дані просто не будуть існувати.

Але всі ці методи виявляться безсилими, якщо виконуватиметься копіювання диска на рівні секторів, а не на рівні файлової системи. На даний час посекторне копіювання підтримує майже будь-яка сучасна програма для створення компакт-дисків, наприклад, Nero Burning ROM. І тому для боротьби з посекторним копіюванням використовують інші методи.

3.7.2 Диски великої ємності

Стандартний компакт-диск вміщує 640 Мбайтів. Дещо змінивши параметри диска, можна розмістити на ньому 700 або навіть 800 Мбайтів.

Навіть якщо ніяких спеціальних перевірок не виконується, для створення точної копії диска великого об'єму, виготовленого методом штамповки, необхідна наявність записуваного диска (болванки), що вмістить необхідну кількість інформації, записувальний привод, що може правильно записати такий диск, і відповідна програма для записування. Декілька років тому це могло служити серйозною перешкодою, але зараз знайти необхідні болванки, CD-Writer з підтримкою Overburn (запис дисків великої ємності) і певну програму для записування дисків не складає труднощів.

3.7.3 Відхилення від стандарту записування на диск

Інколи виробники захищених дисків свідомо йдуть на порушення стандарту, який описує, що і як повинно записуватись на диск. Драйвер файловой системи використовує далеко не всю інформацію, яку можна отримати про диск, а тільки ту, яка необхідна для визначення розмірів диска і доступа до окремих файлів. А програми посекторного копіювання намагаються використати максимум інформації і часто відмовляються працювати з диском, якщо зустрічають суперечливі дані.

Однак, зараз велика кількість програм дозволяє ігнорувати деякі порушення стандарту і успішно копіюють більшість дисків, захищених у такий спосіб. Порушення стандарту мають ще один негативний аспект: вони можуть призвести до того, що диск не буде читатися на деяких комп'ютерах або навіть викличе поломку привода CD-ROM.

3.7.4 Фізичні помилки на диску

Якщо диск містить свідомо внесені порушення в області даних, які приводять до помилок зчитування, це не обов'язково є порушенням стандарту – помилки могли виникнути і через природні причини (наприклад, забруднення або механічне пошкодження носія). Таким чином, всі при-

води повинні правильно обробляти ситуації, коли певний сектор не може бути прочитаний. А програма може приймати рішення про легальність диска на основі того, що деякі строго визначені сектори не зчитуються.

Програми посекторного копіювання часто відмовляються продовжувати роботу з диском, якщо не можуть прочитати черговий сектор, а якщо і створюють новий диск, то на ньому непрочитані з оригіналу сектори заповнюються нулями або довільними даними і більше не містять помилок.

Але існують і інші програми, які працюють з контролером напряму і виконують копіювання не на рівні логічних секторів, а фактично на рівні "сирих" даних, які привод отримує з диска. Іноді це називають побітовим копіюванням.

Одним з найпопулярніших інструментів для побітового копіювання компакт-дисків є програма CloneCD (хоча недавно в Інтернеті з'явилося повідомлення про те, що продаж і розповсюдженя цієї програми зупинено у зв'язку з новим законом про захист авторських прав).

Крім того, існують програми, що емулюють компакт-диски. Вони дозволяють використовувати заздалегідь збережений образ компакт-диска для того, щоб відображати привод CD-ROM з установленим на ньому диском. Більшість таких емуляторів вміють передавати не тільки вміст диска, але й усі помилки, які використовуються для запобігання копіюванню і перевірці автентичності компакт-диска.

Однак, існують і системи захисту компакт-дисків, які довгий час досить успішно протистояли програмам побітового копіювання і емуляторам. Прикладом такої програми може бути StarForce.

3.8 Логічна структура диска

Одним із найпопулярніших методів захисту від несанкціонованого копіювання є прив'язка до комп'ютера (до його фізичних характеристик, до встановленого програмного забезпечення, до файлів, до логічної структури, до таблиці розміщення файлів тощо). Тому слід особливо детально розуміти логічну структуру диска, його складові для того, щоб знати, до чого конкретно у комп'ютері можна "прив'язатись".

Дотепер при роботі з дисками ми не зверталися за допомогою до функцій системи, виконуючи всі дискові операції або на рівні команд контролера НГМД, або на рівні функцій BIOS. Операційна система надає набагато зручніші засоби для роботи з диском, ніж звертання до окремих секторів за їх номерами, а також за номером доріжки і головки .

Якщо можливості ОС, які призначені для обслуговування диска, нас влаштовують, то краще користуватися саме цими функціями. У цьому випадку нам не прийдеться піклуватися про дрібниці і ми будете застраховані від деяких помилок. Крім того, наша програма буде менше

залежати від типу і конфігурації комп'ютера, оскільки дисковий драйвер приховує від нас деякі деталі і особливості реалізації системи.

3.8.1 Розділи диску

Персональний комп'ютер, як правило, комплектується одним чи двома НМД. Однак операційна система дозволяє нам розбивати НМД на частини, причому кожна частина буде розглядатися як окремий, логічний диск. Навіщо ж потрібно розбивати диск на логічні диски?

Перші персональні комп'ютери IBM PC були укомплектовані тільки НГМД. Дискети дозволяють зберігати відносно невеликі обсяги інформації, тому поділяти їх на частини немає сенсу. Наступна модель комп'ютера IBM PC/XT мала НМД обсягом 10÷20 Мбайтів. При використанні таких дисків і операційних систем MS-DOS (включно до версії 3.20) у користувачів не виникало ніяких проблем і бажання розбити диск такого малого обсягу на ще менші частини. Проблеми виникли, коли виробники НМД освоїли випуск дисків обсягом 40 Мбайтів і більше. Виявилося, що використовуваний у MS-DOS механізм 16-розрядної адресації секторів не дозволяє застосовувати диски обсягом більшим ніж 32 Мбайти.

Операційна система MS-DOS версії 3.30 запропонувала вихід з положення, що створилося. За допомогою програми fdisk.exe можна було розбити фізичний диск на логічні, кожен з яких не повинен перевищувати за обсягом 32 Мбайти. Згодом у версії 4.00 операційної системи MS-DOS і у версії 3.31 операційної системи COMPAQ DOS зазначене вище обмеження на розмір логічного диска було знято, однак схема поділу фізичного диска на логічні диски цілком збереглася.

Існують і інші причини, за якими може бути корисний поділ одного великого диска на частини:

- у випадку ушкодження логічного диска пропадає тільки та інформація, що знаходилася на цьому логічному диску;
- реорганізація і вивантаження диска маленького розміру виконується швидше ніж великого;
- на одному диску може знаходитися кілька різних операційних систем, розташованих у різних розділах. У ході початкового завантаження можна вказати розділ диска, з якого повинна завантажуватися потрібна нам у даний момент операційна система.

За своєю внутрішньою структурою логічний диск цілком відповідає дискеті, тому спочатку ми згадаємо логічну структуру жорсткого диска, потім зробимо деякі зауваження, що стосуються дискет.

Таким чином, жорсткий диск може бути розбитий на розділи (partition), які можуть використовуватись або однією ОС, або різними, причому в кожному розділі може бути своя файлова система. Але навіть для однієї файлової системи необхідно визначити принаймні один розділ.

Розділи можуть бути 2-х типів:

- первинний (primary) розділ ;
- розширений (extended) розділ.

Максимальне число первинних розділів дорівнює чотирьом (не менше одного). Якщо первинних розділів декілька, то лише один з них може бути активним, і саме завантажувальнику з цього розділу передається керування при вмиканні комп'ютера (решта поки що вважаються прихованими – hidden). Згідно зі специфікаціями, на одному жорсткому диску може бути лише один розширений розділ, який. у свою чергу, може розділятися на підрозділи.

3.8.2 Головний завантажувальний запис та його складові

Найперший сектор жорсткого диска (сектор 1, доріжка 0, головка 0) містить так званий головний завантажувальний запис (MBR – Master Boot Record). Цей запис займає не весь сектор, а тільки його початкову частину.

Сам по собі головний завантажувальний запис є програмою. Ця програма під час початкового завантаження операційної системи з НМД розміщується за адресою 0000h:007Ch, після чого їй передається керування. Завантажувальний запис продовжує процес завантаження операційної системи. MBR – основний засіб завантаження з жорсткого диска, що підтримується BIOS.

У MBR знаходяться три важливих елементи, вміст і структура яких наведені у таблиці 3.7.

Таблиця 3.7 – Структура головного завантажувального запису

Номер елемента MBR	Зсув, байт	Розмір, байт	Вміст	Опис
1	0	446	Програма аналізу РТ і завантаження System Boot-Strap з активного розділу	Non-system bootstrap (NSB) – внесистемний завантажувальник, програма початкового завантаження. Саме вона завантажує наступну, більш складну програму – стартовий сектор операційної системи
2	1Beh	16	Partition 1	Partition Table (PT) – таблиця розділів, яка описує розміщення і характеристику розділів. Якщо РТ пошкоджена, то ОС не буде завантажуватись і будуть недоступними дані, що розташовані на вінчестері
	1CEh	16	Partition 2	
	1DEh	16	Partition 3	
	1EEh	16	Partition 4	
3	1FEh	2	Сигнатура AA55h	Сигнатура MBR. За її наявністю BIOS перевіряє, що перший блок даних завантажений успішно

Таблиця розділів диска

Наприкінці першого сектора жорсткого диска розташовується таблиця розділів диска (Partition Table). Ця таблиця містить чотири елементи, що описують до чотирьох розділів диска. Всі елементи таблиці розділів диска мають однаковий формат - це структура розміром 16 байтів, що відповідає розділу. У структурі знаходиться інформація про розташування і розмір розділу в секторах, а також про призначення розділу (табл.3.8).

Розділи диска бувають *активними* чи *неактивними*. Активний розділ може використовуватися для завантаження операційної системи. Зауважимо, що диск може містити одночасно кілька активних розділів, що можуть належати різним операційним системам.

Таблиця 3.8 – Структура елементів таблиці розділів

Зсув, байт	Розмір, байт	Опис
0	1	Ознака активного розділу (boot indicator): 0 – не активний, 128 (80h) – активний. З цього чи не з цього розділу починається завантаження системи при включенні комп'ютера
1	1	Номер головки, з якої починається перший сектор розділу
2	2	Номер сектора (біти 0÷5) і номер циліндра (біти 6÷15) завантажувального сектора завантажувальника ОС
4	1	Ідентифікатор файлової системи (System ID): 00 – невідома ОС ("пустий" розділ); 01 – FAT12; 04 – FAT16 (менше 32 Мбайтів); 05 – extended; 06 – FAT16; 07 – HPFS/NTFS; 0B – Windows 95 FAT32; 0C - Windows 95 FAT32 LBA; 0E - Windows 95 FAT16 LBA; 0F - Windows 95 Extended; 64,65 – Novell NetWare; 82,83,85 – Linux.
5	1	Номер головки для останнього сектора даного розділу
6	2	Номер сектора (біти 0÷5) і номер доріжки (біти 6÷15) останнього сектора даного розділу
8	4	Відносний номер початкового сектора даного розділу. Для обчислення відносного номера сектора можна використовувати таку формулу: RelSect=(Cyl·Sect·Head)+(Head·Sect)+(Sect-1), де Cyl - номер доріжки, Sect - номер сектора на доріжці, Head - номер головки
12	4	Розмір розділу (в секторах)

Зауважимо, що розділи починаються з парних номерів доріжок, за винятком найпершого розділу. Цей розділ може починатися із сектора з номером 2 нульової доріжки (головка 0), оскільки найперший сектор диска зайнятий головним завантажувальним записом.

3.8.3 Завантажувальний запис та процес завантаження ОС

У найпершому секторі активного розділу розташований завантажувальний запис (Boot Record), який не слід плутати з головним завантажувальним записом (Master Boot Record). Завантажувальний запис зчитується в оперативну пам'ять головним завантажувальним записом, після чого йо-

му передається керування. Завантажувальний запис і виконує завантаження операційної системи. Задача завантажувального запису – виконати завантаження операційної системи. Кожен тип операційної системи має свій завантажувальний запис. Навіть для різних версій однієї і тієї ж операційної системи програма завантаження може виконувати різні дії.

Вміст завантажувального сектора може бути використаний для визначення загальної кількості секторів на логічному диску, для роботи з таблицею розміщення файлів FAT, про яку ми будемо говорити нижче, для визначення інших характеристик логічного диска.

Формат завантажувального запису для FAT12, FAT16

Крім програми початкового завантаження ОС в завантажувальному записі знаходяться параметри, що описують характеристики даного логічного диска. Усі ці параметри розташовуються на самому початку сектора, у його так званій форматованій області. Формат завантажувального запису для FAT16 та FAT32 наведено у табл.3.9.

Таблиця 3.9 - Формат завантантажувального запису для FAT16, FAT32

Зсув	Розмір	Вміст
0	3	Команда JMP xxxx - ближній перехід на програму початкового завантаження
3	8	Назва фірми-виготовлювача операційної системи і версія
11	25	Extended BPB – розширений блок параметрів BIOS
36	1	Фізичний номер пристрою (0 - НГМД, 80h - НМД)
37	1	Зарезервовано
38	1	Символ ')' – ознака розширеного завантажувального запису
39	4	Серійний номер диска (Volume Serial Number)
43	11	Мітка диска (Volume Label)
54	8	Зарезервовано, звичайно містить запис типу 'FAT12 ', що ідентифікує формат таблиці розміщення файлів FAT

Поле зі зсувом 38 завжди містить символ ')'. Цей символ означає, що використовується формат розширеного завантажувального запису.

Серійний номер диска формується під час форматування диска на основі дати і часу форматування. Це поле може бути використане для визначення факту заміни дискети.

Поле завантажувального сектора зі зсувом 11 містить розширений блок параметрів BIOS. Він складається зі звичайного блоку BPB і додаткового розширення (табл.3.10).

Як звичайний, так і розширений блок параметрів BIOS містить байт-описувач середовища media. Цей байт може служити для ідентифікації носія даних і може містити значення, яке характеризує носій даних за кількістю сторін диска і кількістю секторів на доріжці.

Таблиця 3.10 – Формат розширеного блоку параметрів BIOS

Зсув	Розмір	Ім'я поля	Опис
0	2	sect_siz	Кількість байтів в одному секторі диска
2	1	Clustsiz	Кількість секторів в одному кластері
3	2	res_sect	Кількість зарезервованих секторів
5	1	FAT_cnt	Кількість таблиць FAT
6	2	root_size	Макс. кількість дескрипторів файлів у кореневому каталозі диска
8	2	tot_sect	Загальна кількість секторів на носії даних (у розділі)
10	1	Media	Байт, що описує середовище носія даних
11	2	FAT_size	Кількість секторів, зайнятих однією копією FAT
13	2	Sectors	Кількість секторів на доріжці
15	2	Heads	Кількість магнітних головок
17	2	hidden_l	Кількість схованих секторів для розділу, що за розміром менше 32 Мбайтів
19	2	hidden_h	Кількість схованих секторів для розділу, що перевищує 32 Мбайти
21	4	tot_secs	Загальна кількість секторів на логічному диску для розділу, що перевищує за розміром 32 Мбайти

Для FAT32 завантажувальний запис має такі складові (табл.3.11):

- блок параметрів диска (DPB – disk parameter block);
- покажчик на системну програму завантаження (SB - System bootstrap);
- сигнатура завантажувального запису.

Завантаження операційної системи

Завантаження операційної системи з жорсткого диска – багатоступеневий процес, і проходить він у такому порядку:

1. Процедура початкового завантаження (Bootstrap Loader) викликається як переривання INT 19h.
2. Ця процедура викликає перший готовий пристрій з переліку доступних (floppy, hdd, CD, …) і пробує завантажити в ОЗП коротку головну програму-завантажувач, тобто модулі ініціалізації BIOS зчитують головний завантажувальний запис у пам'ять за адресою 0000h:007Ch і передають їй керування. Для вінчестерів – це NSB (Non-system bootstrap).
3. MBR переглядає таблицю розділів PT і знаходить активний розділ. Якщо активних розділів декілька, на консоль виводиться повідомлення про необхідність вибору активного розділу для продовження завантаження. Після того, як активний розділ знайдено, головний завантажувальний запис зчитує найперший сектор розділу в оперативну пам'ять. Цей сектор містить завантажувальний запис, якому головний завантажувальний запис і передає керування. Завантажувальний запис активного розділу виконує завантаження ОС за допомогою System bootstrap, що знаходиться в активному розділі.

Таблиця 3.11 – Формат завантажувального запису для FAT32

Зсув	Розмір	Вміст
0	3	Команда JMP xxxx – ближній перехід на програму початкового завантаження SB (перші два байти – команда безумовного переходу, третій байт – 90h (NOP – немає операції))
3	8	Системний ідентифікатор (назва фірми-виготовлювача операційної системи і версія)
11	2	Розмір сектора в байтах
13	1	Кількість секторів у кластері
14	2	Число зарезервованих секторів (як правило, 32)
16	1	Число копій FAT
17	2	Максимальне число елементів RDIR (для FAT12, FAT16) 0000h – для FAT32
19	2	Число секторів на магн. диску, якщо його розмір менше 32 Мбайтів (FAT16), 0000h – в інших випадках
21	1	Дескриптор носія
22	2	Розмір FAT (у секторах) – для FAT16 0000h – для FAT32
24	2	Число секторів на доріжці
26	2	Число робочих поверхонь
28	4	Число прихованих секторів (використовується для обчислення абсолютного зсуву RDIR і даних), розташованих перед завантажувальним сектором
32	4	Для FAT16: Число секторів на логічному диску, якщо його розмір більше 32 Мбайтів Для FAT32: Число секторів на магнітному лиску
36	1	Для FAT16: Тип логічного диску (00h – floppy, 80h – hard) Для FAT32: Число секторів у FAT-таблиці
37 38	1 1	Для FAT16: 1 байт – пусто (резерв) 1 байт – маркер 29h Для FAT32: Розширені прапорці
39	4	Для FAT16: Серійний номер тома Для FAT32: Номер кластера для початкового кластеру кореневого Каталогу
43	FAT16: 11 8 FAT32: 2 12	 - мітка тома - ім’я файлової системи - номер сектора з резервною копією завантажувального сектора - зарезервовано
…	…	… … …
62 (3Eh)		Systen Bootstrap
510 (1FEh)	2	Сигнатура

4. Ця програма завантажує необхідні файли ОС і передає керування операційній системі.
5. Операційна система виконує ініціалізацію власних програм і апаратних засобів.

Такий процес завантаження операційної системи необхідний з тієї причини, що спосіб завантаження залежить від самої операційної системи. Тому кожна операційна система має свій власний завантажувач. Фіксованим є тільки розташування завантажувального запису – найперший сектор активного розділу.

Таким чином, після включення комп'ютера попадаємо завжди в одну і ту саму ОС. Але ж це не завжди нас влаштовує. Існують спеціальні програми – boot-менеджери (можуть встановлюватись програмою fdisk), які мають можливість:

- вказати, які розділи можуть бути завантажені;
- вказати час на прийняття рішення;
- завантажити якусь певну ОС за замовчуванням.

Цей менеджер також повинен розташовуватись в розділі з файловою системою FAT, а це не завжди нам підходить. Хоча на сучасному етапі розвитку комп'ютерної техніки і комп'ютерних технологій це питання швидко розв'язується.

3.8.4 Логічний номер сектора та переривання INT 25h і INT 26h

Кожна ОС надає можливість роботи з так званими логічними номерами секторів. Це номера секторів всередині логічного диска.

Ми знаємо, що для адресації сектора за допомогою функцій BIOS необхідно вказувати номер доріжки, номер головки і сектора на доріжці. ОС організовує "наскрізну" (рос. "сквозную") нумерацію секторів, при якій кожному сектору логічного диска привласнюється свій номер. Порядок нумерації обраний таким, що при послідовному збільшенні номера сектора спочатку збільшується номер головки, потім номер доріжки. Це зроблено для скорочення переміщень блоку головок при звертанні до послідовних логічних номерів секторів. Нехай, наприклад, у нас є дискета з дев'ятьма секторами на доріжці. Сектор з логічним номером, рівним 1, розташований на нульовій доріжці і для звертання до нього використовується нульова головка. Це найперший сектор на доріжці, він має номер 1. Наступний сектор на нульовій доріжці має логічний номер 2, останній сектор на нульовій доріжці має логічний номер 9. Сектор з логічним номером 10 розташований також на нульовій доріжці. Це теж найперший сектор на доріжці, але тепер для доступу до нього використовується головка 1. І так далі, у міру збільшення логічного номера сектора змінюються номери головок і доріжок.

Для роботи з логічним диском (чи дискетою) на рівні логічних номерів секторів використовують два переривання:

- INT 25h - зчитування сектора за його логічним номером;
- INT 26h - запис сектора за його логічним номером.

Виклик цих переривань має різний формат для різних версій ОС. Зауважимо, що ці переривання залишають у стеці одне слово – старе значення регістра прапорців. Тому після виклику переривання повинна випливати, наприклад, така команда:

```
pop ax.
```

3.8.5 Таблиця розміщення файлів (FAT-таблиця)

У файловий системі FAT дисковий простір будь-якого логічного диску ділиться на дві області (рис.3.3):

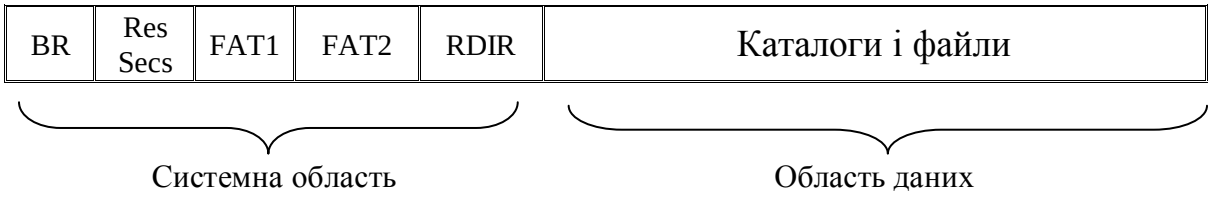


Рисунок 3.3 – Структура логічного диска операційної системи FAT

Системна область логічного диска створюється і ініціалізується при форматуванні, а потім оновлюється. Вона складається з:

- завантажувального запису (BR);
- зарезервованих секторів (ResSecs);
- таблиць розміщення файлів (FAT1, FAT2);
- кореневого каталогу (RDir).

Відразу слідом за завантажувальним сектором на логічному диску знаходяться сектори, що містять таблицю розміщення файлів FAT (File Allocation Table) або FAT-таблицю.

Кластери

Весь диск (точніше тільки область даних) розбивається операційною системою на ділянки однакового розміру, які називаються *кластерами*. Кластер може містити кілька секторів. Для кожного кластера в таблиці FAT є своя індивідуальна комірка, в якій зберігається інформація про використання даного кластера. Іншими словами, таблиця розміщення файлів – це масив, що містить інформацію про кластери, тобто FAT – це карта (образ) диска, в якій описується стан кожної ділянки області даних. Розмір цього масиву визначається загальною кількістю кластерів на логічному диску.

У FAT знаходяться списки кластерів, розподілених файлам. Усі вільні кластери відзначені нулями. Таким чином, якщо файл займає кілька кластерів, то ці кластери зв'язані в список. При цьому елементи таблиці FAT містять номери наступних використовуваних даним файлом кластерів. Кінець списку відзначений у таблиці спеціальним значенням. Номер першого кластера, розподіленого певному файлу, зберігається в елементі каталогу, що описує даний файл.

Програма format.com, призначена для форматування диска і деякі спеціальні програми аналогічного призначення перевіряють диск на наявність дефектних областей. Кластери, що знаходяться в цих дефектних областях, відзначаються у FAT як погані і не використовуються ОС.

Ланцюжок кластерів, розподілених файлу

Отже, FAT – масив інформації про використання кластерів диска, містить списки кластерів, розподілених файлам. Номера початкових кластерів файлів зберігаються в каталогах, про які ми будемо говорити в розділі "Файли і каталоги".

Кореневий каталог

Ім’я файлу	…	Номер 1-го кластера, розподіленого файлу
Autoexec.bat	…	11
Config.sys	…	12
…	…	…

FAT

…	17	13	FFFF	00	00	00	18	FFFF	00	…
Кластер:	11	12	13	14	15	16	17	18	19	

Рисунок 3.4 – Приклад розподілу кластерів для файлів autoexec.bat і config.sys

На рис.3.4 показані фрагменти кореневого каталогу диска C: і елементи FAT для файлів autoexec.bat і config.sys. Тут для файлу autoexec.bat відведено три кластери, а для файлу config.sys – два кластери. Реально ці файли не використовують стільки кластерів, оскільки їх розмір зазвичай невеликий. У каталозі, крім всього іншого, зазначені номери перших кластерів, розподілених цим файлам (відповідно 11 і 12). У своїй 11-й комірці таблиця FAT містить число 17 – номер другого кластера, розподіленого файлу autoexec.bat. Комірка з номером 17 містить число 18.

Це номер третього кластера, що належить файлу autoexec.bat. Остання комірка, що відповідає останньому кластеру, розподіленого цьому файлу, містить спеціальне значення - FFFF. Таким чином, файл autoexec.bat займає три несуміжних кластери з номерами 11, 17 і 18. Що ж стосується файла config.sys, то в нашому прикладі для нього відведено два суміжних кластери з номерами 12 і 13.

Формати FAT-таблиць

Таблиця FAT може мати 12-, 16- та 32-бітовий формат. При цьому в таблиці для збереження інформації про один кластер диска використовується, відповідно, 12, 16 і 32 біти.

Формат FAT-таблиці можна визначити, аналізуючи вміст поля елементів таблиці розділів головного завантажувального запису MBR або із завантажувального запису BR.

12-бітовий формат зручний для дискет з невеликою кількістю секторів – вся таблиця розміщення файлів міститься в одному секторі. Якщо розмір диска такий, що для представлення всіх секторів недостатньо дванадцяти розрядів, можна збільшити розмір кластера, наприклад, до восьми секторів. Однак великий розмір кластера приводить до неефективного використання дискового простору. Це відбувається через те, що мінімальний фрагмент дискової пам'яті, який виділяється файлу, має занадто великий розмір. Навіть для файлу розміром 1 байт виділяється цілий кластер. Виходить, якщо розмір кластера складає 8 секторів, то для збереження одного байта буде використано 4 Кбайти дискової пам'яті (розмір кластера становить 512 байтів).

Використовуючи 12-бітовий формат, не можна було адресувати велику кількість кластерів, тобто, якщо можливий об’єм пам’яті диска був більший за $2^{12} = 4096$ кластерів, то весь простір не використовувався.

16-бітовий формат дозволяє адресувати $2^{16}=65536$ різних 16-розрядних значень, що при розмірі сектора, рівному розміру кластера 512 байтів, складатиме 65536×512 = 32 Мбайти.

32-розрядний формат дозволяє мати 2^{28} кластерів, оскільки у 32-бітовому слові, що використовується для представлення номера кластера, фактично враховується 28 розрядів.

Ідентифікація кластерів у FAT-таблиці

Перший байт FAT-таблиці називається "Описувач середовища" (Media Descriptor). Він має таке саме значення, як і байт-описувач середовища, що знаходиться в завантажувальному секторі логічного диска. Інша частина таблиці FAT складається з 12-, 16- або 32-бітових комірок. Кожна

комірка відповідає одному кластеру диска. Ці комірки можуть містити такі значення:

FAT12	FAT16	FAT32	Опис
000h	0000h	00000000h	Вільний кластер
FF0h - FF6h	FFF0h - FFF6h	FFFFFFF0h - FFFFFFF6h	Зарезервований кластер
FF7h	FFF7h	FFFFFFF7h	Поганий кластер
FF8h - FFFh	FFF8h - FFFFh	FFFFFFF8h - FFFFFFFFh	Останній кластер у списку
002h - FEFh	0002h – FFEFh	00000002h – FFFFFFEFh	Номер наступного кластера в списку

Безпосередній доступ до FAT може знадобитися нам для організації сканування каталогів під час пошуку файлів, для зчитування каталогів як файлів, для організації захисту інформації від несанкціонованого копіювання.

Діставшись до FAT-таблиці, ми зможемо для кожного файлу визначити ланцюжок зайнятих їм кластерів. Для зчитування файлу за допомогою переривання INT 25h нам буде потрібно встановити відповідність між номерами кластерів і номерами секторів, у яких розташовуються ці кластери. Для того, щоб це зробити, необхідно визначити розташування і розмір кореневого каталогу. Тому розглянемо каталоги і файли.

3.8.6 Файли і каталоги

Файлові системи MS-DOS, FAT мають деревоподібну структуру. Кореневий каталог відрізняється від звичайних каталогів тим, що він
для FAT12, Fat16, VFAT:

- розміщається у фіксованому місці логічного диска (відразу за останньою копією FAT);
- має фіксоване число елементів (512);

для FAT32:

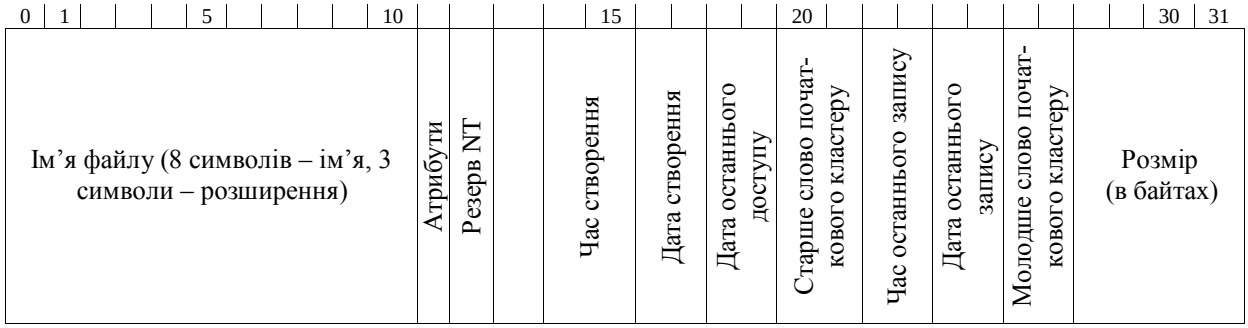
- може знаходитись у довільному місці (початковий кластер кореневого каталогу вказано у завантажувальному записі);
- являє собою ланцюжок кластерів (як і для файлів);
- немає обмеження на довжину каталогу.

У кореневому каталозі розташовуються *дескриптори* – 32-байтові елементи, що містять інформацію про файли й інші каталоги. Для зчитування кореневого каталогу необхідно визначити його розташування і розмір. Структура елементів кореневого каталогу наведена на рис.3.5.

Як бачимо, для довгого імені використовують декілька елементів каталогу. Якщо ім’я займає до 256 символів, то йому може бути виділено до 25 елементів каталогу.

Для коротких імені у Fat12 та FAT16

Для коротких імен у FAT32



Для довгих імен у FAT16 та FAT32

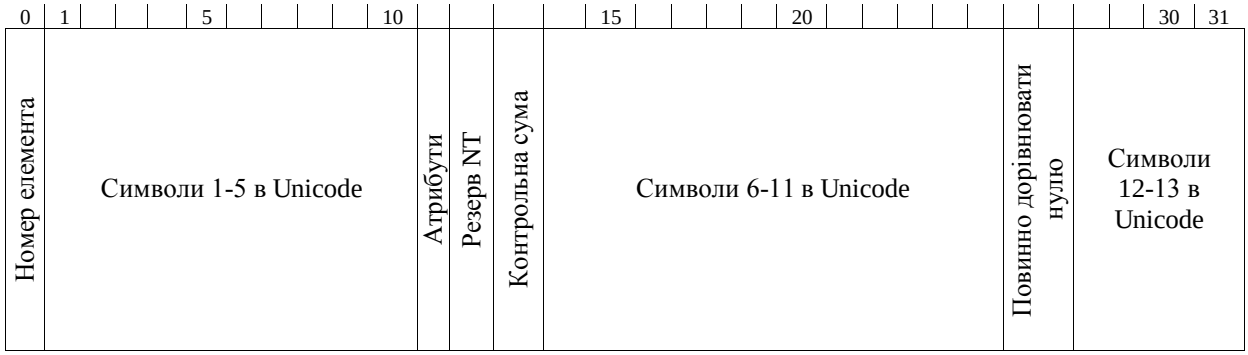


Рисунок 3.5 – Формат елемента кореневого каталогу

Резерв NT – байт, який не використовується в DOS та Windows.

У будь-якому каталозі, крім кореневого, два перших дескриптори мають спеціальне призначення. Перший дескриптор містить у поле імені рядок "**.** " і вказує на утримуючий його каталог. Тобто каталог має посилання сам на себе. Другий спеціальний дескриптор містить у полі імені рядок "**..** " і вказує на каталог більш високого рівня.

Якщо в полі номера першого займаного кластера для дескриптора з ім'ям "**..** " знаходиться нульове значення, це означає, що даний каталог міститься в кореневому каталозі.

Таким чином, у деревоподібній структурі каталогів є посилання як у прямому, так і в зворотному напрямку. Ці посилання можна використовувати для перевірки збереження структури каталогів файлової системи.

Розташування і розмір кореневого каталогу для FAT12 та FAT16

Розглянемо структуру кореневого каталогу на прикладі файлових систем FAT12 та FAT16. Кореневий каталог знаходиться відразу за останньою копією FAT. Кількість секторів, займаних однією копією FAT, знаходиться в блоці параметрів BIOS у завантажувальному секторі (поле FAT_size), а кількість копій FAT – у полі FAT_cnt блоку BPB. Отже, перед кореневим каталогом знаходиться один завантажувальний сектор і FAT_cnt×FAT_size секторів таблиці розміщення файлів FAT.

Розмір кореневого каталогу можна визначити, виходячи зі значення поля rootsize. При форматуванні диска в це поле записується максимальна кількість файлів і каталогів, що можуть знаходитися в кореневому каталозі. Для кожного елемента в каталозі виділяється 32 байти, тому кореневий каталог має довжину 32×rootsize байти.

Кореневий каталог займає суцільну область фіксованого розміру. Розмір кореневого каталогу задається при форматуванні і визначає максимальну кількість файлів і каталогів, що можуть бути в ньому описані.

Для визначення кількості секторів, зайнятих кореневим каталогом, можна скористатися такою формулою:

$$RootSecs = sectsize / (32 \times rootsize),$$

де sectsize – розмір сектора в байтах, який може бути отриманий з відповідного поля завантажувального сектора.

Слідом за кореневим каталогом на логічному диску знаходиться область файлів і підкаталогів кореневого каталогу. Область даних розбита на кластери, причому нумерація кластерів починається з числа 2. Кластеру з номером 2 відповідають перші сектори області даних.

Тепер ми можемо привести формулу, що дозволить нам зв'язати номер кластера з номерами секторів, зайнятих ним на логічному диску:

$$SectNu = DataStart + ((ClustNu - 2) \times clustsize),$$

де SectNu – номер першого сектора, розподіленого кластеру з номером ClustNu; DataStart – початок області даних, обчислюється за формулою:

$$DataStart = ressecs + (FATsize \times FATcnt) + (32 \times rootsize/ sectsize);$$

ClustNu – номер кластера, для якого необхідно визначити номер першого сектора; clustsize – кількість секторів, зайнятих кластером; знаходиться в блоці параметрів BIOS.

Атрибути файлів

Байт атрибутів є приналежністю кожного файлу. Біти цього байта мають значення, наведені у табл.3.12.

Таблиця 3.12 – Значення бітів байта атрибутів файлу

Біт	Опис
0	Файл призначений тільки для зчитування (не можна писати і не можна стирати)
1	Прихований файл (не буде з'являтися в списку файлів за командою DIR)
2	Системний файл. Цей біт звичайно встановлений у файлах, що є складовою частиною операційної системи
3	Даний дескриптор описує мітку диска. Для цього дескриптора поле імені файлу і поле розширення імені файлу повинні розглядатися як одне поле довжиною 11 байтів. Це поле містить мітку диска
4	Дескриптор описує файл, що є підкаталогом даного каталогу
5	Прапорець архівації. Якщо цей біт встановлений у 1, то даний файл не був вивантажений утилітою архівації
6-7	Зарезервовані

Зазвичай файли можуть мати такі атрибути:

00h = 00000000 – звичайні файли (тексти програм, завантажувальні модулі, пакетні файли);

07h = 00000111 – тільки читати, приховані, системні файли. Така комбінація бітів байта атрибутів використовується для файлів операційної системи io.sys, msdos.sys ;

08h = 00001000 – мітка тому. Дескриптор мітки тому може знаходитися тільки в кореневому каталозі логічного диска;

10h = 00010000 – дескриптор, що описує каталог ;

20h = 00100000 - Звичайний файл, до якого не були застосовані програми backup.exe чи xcopy.exe, архівний файл.

Дескриптори вилучених файлів

При видаленні файлу перший байт його імені заміняється на байт E5h (символ "x"). Усі кластери, розподілені файлу, відзначаються у FAT як вільні. Якщо ми щойно видалили файл, його ще можна відновити, оскільки в дескрипторі збереглися всі поля, крім першого байта імені файлу. Але якщо на диск записати нові файли, то вміст кластерів вилученого файлу буде змінений і відновлення стане неможливим.

Час створення або модифікації файлу

Зупинимося докладніше на полях часу і дати чи створення останньої модифікації файлу. ОС оновлює вміст цих полів після будь-якої операції, що змінює вміст файлу – створення файлу, перезапису вмісту файлу, додавання даних у файл чи відновлення вмісту файлу. Після відновлення файлу ОС встановлює біт архівації 5 байта атрибутів у 1. Формат поля часу показаний на рис.3.6.

Старші 5 біт містять значення години модифікації файлу, шість бітів з номерами 5-10 містять значення хвилин модифікації файлу, а у

молодших 5 бітах – значення секунд, поділене на 2. Для того, щоб час відновлення файлу помістився у 16 бітах, довелося піти на зниження точності часу до двох секунд.

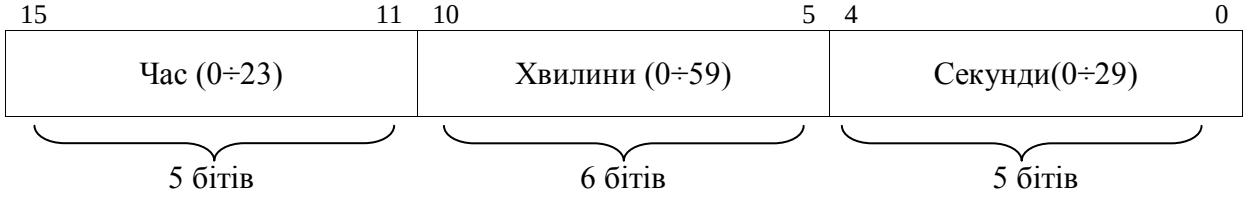


Рисунок 3.6 – Формат поля часу

Дата створення чи зміни файлу

Формат дати відновлення файлу нагадує формат часу (рис.3.7).

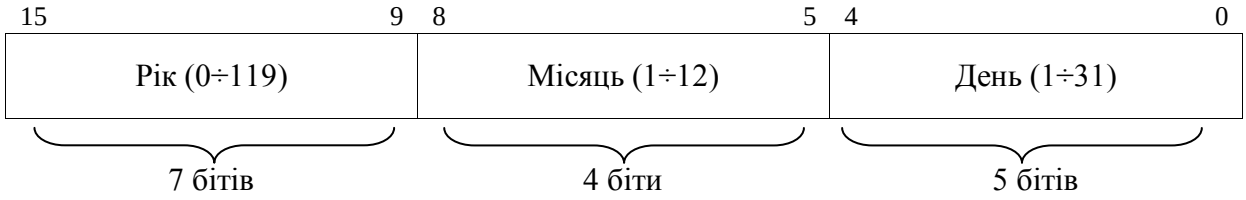


Рисунок 3.7 – Формат поля дати

Для одержання значення року відновлення файлу, необхідно додати до величини, збереженої в старших сімох бітах, число 1980. Поля місяця і дня особливостей не мають, вони відповідають календарній даті.

Довжина файлу

Поле довжини в дескрипторі містить точну довжину файлу в байтах. Для каталогів у поле довжини записане нульове значення. Ми не можемо працювати з каталогом як зі звичайним файлом. Єдиний спосіб прочитати каталог як файл – використання таблиці FAT для визначення ланцюжка займаних каталогом кластерів і зчитування секторів, що відповідають цим кластерам, за допомогою переривання INT 25h .

3.9 Доступ до файлової системи ОС

Оскільки одним з методів захисту програмного забезпечення від копіювання є метод прив’язки до програмного забезпечення комп’ютера, до ключового файлу, до директорії і т.д., то грамотна робота зі складовими файлової системи має неабияке значення.

Сервіс файлової системи доступний програмі через переривання INT 21h. Численні функції цього переривання, що відносяться до файлової системи, можна розбити на групи:

- *одержання довідкової інформації*, що стосується поточного стану дискової системи – поточний диск і поточний каталог, розмір вільного місця на диску, параметри логічного диска і т.д.;
- *робота з каталогами*. Функції цієї групи виконують усі необхідні операції з каталогами – створення, перейменування, знищення каталогів, зміна поточного каталогу тощо;
- *робота з файлами*. Ця група функцій дозволяє програмі виконувати практично будь-які операції над файлами – створення, видалення, зчитування чи запис, перейменування, копіювання, пересилання.

Функції роботи з ФС використовують *ідентифікатори файлу* (file handle). Зміст ідентифікатора файла індексу дуже простий. Для того, щоб приступити до роботи з файлом, програма повинна викликати певну функцію ОС, що відкриває цей файл. Процес відкриття файла полягає в присвоєнні йому певного числа (ідентифікатора) і виконанні деяких інших ініціалізувальних дій. Для виконання будь-яких операцій з файлом програма, викликаючи відповідну функцію, повинна вказати ідентифікатор файла.

Перші п'ять ідентифікаторів зарезервовані операційною системою:

0 – стандартний пристрій введення (клавіатура);
1 – стандартний пристрій виведення (консоль);
2 – стандартний пристрій для виведення повідомлень про помилки (консоль);
3 – стандартний пристрій послідовного введення/виведення, звичайно асинхронний адаптер COM1;
4 – стандартний друкувальний пристрій (звичайно паралельний порт LPT1).

Зарезервовані ідентифікатори завжди доступні програмі і для них не потрібно виконувати операцію відкриття.

Одна з переваг файлових функцій, що використовують ідентифікатори файлів – можливість одночасної роботи з файлами, розташованими в різних каталогах.

Склад функцій, призначених для роботи з файловою системою, досить різноманітний. Тільки в окремих випадках, пов'язаних в основному з організацією захисту інформації від несанкціонованого доступу, нам може знадобитися доступ до диска на більш низькому рівні. Якщо програма використовує для роботи з файлами тільки документовані функції ОС, її робота не буде залежати від апаратних засобів комп'ютера, а також від програми, за допомогою якої були створені розділи диска.

3.9.1 Одержання довідкової інформації

Розглянемо найбільш потрібні, на наш погляд, функції, призначені для одержання довідкової інформації про стан дискової системи.

Функції переривання INT 21h

Переривання INT 21h надає багато можливостей для одержання додаткової інформації, використовуючи для цього наступні функції:

19h – для того, щоб узнати номер поточного диска;

47h – для того, щоб узнати поточний каталог;

3Bh – для установлення поточного каталогу;

1Bh – надається інформація про FAT-таблицю для поточного диска;

1Ch – інформація про FAT-таблицю будь-якого диска (не обов'язково поточного).

36h – розмір вільного місця на диску;

33h – перевірити або встановити прапор переривання за допомогою комбінації клавіш <Ctrl+Break> і, крім того, довідатися номер диска, з якого виконувалося завантаження операційної системи;

54h – дозволяє програмі узнати поточний стан прапорця перевірки запису інформації на диск. Якщо вміст регістра дорівнює 1, після запису сектора операційна система зчитує його для перевірки. Зрозуміло, така перевірка знижує швидкість роботи програми. Якщо після виклику функції регістр AL містить 0, перевірка запису не виконується;

2Eh – для установлення прапорця перевірки запису.

Функції бібліотеки Borland C++

Стандартна бібліотека Borland C++ містить кілька функцій, які полегшують одержання довідкової інформації про стан дискової системи.

_dos_getdiskfree()

Функція _dos_getdiskfree() використовує функцію 36h для одержання інформації про диск. Файл <dos.h> містить такий опис цієї функції:

```
unsigned _dos_getdiskfree (unsigned drive,  struct diskfree_t *diskspace);
```

де drive – задає номер пристрою: 0 – поточний, 1 – A:, і т.д.

Інформація повертається в структурі diskfree_t, що визначена також у файлі <dos.h>:

```
struct diskfree_t
{ unsigned total_clusters;          // Загальна кількість кластерів на диску
  unsigned avail_clusters;          // Кількість вільних кластерів
  unsigned sectors_per_cluster;     // Кількість секторів в одному кластері
  unsigned bytes_per_sector;        // Розмір сектора в байтах
};
```

_dos_getdrive() та _dos_setdrive()

Ці функції використовують для одержання номера поточного диска і для установлення номера поточного диска.

Функція _dos_getdrive() має такий прототип:

void _dos_getdrive (unsigned *drive).

Вона користується функцією 19h переривання INT 21h для одержання номера поточного диска, що записується за адресою, що задається параметром drive. Значення 1 відповідає диску A:, 2 – B:, і т.д.

Функція _dos_setdrive() призначена для установлення поточного диска і може бути використана для визначення загального числа дисків у системі:

void _dos_setdrive (unsigned drive, unsigned *drivecount).

Параметр drive визначає поточний диск (1 – A:, і т.д.). У змінну, адреса якої передається через другий параметр, функція записує загальну кількість логічних дисків, встановлених у системі. Функція _dos_setdrive() використовує функцію 0Eh переривання INT 21h .

3.9.2 Робота з каталогами

Після форматування логічний диск містить кореневий каталог. Якщо диск форматується як системний, у цьому каталозі можуть знаходитись дескриптори файлів операційної системи io.sys, msdos.sys, command.com .

Операційна система надає програмам користувача зручний сервіс для створення, знищення і перейменування каталогів. Ми можемо змінювати структуру каталогів самі, не вдаючись до послуг операційної системи. Однак це варто робити тільки тоді, коли операції з каталогами з якихось причин небажано виконувати з використанням функцій ОС.

Функції переривання INT 21h

Переривання INT 21h має в своєму арсеналі такі функції для цього:
39h – створити каталог;
3Ah – видалити існуючий каталог;
56h – перейменування каталогу або переміщення файлу.

Функції бібліотеки Borland C++

Стандартна бібліотека Borland C++ містить кілька функцій, призначених для роботи з каталогами.

getcwd()

Функція getcwd() призначена для визначення поточного каталогу. Прототип цієї функції описаний у файлі direct.h:

```
char *getcwd (char *path, int n).
```

Перший параметр цієї функції – адреса буфера, у який функція запише рядок, що містить ім'я поточного каталогу. Розмір цього буфера визначається другим параметром.

Якщо як перший параметр вказати NULL, функція динамічно замовить буфер довжиною n байтів з вільної області пам’яті і поверне його адресу. Цю пам'ять згодом необхідно буде звільнити за допомогою функції free(). Функція getcwd() завжди повертає покажчик на буфер, що містить поточний каталог.

mkdir(), rmdir(), chdir()

Для створення і видалення каталогів, зміни поточного каталогу існують функції mkdir(), rmdir(), chdir().

```
int mkdir(const char *name);
int rmdir(const char *name);
int chdir(const char *name).
```

Усі ці функції мають один параметр - шлях каталогу, що має тип (char *). У випадку успішного виконання операції функції повертають 0, при помилці – -1.

rename()

Для перейменування каталогів (і файлів) призначена функція rename():

```
int rename (char *oldname, char *newname).
```

Спосіб її використання очевидний: необхідно задати старе і нове ім'я каталогу. Можна задавати як повний шлях, так і простий – ім'я каталогу або файлу. В останньому випадку операція перейменування виконується над каталогами чи файлами, що знаходяться в поточному каталозі.

Функція може повертати один із приведених нижче кодів помилки:

ENOENT – немає такого файлу або каталогу;

EACCES – немає прав доступу;

EXDEV – інший диск.

Код помилки EXDEV повертається в тому випадку, коли програма вказує різні диски для старого і нового імен файлів або каталогів.

Зауваження: якщо ми задаємо повний шлях у програмі, складеній на C чи C++, слід повторювати символ '\' два рази в рядку шляху. Це потрібно для того, щоб уникнути конфлікту з форматом представлення констант у мові C. Наприклад: ret_code = rename ("c:\\games","c:\\games_new").

3.9.3 Пошук у каталогах

Часто перед програмістом, особливо у випадках захисту від копіювання, стоїть задача визначення вмісту каталогу.

Функції переривання INT 21h

Для виведення на екран вмісту кореневого каталогу й інших каталогів можна використовувати завантажувальний сектор логічного диска і таблицю розміщення файлів. Однак якщо нам не потрібна інформація про номери початкових кластерів файлів і дескриптори вилучених файлів, краще застосувати спеціальні функції, призначені для пошуку файлів у каталогах. Призначені для цього функції 4Eh і 4Fh виконують цю роботу:

4Eh – пошук в каталозі файлу, що відповідає зразку;
4Fh – подальший пошук файлів за зразком.

Функції бібліотеки Borland C++

Стандартна бібліотека Borland C++ містить дві функції, призначені для сканування каталогів.

_dos_findfirst() і _dos_findnext()

Наведемо прототипи цих функцій, описані у файлі <dos.h>:

```
int _dos_findfirst (char *pattern, struct find_t *found, unsigned attr);
                    int _dos_findnext (struct find_t *found).
```

У цих функціях параметр pattern визначає зразок для пошуку файлів, параметр attr (атрибути файлу) використовується як додатковий критерій пошуку. Параметр found являє собою покажчик на структуру, у яку буде записуватися інформація про знайдені файли. Ця структура визначена у файлі <dos.h>:

```
struct find_t
{
  char reserved[21];      // зарезервовано для DOS
  char attrib;            // атрибути файлу
  unsigned wr_time;       // час зміни файлу
  unsigned wr_date;       // дата зміни файлу
  long size;              // розмір файлу в байтах
  char name[13];          // ім'я файлу і розширення
}.
```

3.9.4 Робота з файлами

Розглянемо функції, призначені для створення, відкриття, видалення, перейменування і переміщення файлів.

Функції переривання INT 21h

Переривання INT 21h надає можливість виконувати такі функції для роботи з файлами:

3Ch – створення файлів;

5Bh – перевіряє заданий шлях на предмет наявності зазначеного файлу. Якщо такий файл вже існує, функція повертає ознаку помилки;

5Ah – створити тимчасовий файл. Операційна система не буде автоматично видаляти створений тимчасовий файл після завершення роботи програми або перезавантаження. Програма повинна сама видалити цей файл;

3Dh – відкриття файла;

6Ch – розширені можливості для створення і відкриття файлів (функція є комбінацією функцій 3Dh і 3Ch);

41h – видалення файлу;

56h – перейменування або переміщення файлу (або каталогу).

Функції бібліотеки Borland C++

Стандартна бібліотека Borland C++ містить функції для роботи з файлами. Ці функції можна розділити на дві групи – функції низького рівня і функції введення/виведення потоком (друга група функцій використовує буферизацію і буде розглянута нижче).

Функції низького рівня відображаються на описані вище функції переривання INT 21h (а також на функції цього ж переривання, призначені для зчитування запису, позиціювання і т.д.).

creat()

Для створення файлу можна використовувати функцію creat, описану у файлах <io.h>, <sys\types.h>, <sys\stat.h>, <errno.h>:

int creat (char *filename, int mode).

Перший параметр визначає шлях створюваного файлу і його ім'я. Якщо файл із зазначеним ім'ям існує і не має атрибут ТІЛЬКИ ЧИТАТИ, функція встановлює довжину файлу у нуль. Вміст файлу при цьому знищується.

Другий параметр дозволяє задати атрибути створюваного файлу. Він може мати такі значення: S_IWRITE (запис), S_IREAD (зчитування).

В операційній системі неможливо створити файл, у який можна було б писати, але з якого не можна було б читати інформацію. Тому якщо вказати другий параметр як S_IWRITE, буде створений такий файл, для якого дозволені як операція запису, так і операція зчитування.

Після створення файлу функція creat() відкриває новий файл, повертаючи ідентифікатор файлу або код помилки.

open()

Могутня функція open призначена як для відкриття існуючих файлів, так і для створення нових:

int open (char *filename, int oflag [, int pmode]).

Перший і третій параметри цієї функції аналогічні параметрам функції creat(), причому третій параметр потрібний тільки при створенні нового файлу. Квадратні дужки вказують на то, що цей параметр можна не вказувати.

Параметр oflag може бути результатом логічної операції АБО над такими константами, визначеними у файлі <fcntl.h>:

O_APPEND – при записі у файл інформація буде додаватися в кінець файлу;

O_BINARY – файл відкривається для роботи у двійковому режимі (ігноруються керуючі символи такі, як кінець рядка);

O_CREAT – створюється новий файл і відкривається для записування. Ця константа ігнорується, якщо зазначений у першому параметрі файл вже існує;

O_EXCL – використовується разом з O_CREAT. Якщо зазначений у першому параметрі файл існує, функція поверне ознаку помилки;

O_RDONLY – файл відкривається тільки для зчитування, спроба записування у файл приведе до того, що функція запису поверне ознаку помилки;

O_RDWR - файл відкривається і для зчитування, і для записування;

O_TEXT - файл відкривається в текстовому режимі;

O_TRUNC - існуючий файл відкривається й обрізається до довжини 0 (якщо для цього файлу дозволена операція записування);

O_WRONLY - файл відкривається тільки для записування (у MS-DOS для файлу, відкритого з ознакою O_WRONLY, дозволене виконання операції зчитування).

close()

Для того, щоб закрити файл, відкритий функціями creat() чи open(), потрібно використовувати функцію close():

```
int close (int handle).
```

Як параметр функції передається ідентифікатор файлу, отриманий при відкритті чи при створенні файлу. Функція повертає 0 у випадку успішного закриття файлу або -1 – при помилці.

Код помилки для цієї й іншої функцій стандартної бібліотеки Borland C++ записується в глобальну змінну errno.

3.9.5 Зчитування і записування інформації файлів

Після того як ми відкрили файл, можна виконувати над ним операції зчитування чи записування. При цьому відповідним функціям необхідно передати ідентифікатор файлу. Після завершення операцій зчитування чи записування файл варто закрити.

Функції переривання INT 21h

Існують такі функції переривання INT 21h для роботи з інформацією файлів:
40h – записування даних у файл;
3Fh – зчитування даних з файлу.

Функції бібліотеки Borland C++

write() і read()

Якщо програма складена мовою програмування C чи C++, для записування і зчитування даних вона може скористатися функціями write() і read():

```
int write (int handle, void *buffer, unsigned count);
int read (int handle, void *buffer, unsigned count).
```

Ці функції працюють аналогічно функціям 40h і 3Fh переривання INT 21h. Параметр handle визначає файл, для якого необхідно виконати операцію записування-зчитування. Параметр buffer – покажчик на блок пам'яті, що містить дані для записування або куди необхідно помістити прочитані дані. Кількість записуваних чи прочитаних байтів визначається третім параметром – count.

Після виконання операції функція повертає кількість дійсно записаних чи прочитаних байтів або -1 у випадку помилки.

Якщо ми записуємо або читаємо більше 32 Кбайтів – можемо одержати ознаку помилки, хоча передача даних виконалася правильно. Великі масиви даних бажано записувати вроздріб.

eof()

Для визначення моменту досягнення кінця вихідного файлу в програмі використана функція eof():

int eof(int handle).

Для файлу з ідентифікатором handle ця функція повертає одне з трьох значень:

1 – досягнуто кінець файлу;

0 – кінець файлу не досягнуто;

-1 – помилка, наприклад, неправильно зазначено ідентифікатор файлу.

3.9.6 Позиціювання

Керуючи вмістом файлового покажчика позиції, програма може довільно зчитувати чи перезаписувати різні ділянки файлу, тобто організувати прямий доступ до вмісту файлу. Прямий доступ до файлу може нам знадобитися, наприклад, для створення системи керування базами даних. Переривання 21h використовує для цього функцію:
42h – установлення файлового покажчика у потрібну позицію.

Визначення розміру файлу

Якщо використовувати метод кодування 02h і при цьому задати нульовий зсув, функція установить покажчик на кінець файлу. Ця обставина може бути використане для визначення розміру файлу в байтах.

Функції бібліотеки Borland C++

Стандартна бібліотека Borland C++ містить функції, призначені для керування файловим покажчиком позиції й одержання поточного значення цього покажчика. Це функції lseek() , tell() , filelength() .

lseek()

Функція lseek() працює аналогічно щойно описаній функції 42h. Приведемо її прототип:

long lseek (int handle, long offset, int origin).

Перший параметр визначає файл, для якого виконується операція позиціювання. Параметр offset визначає зсув. Останній параметр задає метод кодування зсуву. Він може приймати такі значення, описані у файлі <stdio.h>:

SEEK_SET – абсолютний зсув від початку файлу;

SEEK_CUR – зсув відносно поточної позиції;

SEEK_END – зсув відносно кінця файлу.

Функція повертає величину поточного зсуву в байтах відносно початку файлу або значення -1 – у випадку помилки. Як і для інших функцій бібліотеки, код помилки знаходиться в глобальній змінній errno.

filelength()

Звичайно, можна використовувати функцію lseek() для визначення розміру файлу або поточної файлової позиції. Однак, для того щоб довідатися розмір файлу, краще скористатися спеціальною функцією filelength() :

long filelength (int handle).

Ця функція повертає розмір файлу в байтах. Файл задається параметром handle. У випадку помилки функція повертає значення -1.

tell()

Для того, щоб визначити поточну файлову позицію, можна використовувати функцію tell():

long tell (int handle);

Ця функція повертає поточну позицію для файлу, визначеного параметром handle, або -1 – у випадку помилки.

3.9.7 Зміна дескриптора файлу

Атрибути файла, час і дата його створення або останньої модифікації, а також розмір файла зберігаються в дескрипторі файла. Дескриптор файла знаходиться в каталозі. Операційна система надає всі необхідні засоби для зміни всіх полів дескриптора файлу, крім номера початкового кластера. Для зміни цього номера вам доведеться працювати з каталогом через таблицю розміщення файлів FAT. Для цього потрібно спочатку прочитати каталог кластерами за допомогою переривання INT 25h, модифікувати потрібні поля і записати каталог назад на диск за допомогою переривання INT 26h.

Функції переривання INT 21h

Переривання INT 21h надає такі функції:
43h – атрибути файлу (отримати і встановити);
57h – час і дата зміни файлу (отримати і встановити).

Функції бібліотеки Borland C++

Стандартна бібліотека Borland C++ містить функції для зчитування і зміни атрибутів файлів, а також часу і дати їх останньої модифікації.

_dos_getfileattr()

Це функція для визначення атрибутів файлу:

unsigned _dos_getfileattr (char *path, unsigned *attrib).

Ця функція одержує атрибути файлу, заданого першим аргументом, і записує байт атрибутів у молодший байт за адресою, зазначеною другим параметром.

У випадку успішного завершення функція повертає 0. У разі помилки вона повертає код помилки, отриманий від операційної системи і встановлює глобальну змінну errno у значення ENOENT, що означає відсутність файлу, зазначеного в параметрі path.

_dos_setfileattr()

Дана функція використовується для зміни атрибутів файлу:

unsigned _dos_setfileattr (char *path, unsigned attrib).

Параметр attrib може приймати такі значення:

_A_ARCH – установлення біта архівації;
_A_HIDDEN – прихований файл;
_A_NORMAL – звичайний файл;
_A_RDONLY – файл, що тільки читається;
_A_SUBDIR – каталог;
_A_SYSTEM – системний файл;
_A_VOLID – мітка диска.

_dos_getftime()

Функція використовується для визначення часу останньої модифікації файлу:

unsigned _dos_getftime (int handle, unsigned *date, unsigned *time).

Перед використанням цієї функції програма повинна відкрити файл. Дата і час записуються за адресою, що вказується, відповідно, другим і третім параметрами.

_dos_setftime()

Дана функція для зміни часу або дати останньої модифікації файлу:

unsigned _dos_setftime (int handle, unsigned date, unsigned time).

Параметри цієї функції аналогічні використовуваним у функції _dos_getftime(), за винятком того, що в якості другого і третього параметрів застосовуються не покажчики, а безпосередні значення дати і часу.

3.9.8 Таблиця відкритих файлів

Для всіх відкритих файлів інформація про них зберігається в спеціальній таблиці. Її адреса знаходиться у полі file_tab векторної таблиці зв'язку. У цій таблиці для кожного відкритого файлу зберігається інформація про кількість ідентифікаторів, зв'язаних з даним файлом, режими відкриття файлу (зчитування, записування і т.д.), слово інформації про пристрій, покажчик на заголовок драйвера, що обслуговує даний пристрій, елемент дескриптора файлу (дата, час, ім'я файлу, номер початкового кластера, розподіленого файлу), номер останнього прочитаного кластера і т.д.

Тепер, коли ми вивчили способи роботи з файлами, має сенс ще раз повернутися до розділу, присвяченого таблиці відкритих файлів.

Ми можемо самостійно експериментувати з цією таблицею. Можна, наприклад, спробувати створити кілька ідентифікаторів для будь-якого файлаі подивитися після цього вміст поля, у якому знаходиться кількість ідентифікаторів, зв'язаних з даним файлом.

Можна спробувати організувати зчитування файлу порціями розміром в один кластер, і при цьому щоразу виводити вміст поля, у якому знаходиться номер останнього прочитаного кластера. Це один з найпростіших способів одержати список кластерів, розподілених даному файлу.

Однак не слід захоплюватись – все, що пов'язано з таблицею файлів, відсутнє у документації з операційної системи. Використовуючи цю таблицю для визначення списку кластерів чи для будь-яких інших цілей, ми ризикуємо втратити сумісність з наступними версіями операційної системи.

3.9.9 Оброблення критичних помилок

Операційна система дозволяє програмам установлювати власний оброблювач критичних помилок апаратури. Вектор 0000h:0090h, що відповідає перериванню INT 24h, містить адресу оброблювача критичних помилок. Цей оброблювач одержує керування від ОС, коли драйвер якого-небудь пристрою виявляє помилку апаратури.

Зауважимо, що оброблювач критичних помилок не викликається при роботі з диском через переривання INT 25h чи INT 26h. Також він не викликається при роботі з диском на рівні переривання INT 13h .

Під час запуску програми операційна система копіює адресу оброблювача в префікс сегмента програми PSP, а після завершення роботи програми відновлює його з PSP.

Стандартний оброблювач виводить на екран повідомлення:

Abort, Retry, Ignore, Fail?

Якщо програма повинна сама обробляти помилки апаратури, вона може установити свій власний оброблювач критичних помилок.

Аналіз регістрів

Коли оброблювач отримує керування, регістри процесора містять інформацію, необхідну для визначення причини і місця появи помилки:

Регістр	Вміст
AH	Інформація про помилку. Біт 0: тип операції: 0 - зчитування, 1 - запис Біти 1,2: область диска, де відбулася помилка: 00 - системні файли; 01 - область FAT; 10 - область каталогу; 11 - область даних. Біт 3: якщо дорівнює 1, можливий вихід з кодом FAIL Біт 4: якщо дорівнює 1, можливий вихід з кодом RETRY Біт 5: якщо дорівнює 1, можливий вихід з кодом IGNORE Біт 6 зарезервований, дорівнює 0 Біт 7 тип пристрою: 0 - диск; 1 - символьний пристрій
AL	Номер диска (якщо біт 7 регістра AH дорівнює 0)
DI	Код помилки (біти 0...7, інші біти не визначені)
BP:SI	Адреса заголовка драйвера пристрою, у якому відбулася помилка

Оброблювач критичних помилок не повинен користуватися функціями з кодами, більшими за 0Ch.

Програма оброблення критичних помилок може вивести на екран повідомлення про помилку і запросити оператора про необхідні дії. Їй дозволено також одержати додаткову уточнюючу інформацію про помилку за допомогою функції 59h переривання INT 21h чи довідатися версію за допомогою функції 30h цього ж переривання.

Додаткова інформація про пристрій, у якому відбулася помилка, може бути отримана з використанням адреси заголовка драйвера пристрою, що передається операційною системою при виклику оброблювача в регістрах BP:SI.

Аналіз стека

Для визначення номера функції, у якій відбулася критична помилка, програма-оброблювач може виконати аналіз стека. Коли оброблювач одержує керування, стек має таку структуру:

Адреса повернення в DOS для команди IRET
IP
CS
FLAGS
Вміст регістрів програми перед викликом INT 21h
AX, BX, CX, DX, SI, DI, BP, DS, ES
Адреса повернення в програму, що викликала функцію DOS
IP
CS
FLAGS

Виконавши аналіз регістра AH, можна визначити номер функції, при виклику якої відбулася помилка, а знаючи вміст інших регістрів – і всі параметри цієї функції.

Код дії

Після виконання всіх необхідних дій програма оброблення критичних помилок повинна повернути в регістрі AL код дії, що повинна виконати операційна система для оброблення даної помилки:

0 – ігнорувати помилку;

1 – повторити операцію;

2 – завершити задачу аварійно, використовуючи адресу завершення, записану у векторі переривання INT 23h;

3 – повернути програмі керування з відповідним кодом помилки.

При відкритті файлів за допомогою функції 6Ch програма може заблокувати виклик оброблювача критичних помилок.

Функції бібліотеки Borland C++

Для складання програми оброблення критичних помилок можна скористатися мовою ассемблера чи функціями стандартної бібліотеки Borland C++ з іменами _dos_getvect(), _dos_setvect(), _chain_intr(). Однак найкраще використовувати спеціально призначені для цього такі функції.

_harderr()

Функція _harderr() призначена для установлення нового оброблювача критичних помилок, вона має такий прототип:

```
void _harderr (void (far *handler)()),
```

де handler – покажчик на нову функцію оброблення критичних помилок.

_hardresume()

Функція _hardresume() і описана нижче функція _hardretn() повинні бути використані в оброблювачі критичних помилок, установленому функцією _harderr().

Функція _hardresume() повертає керування ОС, вона має прототип:

```
_hardresume (int result),
```

де параметр result може мати такі значення (відповідно до необхідних дій):

_HARDERR_ABORT – завершити програму аварійно;

_HARDERR_FAIL – повернути код помилки;

_HARDERR_IGNORE – ігнорувати помилку;

_HARDERR_RETRY – повторити операцію.

Ці параметри описані у файлі <dos.h>.

_hardretn()

Функція _hardretn() повертає керування безпосередно програмі, передаючи їй код помилки, обумовлений параметром error функції:

void _hardretn (int error).

При цьому програма одержує код помилки error після повернення з викликаної нею функції. Якщо помилка відбулася при виконанні функції з номером, більшим за 38h, додатково встановлюється прапорець переносу. Якщо номер функції був меншим ніж зазначене значення, у регістр AL записується величина FFh.

Функція оброблення критичних помилок

handler()

Функція оброблення критичних помилок handler() має такі параметри:

void far handler(unsigned deverror, unsigned errcode, unsigned far *devhdr),

де deverror – код помилки пристрою. Він дорівнює вмісту регістру AX при виклику оброблювача переривання INT 24h; errcode – відповідає вмісту регістру DI – код помилки; devhdr – це покажчик на заголовок драйвера пристрою (переданий у регістрах BP:SI).

3.10 Захист ПЗ методом прив'язки до комп'ютера

Прив'язка програми до дистрибутивної дискети шляхом описаних раніше способів захисту від несанкціонованого копіювання несе в собі велику незручність для користувача, пов'язану з необхідністю роботи тільки при наявності вставленої в дисковод оригінальної дискети.

Набагато зручніше мати необхідний програмний продукт записаним на вінчестері. Тому необхідно, щоб контролююча частина захищуваної програми (КЧЗП) "запам'ятала" свій комп'ютер і потім під час запуску порівнювала наявні характеристики з характеристиками "рідного" комп'ютера. У випадку їх розбіжності можна вважати, що програма незаконно скопійована, і перервати її виконання. Для цього треба знайти якісь параметри, які б індивідуально характеризували кожну обчислювальну систему. Насправді це дуже нетривіальна задача, оскільки відкрита архітектура будови комп'ютерів IBM PC має на увазі їхню знеособленість.

Стійкість до зламу даного методу захисту набагато вища ніж у попередніх, при невеликих витратах на впровадження. Однак через особливості реалізації механізму захисту він є самим незручним для кінцевих користувачів і викликає численні дорікання. Адже програму, захищену

подібним чином, не можна перенести на інший комп'ютер, виникають труднощі з модернізацією і т.п. Застосування такого захисту доцільно у випадках, коли виробник упевнений, що це не відлякує клієнтів.

Серед усіх методів прив’язки до комп’ютера перевірка окремих його підсистем є найбільш придатна і ефективна, оскільки такі характеристики, як швидкодія диска або процесора є досить унікальними для кожного конкретного комп’ютера.

Зазвичай процес установлення захищеного від копіювання програмного продукту полягає в такому:

- в НГМД вставляється дистрибутивна дискета, з неї запускається програма установленя;
- програма установлення зменшує на 1 лічильник виконаних установлень (цей лічильник може знаходитись у нестандартному секторі або в будь-якому іншому місці на дискеті);
- якщо кількість установлень, виконаних з цієї дискети, перевищує максимально допустиме, видається повідомлення про це і робота програми установлення завершується;
- якщо ресурс кількості установлень не вичерпано, виконується копіювання файлів програмного продукту на жорсткий диск та інші необхідні дії;
- виконується налаштування програмного продукту на параметри даного комп’ютера.

Останній крок необхідний для того, щоб захищену від копіювання програму стало неможливо перенести на інший комп’ютер, використовуючи програми копіювання файлів або розвантаження дисків на дискети з подальшим відновленням на жорсткому диску іншого комп’ютера.

Розглянемо, що все-таки можна запропонувати для КЧЗП як характеристики, що могли б перевірятися при роботі програми, що захищається.

Розрізняють такі методи прив’язки до комп’ютера:

- прив’язка до вінчестера;
- прив’язка до BIOS;
- прив’язка до архітектури, набору ПЗ і ін.;
- вимірювання продуктивності апаратури;
- метод з частковим стиранням пам’яті.

3.10.1 Прив’язка до вінчестера

У інформації, що записана на вінчестері, можна знайти унікальні характеристики, наприклад, час створення файлу або каталогу, до номера кластера тощо.

Прив'язка до розташування на вінчестері

Цей метод базується на тому, що при відновленні файлів на іншому комп'ютері вони будуть розташовуватись в іншому місці – в інших секторах вінчестера. При цьому передбачається дослідження розташування якогось досить довгого файлу, записаного на диск під час установлення на предмет визначення його розташування на диску.

Програма установлення, користуючись таблицею розміщення файлів FAT, визначає список кластерів, розподілених файлу і записує цей список в кінець захищуваного файлу або в окремий файл. Можна використати, наприклад, файл конфігурації, призначений для зберігання поточних параметрів програмного пакета. Список кластерів можна зашифрувати, склавши з якимось числом, наприклад, з використанням якоїсь логічної операції (виключне АБО).

Після запуску програмний продукт визначає розташування захищеного файла на диску і порівнює його з записаним при установленні. Якщо розташування змінилось, це означатиме, що запущено незаконну копію.

У цього методу є *недоліки:*

- неможлива оптимізація диска такими програмами, які можуть змінити розташування файлів на диску, наприклад, Norton Speed Disk. Хоча цей недолік можна подолати, використовуючи процедуру деінсталяції, або знімання програмного пакету з диска комп'ютера. Тобто, з диска знімаються файли програмного продукту, потім відповідний лічильник на дистрибутивній дискеті зменшується на 1. Після виконання всіх необхідних дій з диском можна виконати повторне установлення програмного продукту. Таким чином, можна переносити програмний продукт з одного комп'ютера на інший, але не можна його розмножувати на декілька комп'ютерів;
- можна взяти такий самий комп'ютер з таким самим типом жорсткого диску, і за допомогою нескладної програми переписати вміст всіх секторів з диска одного комп'ютера на диск іншого. Подолати цей недолік складніше, оскільки при ідентичності структури дисків і розташування файлів на ньому буде ідентичним, і з цим вже нічого не можна вдіяти.

Записування контрольної інформації на ділянку, що не використовується.

Такий спосіб захисту зробить неможливим копіювання програмного продукту засобами розвантаження дисків, оскільки при відновленні файлів на іншому комп'ютері ділянки, що не використовуються, пропадають. Але знову ж таки лишається можливість використання програм копіювання вмісту диска по секторах.

Використання фізичних дефектів вінчестера

При роботі жорсткого диска (вінчестера) можливе виникнення збійних ділянок на магнітній поверхні. Такі дефектні місця можуть існувати навіть на зовсім новому вінчестері. Номера цих збійних ділянок містяться в FAT (їх ознака - код FF...7). При інсталяції програми, що захищається, на вінчестер у її контролюючу частину записуються їх адреси. У процесі виконання програми здійснюється порівняння адрес збійних ділянок, записаних у КЧЗП і в FAT. У випадку запуску незаконно скопійованої програми буде виявлена розбіжність порівнюваних адрес (перші не будуть складати підмножину других), і відбудеться виконання аварійних дій, передбачених для такого випадку. Але слід зазначити, що зараз досить рідко можна зустріти насправді "поганий" вінчестер.

Розвитком цієї ідеї є метод, у якому деякі справні кластери позначаються як збійні, і в них міститься інформація для КЧЗП. (При копіюванні такі кластери не передаються.)

Використання серійного номера вінчестера.

Контролер IDE, SCSI має спеціальну команду видачі інформації про підключений пристрій. Можна отримати блок 512 байтів інформації про жорсткий диск, якщо в системі є контролер IDE і жорсткий диск. Інформація містить параметри диска і його серійний номер. Деякі ОС не дають доступ до цієї інформації. При форматуванні диска на ньому створюється так званий серійний номер, але його нескладно змінити за допомогою програм типу DiskEdit.

3.10.2 Прив'язка до BIOS

Метод перевірки версії, дати трансляції або контрольних сум BIOS придатний для захисту від копіювання між комп'ютерами, які містять різні версії BIOS. Однак, при купівлі партії комп'ютерів, всі вони майже напевне будуть мати однакову систему BIOS. І тому, хоча цей метод захисту є досить простим, його ефективність відносно невисока.

Перевірка дати створення BIOS

Можна спробувати звузити клас комп'ютерів, на яких можливе функціонування незаконно скопійованої програми. Це досягається, наприклад, шляхом введення перевірки дати створення BIOS, що записана в ПЗУ кожного комп'ютера. Ця дата заноситься в КЧЗП, і в процесі виконання програми, що захищається, здійснюється порівняння дат створення BIOS, записаних у КЧЗП і ПЗП комп'ютера. У випадку, якщо захищена програма

буде незаконно скопійована і встановлена на комп'ютер іншої серії, то КЧЗП це знайде і будуть виконані аварійні дії.

Дата створення BIOS записана в ПЗУ за адресою 000F:FFF5-FFFD і займає 8 байтів.

Прочитати цю дату можна таким чином:

```
. . . .
void far *biosdate;
biosdate = (void far*)MK_FP(0xf0000,0xfff5);
. . . .
```

Перевірка типу комп'ютера

Для забезпечення роботи програми, що захищається, тільки на комп'ютері одного клону треба, щоб вона могла визначати його тип. Така інформація міститься в байті, розташованому за адресою FFFF:000E (F000:FFFE) у ROM BIOS (див. табл.3.1).

Тип комп'ютера можна прочитати безпосередньо за адресами, а можна взяти цей тип, прочитавши таблицю параметрів BIOS за допомогою функції C0h переривання INT 15h (див. далі).

Таблиця конфігурації BIOS

Якщо BIOS створена після 10.1.86, то функція C0h переривання INT 15h повертає адресу таблиці конфігурації BIOS (табл.3.13) в ES:BX. Цю адресу можна прочитати таким чином:

```
. . . .
union REGS rg;
struct SREGS srg;
rg.h.ah = 0xc0;
int86x(0x15, &rg, &rg, &srg);
. . . .
```

Таблиця 3.13 - Таблиця конфігурації BIOS

Зміщення (байт)	Розмір (байт)	Опис
0	2	Розмір таблиці в байтах
2	1	Код моделі комп'ютера
3	1	Додатковий код моделі (підмодель)
4	1	Версія змін BIOS (0 – перша реалізація, 2 – друга і т.д.)
5	1	Байт конфігурації обладнання
6	2	Зарезервовано
8	2	Зарезервовано

При написанні програми для розміщення параметрів цієї таблиці можна підготовити структуру:

```
typedef struct _BIOSINFO_
{
  unsigned size;
  unsigned char model;
  unsigned char submodel;
  unsigned char version;
  unsigned char hardcfg;
  unsigned reserved1;
  unsigned reserved2;
} BIOSINFO.
```

А далі можна звернутися до таблиці таким чином:

```
. . .
BIOSINFO far *binfo;
binfo = (BIOSINFO far*)MK_FP(srg.es, rg.x.bx);
. . .
```

Використання контрольних сум BIOS

При цьому бажано розраховувати також контрольні суми BIOS жорсткого диска і екрана.

3.10.3 Прив'язка до архітектури, до набору ПЗ і т.д.

Прив'язка до програмного забезпечення означає:

- *прив'язку до версії ОС.* Оскільки версія операційної системи, з якою працює користувач на даному комп'ютері, не міняється протягом досить тривалого часу, то її також можна використовувати як параметр для організації перевірки в КЧЗП. Цей контроль бажано проводити в комплексі з іншими методами захисту, наприклад, додатково до перевірки дати створення BIOS, що ще більш звужує клас машин, на яких може працювати програма, що захищається;
- *перевірку наявності драйверів, програм* на вінчестері або в ОЗП;
- *перевірку вмісту файлів "autoexec.bat", "config.sys"* на наявність певної послідовності запусків програм і команд DOS;
- *модифікацію деяких системних файлів*, а потім їх перевірку.

Прив'язка до архітектури, до конфігурації комп'ютера

Прив'язка до архітектури означає, що можна "відловити" число і тип НГМД, число і тип НМД, тип відео-адаптера, об'єм ОЗП, наявність додаткової і/або розширеної пам'яті і т.д.

Для розвитку ідей, викладених вище, розглянемо спосіб, при якому КЧЗП як параметр використовує всю конфігурацію системи. Звичайно треба враховувати, що цей спосіб буде давати помилки і відмовляти у

виконанні законної копії програми, якщо деякі складові обчислювальної системи будуть змінені (наприклад, відключений принтер), або навпаки законна і незаконна копії програми будуть працювати на системах з однаковою конфігурацією.

Розглянемо способи визначення параметрів системи, що далі можуть бути перевірені у КЧЗП.

Способи визначення параметрів системи

Аналіз CMOS пам'яті

Параметри системи, як ми вже зазначали вище, зберігаються у CMOS пам'яті. Вона має 64 однобайтових регістра, пронумерованих від 00 до 3Fh. Призначення більшості регістрів представлені в табл.3.14. Деякі регістри (10-й, 12-й та 14-й) ми вже розглядали.

Таблиця 3.14 - Призначення деяких регістрів CMOS пам'яті

Регістр	Призначення
00h	Секунди
01h	Second Alarm
02h	Хвилини
03h	Minute Alarm
04h	Години
05h	Hour Alarm
06h	День тижня
07h	День місяця
08h	Місяць
09h	Рік
10h	Типи НГМД A: (біти 7-4) і B: (біти 3-0) (0000 - не встановлений; 0001 - 360Кб; 0010 - 1.2Мб, 0011 - 1.44Мб)
12h	Типи НЖМД C: (біти 7-4) і D: (біти 3-0)
14h	Байт встановленої периферії
15h-16h	Розмір пам'яті на основній системній платі (0100h - 256Кб; 0200h - 512Кб; 0280h - 640Кб)
17h-18h	Розмір пам'яті каналу введення/виведення
30h-31h	Розмір додаткової пам'яті понад 1Мбайт

Інформація про периферію зберігається в байті 14h CMOS пам'яті. Для її одержання треба спочатку записати номер регістру (тут 14h) у порт з адресою 70h, а потім прочитати вміст регістру через порт 71h. Значення бітів регістру 14h представлені в табл.3.15.

Комп'ютер може бути укомплектований різними типами дисководів для гнучких дисків і вінчестерів. Розглянемо, за допомогою яких засобів можна одержати інформацію про їхні характеристики крім доступу до CMOS пам'яті.

Таблиця 3.15 - Призначення бітів 14h-го регістра CMOS пам'яті

Біт	Вміст	Значення
7-6	00/01	Один/два НГМД
5-4	11/01/10	Відеорежим (монохромний/кольоровий 40*25/кольоровий 80*25)
3-2	Не використовується	
1	1	Наявність математичного співпроцесора
0	0/1	Немає/є НГМД

Функції переривання INT 21h

Переривання 21h має функції 32h і 36h, пов'язані з визначенням характеристик установлених накопичувачів.

36h – подає поточні відомості про доступний простір на диску, номер якого завантажується в регістр DL. Вона повертає:

- число секторів на один кластер у регістрі AX;
- число незайнятих кластерів – у BX;
- число байтів в одному секторі – у CX;
- число кластерів на диску – у DX.

Тепер легко довідатися, який обсяг диска, номер якого містився в регістрі DL, обчисливши добуток значень регістрів AX, CX, DX.

32h – дозволяє одержати таблицю з параметрами накопичувача, номер якого завантажений у регістр DL. Її адреса буде міститися в регістрах DX:BX. Докладний опис байтів цієї таблиці можна знайти в будь-якому довіднику з переривань

Склад встановленого устаткування перевіряється при завантаженні, і результат перевірки міститься в регістрі статусу. Цей регістр займає два байти, починаючи з адреси 0040:0010, і в табл.3.16 представлені значення його бітів.

Таблиця 3.16 - Регістр складу встановленого устаткування

Біт	Вміст	Значення
0	0/1	Немає/є НГМД
1	0/1	Немає/є математичний співпроцесор 80x87
2-3	11	Оперативна пам'ять 64 Кбайта (у AT не використовується і завжди дорівнює 11)
4-5	11/01/10	Початковий відеорежим (монохромний/кольоровий /кольоровий 80*25)
6-7	00/01/10/11	Число НГМД, якщо біт 0 =1 (відповідно 1,2,3,4)
8		XT/AT не використовується (наявність мікросхеми DMA)
9-11		Число адаптерів комунікації RS232
12	1	Є ігровий адаптер (у AT не використовується)
13		XT/AT не використовується
14-15		Число приєднаних принтерів

Переривання INT 11h

Для доступу до цього регістру крім прямого звертання за адресою можна скористатися перериванням 11h BIOS, що повертає два байти його значень у регістрі AX.

Переривання INT 15h

Ще одне переривання BIOS – 15h через функцію C0h дозволяє одержати адреси в ПЗУ, обумовлені регістрами ES:BX. У п’ятому байті можна знайти інформацію, зазначену у табл.3.17.

Таблиця 3.17 - Вміст байту конфігурації у таблиці параметрів BIOS

Біти	Маска	Значення
7	80h	3-й канал DMA, використовується BIOS
6	40h	другий контролер переривань i8259 встановлений
5	20h	таймер реального часу встановлений
4	10h	int15h/AH=4Fh викликається перед int 9h
3	8h	припустиме чекання зовнішньої події
2	4h	розширення BIOS розміщене в 640 Кбайтах
1	2h	шиною є Micro Channel замість шини ISA
0	1h	резерв
	Зауваження: 1/10/86 XT BIOS повертає некоректне значення 5-го байта	

Для визначення типу дисплея треба перевірити біт номер 1 байта, що знаходиться за адресою 0040:0087. Коли цей біт дорівнює 1 – приєднується монохромний дисплей, а коли він дорівнює 0 – кольоровий.

Функції бібліотеки Borland C++

У мовах C та C++ є спеціальні функції з бібліотеки стандартних функцій <bios.h>, які повертають ціле число, що описує устаткування, яке входить у систему (вони використовують переривання INT 11h).

```
int biosequip (void);
int _bios_equiment(void).
```

Значення, що повертається, інтерпретується набором бітових полів, як представлено в табл.3.16.

3.10.4 Вимірювання продуктивності апаратури

Перевіряючи продуктивність підсистем комп’ютера, можна вимірювати, наприклад:

- *швидкодію комп'ютера,* фіксуючи відлік таймера за час виконання якогось певного фрагмента програми, оскільки час виконання однієї й тієї самої ділянки програми на різних комп'ютерах швидше за все буде відрізнятися, що і дозволить виділити визначену ЕОМ із ряду аналогічних;

- *швидкість роботи оперативного запам'ятовувального пристрою* (ОЗП), тактову частоту, час доступу до ОЗП і т.п. (змінюючи час виконання контрольної ділянки програми);
- *швидкість обертання двигунів дисковода* (через команду READ_ID) або вінчестера, що може бути знайдено, наприклад, замірюванням таймером операції зчитування/записування деякого сектора чи доріжки дискети, як це робить програма “COPYLOCK”;
- *швидкість підсистем введення-виведення*. Ця швидкість, на відміну від швидкості доступу до ОЗП, може не збігатись навіть на машинах однієї партії;
- *швидкість реакції на зовнішні впливи*, тобто часу, необхідного комп'ютеру на відгук після одержання команди зовнішнього пристрою або розмір паузи між видачею команди, наприклад, у контролер накопичувача гнучких магнітних дисків (НГМД), і приходом сигналу готовності (апаратне переривання IRQ7 для НГМД). Вимірювання проводяться за відліком таймера, але можна використовувати і свій програмний лічильник, як це робить BIOS при дискових операціях.

На жаль, надійність таких перевірок і їх стійкість до часу, до коливань температури, вологості повітря і т.п., не гарантована і повинна кожний раз досліджуватись.

3.10.5 Метод з частковим стиранням пам’яті

Сутність методу полягає в маніпулюванні refresh-каналом DMA. Це викличе втрату інформації в частині ОЗП. Справа у тому, що характер реакції DMA (конкретно скорочені області ОЗП, їх нове значення, час зникнення даних і т.п.) специфічний для кожного кристала, хоча, можливо, і може змінюватись в залежності від зовнішніх умов (вік кристала, температура, тощо). Найголовніше при цьому – не пошкодити життєво важливих ділянок ОЗП (наприклад, область DOS, таблицю переривань, саму програму).

Контрольні питання

1. Дати поняття логічних дисків та причини розбиття диску на розділи.
2. Охарактеризувати головний завантажувальний запис, його функції та складові.
3. Що таке таблиця розділів диску, що таке активні та неактивні розділи, первинні та розширені розділи?
4. Яку інформацію несуть елементи розділів диску і як їх можна використати для захисту інформації?

5. Охарактеризувати процес завантаження ОС і роль завантажувального запису.
6. Формат завантажувального запису для FAT12, Fat16. Формат розширеного блоку параметрів BIOS.
7. Формат завантажувального запису для файлової системи FAT32.
8. Логічні номери секторів та переривання для доступу до логічних секторів.
9. Яка роль таблиці розміщення файлів у файловій системі? Які бувають формати fat-таблиць?
10. Що таке кластери? Що являє собою ланцюжок кластерів, розподілених файлу?
11. Охарактеризувати кореневий каталог, розташування і розмір, його особливості для різних файлових систем.
12. Складові кореневого каталогу для різних файлових систем.
13. З чого складаються дескриптори файлів? Якими можуть бути атрибути файлів?
14. Вкажіть методи доступу до файлової системи комп'ютера.
15. Наведіть засоби для отримання довідкової інформації про дискову систему.
16. Охарактеризуйте засоби для роботи з каталогами файлової системи.
17. Дайте характеристику засобам для пошуку інформації у каталогах. Як це можна використати для побудови системи захисту?
18. Програмні засоби для роботи з файлами (записування, зчитування, позиціювання тощо).
19. Засоби для доступу до дескрипторів файлів і можливість їх зміни.
20. Можливості обробки критичних помилок.
21. Дати поняття таблиці відкритих файлів та векторної таблиці зв'язку.
22. Охарактеризувати методи прив'язки до вінчестера як захист від несанкціонованого копіювання.
23. Прив'язка до BIOS як метод захисту від копіювання.
24. Методи доступу до таблиці параметрів BIOS і використання для захисту програм.
25. Способи визначення параметрів системи. Дати коротку характеристику кожному з них.
26. Вимірювання продуктивності апаратури комп'ютера як метод захисту.
27. Охарактеризувати сутність методу захисту ПЗ з частковим стиранням пам'яті.

3.11 Електронні ключі захисту

3.11.1 Поняття електронного ключа, доцільність його використання

На сьогоднішній день електронний ключ – найбільш надійний і зручний метод захисту тиражованого ПЗ середньої і вищої ціновій категорії. Він має високу стійкість до зламу і не накладає обмежень на використання легальної копії програми. Застосування цього методу економічно виправдано для програм вартістю понад $80. Оскільки використання навіть найдешевших електронних ключів збільшує вартість ПЗ на $10-15. Електронний ключ – це те "залізо", яке нам доводиться використовувати без нашого бажання. Крім того, виробники дуже неохоче відкривають його параметри, а його опис досить специфічний, тому електронний ключ дотепер залишається найбільш невідомою частиною комп'ютера для користувача.

Електронними ключами, в основному, захищають так званий "діловий" софт: бухгалтерські і складські програми, правові і корпоративні системи, будівельні кошториси, САПР, електронні довідники, аналітичний софт, екологічні і медичні програми і т.п. Витрати на розроблення таких програм великі, і, відповідно, висока вартість софта, тому збиток від піратського поширення значний. У цьому випадку електронні ключі є оптимальним засобом захисту.

Електронний ключ запобігає незаконному використанню (експлуатації) програми. Часто говорять, що ключ захищає від копіювання, але це не зовсім вірно. Захищену програму можна скопіювати, тільки копія без ключа працювати не буде. Таким чином, копіювання просто не має сенсу.

3.11.2 Будова електронного ключа

Електронний ключ (ЕК) звичайно є невеликим пристроєм (розміром із сірникову коробку), що приєднується до комп'ютера через один з можливих портів (звичайно COM, LPT чи USB порти, хоча є варіанти й у ISA-card виконанні). Найбільш розповсюджений варіант виконання – це LPT-ключ. Такий ключ "прозорий" для принтера й інших LPT-пристроїв, однак насправді це не завжди так.

Ключ складається з плати з мікросхемами (допоміжні елементи, мікроконтролер і пам'ять), вкладеної в пластиковий корпус. Мікроконтролер містить так звану "математику" – набір команд, що реалізують деяку функцію (або функції), яка служить для генерації інформаційних блоків обміну ключа і захищеної програми. Інакше ці блоки називаються "питання і відповіді". Пам'ять електронного ключа містить інформацію про його характеристики, а також дані користувача. Ключ має два роз’єми. За допомогою одного він під’єднується до LPT-порту (паралельного порту)

комп'ютера, інший служить для під’єднання периферійного пристрою. При правильній експлуатації сучасний електронний ключ звичайно не вносить перешкод у роботу принтерів, сканерів і іншої периферії, що під’єднана через нього до паралельного порту.

3.11.3 Класифікація електронних ключів за їх будовою

Електронні ключі надзвичайно різні за своїм виконанням (внутрішні і зовнішні), призначенням, зовнішнім виглядом і т.п. Їх можна також класифікувати за сумісністю з програмними середовищами і типами комп'ютерів, за способом підключення і ступенем складності (функціональності) і т.д. Але ми зупинимось на самих застосовуваних рішеннях.

Найчастіше використовуються електронні ключі, призначені для захисту локальних і мережних Windows і DOS-додатків. Основну масу ключів на сьогоднішній день складають пристрої для паралельного порту. Однак, все більшу популярність набувають USB-ключі, і велика імовірність, що в найближчому майбутньому вони складуть серйозну конкуренцію LPT-ключам.

Для захисту дорогого ПО використовують складні (багатофункціональні) ключі, для захисту більш дешевих програм застосовують і більш прості ключі.

За своєю будовою електронні ключі поділяються на такі групи:

- ключі, що не містять вбудованої пам’яті;
- ключі, що містять тільки пам’ять;
- ключі з таймером;
- ключі на замовленому ASIC-чіпі;
- мікропроцесорні ключі.

Ключі, що не містять вбудованої пам’яті

Ці ключі реалізують тільки функцію вигляду $y=f(x)$. Звичайно в ключах використовують невзаємообернені функції, коли, знаючи значення y, не можна обчислити значення x.

Такі ключі не забезпечують належний ступінь захищеності програмного продукту. Адже тільки наявність пам'яті в доповнення до логічного блоку ключа дозволяє будувати систему захисту будь-якої складності. У пам'яті ключа можна зберігати інформацію, необхідну для роботи програми, списки паролів (власне кажучи, електронний ключ може використовуватися як засіб ідентифікації) і т.п. Обсяг пам'яті більшості сучасних ключів досягає звичайно кілька сотень байтів. Використання ключів без вбудованої пам'яті може бути виправданим тільки для захисту дешевих багатотиражних програм.

Ключі, що містять тільки пам'ять

Це, мабуть, найпростіший вид ключів. Вони мають певне число комірок. з яких дозволено зчитування інформації, у деякі з цих комірок також дозволено записування. Як правило, в комірках, недоступних для записування, зберігається унікальний ідентифікатор ключа.

Ключі з пам'яттю нездатні протистояти емуляції. Досить один раз прочитати всю пам'ять і зберегти її в емуляторі. Після цього правильно емулювати відповіді на всі запити до ключа не становитиме великих труднощів.

Цей клас ключів є морально застарілим. Такі ключі більше не випускаються, але досить велика їх кількість поки що зберігається у кінцевих користувачів ПЗ.

Ключі з таймером

Деякі виробники апаратних ключів пропонують моделі, що мають вбудований таймер. Але для того. щоб таймер міг працювати у той час, коли ключ не під'єднаний до комп'ютера, необхідна наявність вбудованого джерела живлення. Середній час роботи батареї, що живить таймер, складає 4 години, а після її розрядки ключ буде функціонувати неправильно. Можливо, саме через короткий час функціонування ключі з таймером використовуються досить рідко.

Але все ж таймер може посилити захищеність. Так, ключі HASP Time надають можливість дізнаватись про поточний час, що встановлений на вбудованому в ключ годиннику. А захищувана програма може використовувати ключ для того, щоб відслідковувати закінчення тестового періоду.

Гарною комбінацією з точки зору захисту ПЗ є алгоритм, пов'язаний з таймером. Якщо алгоритм може бути деактивований у певний день і час, дуже легко можна реалізовувати демонстраційні версії програм, обмежені за часом.

Ключі на замовленому ASIC-чіпі

ASIC – Applications Specific Integrated Circuit – розповсюджений тип мікросхем. На сьогоднішній день – це найпоширеніший клас ключів. Їхня функціональність визначається конкретним видом ASIC-чіпа. Недоліком таких ключів є, якщо можна так висловитися, "завершеність" конструкції. Діапазон їхніх властивостей обмежений визначеними при створенні мікросхеми рамками. Усі ключі однієї моделі працюють за однаковими алгоритмами (тобто в них містяться функції однакового виду). Така особливість може несприятливо позначатися на ступені стійкості системи

захисту. Адже часто повторювана модель захисту полегшує задачу зламщика. Потенційно такі ключі дуже небезпечні тим, що професійний хакер, "розкусивши" ключ, може з легкістю обдурити будь-яку частину програми і, таким чином, "відв'язати" її від ключа (попросту кажучи – зламати). Однак, з апаратної точки зору такий ключ дуже стійкий, оскільки використовує свій власний замовлений чіп і його дослідження можливе тільки за допомогою спеціальних засобів (електронний мікроскоп і т.д.).

Мікропроцесорні ключі

Цей тип ключів, на відміну від попереднього, має більш гнучкий пристрій. У контролер мікропроцесорного ключа можна "прошивати" програму, що реалізує функції, різні для кожного клієнта. У принципі, будь-який мікропроцесорний ключ легко можна запрограмувати так, що він буде працювати за своїм унікальним алгоритмом. І тому розрізняють ключі:

- *засновані на власному мікропроцесорі* (замовлений MCU-чіп). Цей вид логіки найдорожчий у виробництві, однак дає дуже гарні результати з точки зору захисту. Такі типи ключів найбільш стійкі до зламу з будь-якого погляду (програмного й апаратного), оскільки генерують деяку *непостійну* функцію і досліджувати її з математичної сторони досить складно; досліджувати чіп також досить дорого і довго, оскільки використовувана логіка – невідома;
- *засновані на готовому мікропроцесорі* (PIC-чіп). Ці блоки логіки засновані на готовому мікропроцесорі (наприклад, на PIC 16X8X фірми Microchip Technology Inc.) дають невисоку собівартість і високу надійність із програмної точки зору. Однак, застосування стандартного микрочіпа уможливлює і його стандартне вивчення. Відомі випадки "зняття" бітів захисту від зчитування з такого мікроконтролера і зчитування (з наступним вивченням) усієї мікропрограми, що дає ще і можливість так званого апаратного "клонування" ключів. Тобто покупки чистого PIC-а і зашиття в нього тієї ж мікропрограми й одержання 100%-го аналога конкретного ключа.

Пам'ять, що зустрічається у ключах, складає, як правило, від 0 до 4096 байтів пам'яті. У ній може зберігатися інформація, необхідна програмі для запуску, інформація про кількість ліцензій на запуск, найменування фірми покупця, номер версії програми і т.д.

Електронний ключ – це апаратна частина захисту. Програмну частину електронного ключа складає спеціальне ПЗ для роботи з ключами. У його склад входять інструменти для програмування ключів, утиліти установлення захисту і діагностики, драйвери ключів і ін.

3.11.4 Класифікація електронних ключів за їх програмною частиною

Ключі з невідомим алгоритмом

Ряд сучасних апаратних ключів містять секретну функцію перетворення даних, на якій і базується секретність ключа. Іноді програмісту надається можливість вибрати константи, що є параметрами ключа, але сам алгоритм лишається невідомим.

Перевірка наявності ключа здійснюється таким чином. При розробці захисту програміст робить декілька запитів до алгоритма і запам'ятовує отримані відповіді. Ці відповіді певним чином кодуються у програмі. Під час виконання програма повторює ті самі запитання і порівнює отримані відповіді зі збереженими значеннями, в результаті чого робить висновок про оригінальність ключа.

Ключі з алгоритмами, що мають атрибути

В деяких ключах алгоритми можуть мати додаткові атрибути. Так, у ключах Sential SuperPro алгоритм може бути захищений паролем і починає працювати лише після активації, в ході якої правильний пароль повинен бути переданий ключу. Активація дозволяє розробнику передбачити можливість зміни функціональності ключа на стороні користувача. Тобто програма може мати декілька версій (наприклад. базову, розширену і професійну), а в ключі передбачено лише ті алгоритми, які необхідні для спрацьовування базової версії програми. Якщо ж користувач вирішить перейти до більш повної версії, розробник надішле йому інструкції для активації алгоритмів, що відповідають розширеній або професійній версіям.

Однак, алгоритми, що активуються паролем, опираються на секретність паролів, а не на властивості апаратних ключів. Отже, аналогічний захист може бути реалізований чисто програмними засобами.

Інший тип атрибутів таких алгоритмів, що підтримуються ключами Sential SuperPro, – це лічильники. З активним алгоритмом може бути пов'язаний лічильник, що на початку має ненульове значення. Програма під час кожного запуску (або при виконанні певної операції, наприклад, при експорті даних) викликає спеціальну функцію API-ключа, що зменшує значення лічильника на одиницю. Як тільки лічильник приймає нульове значення, алгоритм деактивується і припиняє працювати.

Слід зазначити, що дана схема захисту не може стати завадою застосуванню емуляторів. Противник може перехопити і передбачити всі спроби зменшення значення лічильника, тоді алгоритм ніколи не буде деактивований. Хоча можна так побудувати роботу алгоритму, щоб запобігти можливості емуляції.

3.11.5 Захист програм за допомогою електронного ключа

Щоб встановити систему захисту за допомогою електронного ключа, необхідно виконати такі кроки:

- *запрограмувати* потрібним чином електронний ключ, тобто внести в його пам'ять інформацію, за якою захищена програма буде ідентифікувати ключ;
- *прив'язати до ключа програму* шляхом установлення автоматичного захисту і/або захисту за допомогою функцій API.

Програмування електронного ключа

Для програмування пам'яті ключа зазвичай використовують спеціальні утиліти. Є ключі, що програмуються за допомогою особливої плати (Crypto-Programmer Card). Але, в остаточному підсумку, це не кращим чином відбивається на вартості ключа. В основному для програмування ключів використовують функції API (Application Programming Interface – інтерфейс прикладного програмування – набір функцій, використовуваних програмами для запитів і виконання низькорівневих сервісів), за допомогою яких зчитується і перезаписується вміст полів пам'яті, редагуються, змінюються чи видаляються самі поля, здійснюється дистанційне програмування ключа. Також утиліти програмування використовуються для налагодження схеми захисту. З їх допомогою перевіряють правильність виконання функцій API, створюють масиви питань і відповідей ключа і т.п. Від якості API залежать: стабільність роботи в різних операційних системах, чіткість і стабільність роботи в локальній мережі, стійкість до програмного зламу, зручність у використанні для програміста, можливість використання широкого спектру мов програмування і т.д.

Щодо сучасних ключів, що працюють у системах типу WinXX, API розділяється на 2 рівні:

- API, що включається в захищену програму,
- драйвер ключа – системний драйвер, що забезпечує взаємодію між програмою і ключем.

Такий "поділ праці" обумовлюється декількома причинами: неможливість виконання операцій введення/виведення на рівні прикладних задач (у захищеному режимі для виконання операцій введення/виведення потрібен рівень привілеїв "0", у той час як всі прикладні задачі виконуються на рівні 3, утруднення розуміння роботи ключа, можливість введення додаткової шифровки (і, таким чином, підвищення стійкості до зламу) і т.д. Крім того, на рівні API відбувається трансформація даних, отриманих із ключа і переданих у програму, і навпаки.

Виробники ключів, як і всі інші виробники, люблять перебільшувати – всі їх криптоалгоритми на 90% винесені в програмну частину, а апаратна частина виконує решту 10% роботи. Воно і зрозуміло - у випадку ускладнення такого пристрою воно зросте в ціні й автоматично зменшиться попит, а стійкість до зламу вони спробують збільшити іншими методами. Крім того, деякі функції API виконує цілком і повністю, навіть не звертаючись до ключа! (Наприклад, ключ Novex містить тільки пам'ять, а вся логіка цілком виконується на програмному рівні).

Варто відзначити, що внутрішності API – це свята святих, і подібно принциповій схемі ключа, є одним з найбільших секретів виробників.

Способи захисту

Є системи захисту, що встановлюються на виконувані програмні модулі (навісний чи автоматичний захист), і системи захисту, що вбудовуються у початковий код програми (захист за допомогою функцій API).

Автоматичний захист

Файл програми, що виконується, обробляється відповідною утилітою, що входить у комплект – ПЗ для роботи з ключами. Як правило, даний спосіб захисту майже цілком автоматизований, процес установлення займає всього декілька хвилин і не вимагає спеціальних знань. Після цього програма виявляється надбудованою на електронний ключ з визначеними параметрами. Утиліти автоматичного захисту звичайно мають безліч сервісних функцій, що дозволяють вибирати різні режими прив'язки програми до ключа і реалізовувати додаткові можливості. Наприклад, такі, як захист від вірусів, обмеження часу роботи і числа запусків програми і т.д.

Однак, варто мати на увазі, що цей спосіб не може забезпечити достатню надійність. Оскільки модуль автоматичного захисту прикріплюється до готової програми, то існує імовірність, що досвідченому хакеру вдасться знайти "точку з'єднання" і зняти такий захист. Надійна утиліта автоматичного захисту повинна мати опції, що утрудняють спроби налагодження і дизасемблювання захищеної програми.

Захист за допомогою функцій API

Цей метод захисту заснований на використанні функцій API, зібраних в об'єктних модулях. Функції API дозволяють виконувати з ключем будь-які операції (пошук ключа з заданими характеристиками, читання і записування даних, підрахунок контрольних сум, перетворення інформації і т.п.). Це дозволяє створювати нестандартні схеми захисту, що підходять для будь-яких випадків. Взагалі, можна сказати, що можливості API-захисту обмежені тільки багатством фантазії розроблювача.

Бібліотеки спеціальних функцій API і приклади їх використання, написані на різних мовах програмування, повинні входити в комплект про-

грамного забезпечення для роботи з ключами. Для установлення захисту необхідно написати виклики потрібних функцій API, вставити їх у вихідний код програми і скомпілювати з об'єктними модулями. У результаті захист виявиться впровадженим глибоко в тіло програми. Використання функцій API забезпечує набагато більш високий ступінь захищеності, ніж автоматичний захист. Практично єдиний "недолік" цього способу захисту, на думку деяких виробників ПЗ, полягає в додаткових витратах на навчання персоналу роботі з API-функціями. Однак без використання API неможливо розраховувати на прийнятну стійкість системи захисту. Тому з метою полегшення життя розроблювачів виробники систем захисту працюють над програмами, що спрощують установлення API-захисту.

Робота електронного ключа

Загалом роботу системи захисту можна представити в такий спосіб:

1. У процесі роботи захищена програма передає електронному ключу інформацію, так зване "питання".
2. Електронний ключ її обробляє і повертає назад – "відповідає".
3. Програма на основі повернутих даних ідентифікує ключ. Якщо він має правильні параметри, програма продовжує працювати. Якщо ж параметри ключа не підходять або він не приєднаний, то програма припиняє свою роботу чи переходить у демонстраційний режим.

3.11.6 Методи зламу та протидії йому

Протистояння розроблювачів систем захисту і зламщиків (хакерів чи крекерів) – це своєрідна “гонка озброєнь”. Постійне удосконалювання засобів і способів зламу змушує розроблювачів захисту безупинно обновляти або винаходити нові засоби і методи захисту, щоб знаходитися на крок вперед. Адже схема, що була ефективною вчора, сьогодні може виявитися непридатною. Приведемо основні методи зламу захисту і способи протидії зламу.

Виготовлення апаратної копії ключа

Цей метод полягає в зчитуванні спеціальними програмними й апаратними засобами вмісту мікросхеми пам'яті ключа. Потім дані переносяться в мікросхему іншого ключа ("болванку"). Спосіб цей досить трудомісткий і може застосовуватися, якщо пам'ять ключа не захищена від зчитування інформації (що було характерно для ключів, що містять тільки пам'ять). До того ж, створення апаратної копії ключа не вирішує проблему тиражування програми, адже вона все рівно залишається прив'язаною, але тільки до іншого ключа. З цих причин виготовлення апаратних копій ключів не одержало широкого поширення

Виготовлення емулятора (програмної копії) ключа

Найпоширеніший і ефективний метод зламу, що полягає в створенні програмного модуля (у вигляді драйвера, чи бібліотеки резидентної програми), що відтворює (емулює) роботу електронного ключа. В результаті захищена програма перестає мати потребу в ключі.

Емулятори можуть відтворювати роботу ключів визначеної моделі, чи ключів, що поставляються з якоюсь програмою, чи одного конкретного ключа. За способом організації їх можна розділити на:

- *емулятори структури*, які відтворюють структуру ключа в деталях (звичайно це універсальні емулятори);
- *емулятори відповідей*, що працюють на основі таблиці питань і відповідей конкретного ключа. У найпростішому випадку для створення емулятора хакер повинен знайти всі можливі правильні питання до ключа і скласти їм відповіді, тобто одержати всю інформацію, якою обмінюються ключ і програма.

Ускладнення протоколу обміну як спосіб запобігання емуляції ЕК

Сучасні ключі мають цілий набір засобів, що запобігають емуляції.

До методів запобігання емуляції електронних ключів відносяться, насамперед, різні варіанти ускладнення протоколу обміну ключа і захищеної програми, а також кодування переданих даних. Використовуються такі основні види захищених протоколів обміну або їх поєднання:

- *плаваючий протокол* – разом з реальними даними передається "сміття", причому згодом порядок чергування і характер як реальних, так і непотрібних даних, змінюється хаотично;
- *кодований протокол* – всі передані дані кодуються;
- *з автоматичною верифікацією* – будь-яка операція записування в пам'ять ключа супроводжується автоматичною перевіркою даних на адекватність;
- додаткове ускладнення протоколу обміну досягається за рахунок *збільшення обсягу переданих зведень і кількості питань до ключа*. Сучасні ключі мають пам'ять, достатню для обробки досить великих обсягів даних. Наприклад, ключ з пам'яттю 256 байтів може обробити за один сеанс до 200 байтів інформації. Складання переліку питань до такого ключа досить трудомістка задача.

Видалення модуля автоматичного захисту

Як уже говорилося раніше, автоматичний захист не має достатнього ступеня стійкості, оскільки не складає з захищеною програмою єдиного цілого. Внаслідок чого «конвертовий захист» можна, при відомих зусиллях, зняти. Існує цілий ряд інструментів, використовуваних хакерами для

цієї мети: спеціальні програми автоматичного зламу, налагоджувачі і дизасемблери. Один зі способів обходу захисту – визначити точку, в якій завершується робота «конверта» і керування передається захищеній програмі. Потім примусово зберегти програму в незахищеному вигляді.

Однак в арсеналі виробників систем захисту є кілька прийомів, що дозволяють максимально утруднити процес зняття захисту. Надійна утиліта автоматичного захисту обов'язково включає опції, що забезпечують:

- протидію автоматичним програмам зламу,
- протидію налагоджувачам і дизасемблерам (блокування стандартних налагоджувальних засобів, динамічне кодування модуля захисту, підрахунок контрольних сум ділянок програмного коду, технологія "божевільного коду" і ін.),
- кодування захищених тіла і оверлеїв програми за допомогою алгоритмів (функцій) перетворення.

Видалення викликів функцій API

Щоб видалити виклики функцій API з вихідного тексту програми, хакери, використовуючи налагоджувачі і дизасемблери, знаходять місця, з яких відбуваються виклики чи точки входу у функції, і відповідним чином виправляють програмний код. Однак при правильній організації API-захисту цей спосіб стає дуже трудомістким. До того ж зламщик ніколи не може бути до кінця упевнений, що правильно і цілком видалив захист, і програма буде працювати без збоїв.

Існує кілька ефективних прийомів протидії спробам видалення чи обходу викликів функцій API:

- використання "божевільного коду": при створенні функцій API їхні команди перемішуються з "сміттям" – непотрібними командами, таким чином код сильно зашумлюється, що утрудняє дослідження логіки роботи функцій;
- використання безлічі точок входу в API: у гарному API-захисті кожна функція має свою точку входу. Для повної нейтралізації захисту зловмисник повинен відшукати всі точки.

3.11.7 Підвищення стійкості захисту

Програмно-апаратний захист надає людині, що її впроваджує, досить велику свободу дій. Навіть при автоматичному захисті можна вибирати серед наявних опцій (і, відповідно, визначати) властивості захищеної програми. А вже при використанні функцій API можна реалізувати будь-яку, навіть саму витончену модель захисту. Таким чином, єдиної і детально розписаної схеми побудови захисту не існує. Однак є багато способів додати захисту додаткову стійкість (нижче приводяться лише деякі з них).

Комбінування автоматичного і API захисту

Як говорилося вище, кожний з цих видів захисту має свої “вузькі” місця. Але разом вони прекрасно доповнюють один одного і складають перешкоду, яку важко подолати навіть досвідченому зломщику. При цьому автоматичний захист відіграє роль своєрідної шкаралупи, зовнішнього рубежу, а захист API є ядром.

Тому рекомендується спочатку вмонтувати виклики API у вихідний код програми, а потім обробити файл, що виконується, за допомогою утиліти автоматичного захисту.

API захист

При API-захисті рекомендується використовувати кілька функцій. Їх виклики необхідно розподілити по коду програми і перемішати змінні цих функцій зі змінними спмої програми. Таким чином, захист API виявляється глибоко впровадженим у програму, і зломщику прийдеться чимало попрацювати, щоб визначити і вибрати всі функції захисту.

Обов'язковим є використання алгоритмів (чи функцій) перетворення даних. Кодування інформації робить безглуздим видалення викликів функцій API, адже при цьому дані не будуть декодовані.

Ефективний прийом ускладнення логіки захисту - це відкладання реакції програми на коди повернення функцій API. У цьому випадку програма приймає рішення про подальшу роботу через якийсь час після одержання кодів повернення. Це змушує зломщика відслідковувати складні причинно-наслідкові зв'язки і досліджувати у налагоджувачу занадто великі ділянки коду.

Автоматичний захист

При автоматичному захисті необхідно задіяти опції захисту від налагоджувальних і дизасемблювальних засобів, опції кодування і перевірки ключів за часом. Корисно також використовувати захист від вірусів. При цьому перевіряється CRC ділянок коду, і виходить, що файл охороняється і від модифікації.

Відновлення системи захисту

Після впровадження системи захисту важливо не забувати про своєчасне відновлення ПЗ для роботи з ключами. Кожен новий реліз – це усунуті помилки, закриті "діри" і нові можливості захисту. Також необхідно постійно відслідковувати ситуацію на ринку систем захисту і, у разі потреби, вчасно змінювати систему захисту на більш прогресивну і надійну.

3.11.8 Можливості електронного ключа

Звичайно, насамперед, ключ призначений для захисту програм. Однак, потенціал сучасного програмно-апаратного захисту настільки великий, що дозволяє застосовувати електронні ключі для реалізації маркетингової стратегії й оптимізації продажів. Ось декілька варіантів такого "нецільового" використання.

Демо-версії

За допомогою електронних ключів можна легко створювати демо-версії програмних продуктів без написання демонстраційного варіанта програми. Можна вільно поширювати копії, заблокувавши або обмеживши деякі можливості програми, що активуються тільки за допомогою електронного ключа. А можна надавати клієнтам повнофункціональну програму в якості пробної ("trial") версії, обмеживши кількість її запусків. А після оплати продовжувати термін користування програмою або ж зовсім знімати обмеження.

Оренда і лізинг

Якщо програма дорога, то часто буває зручно і вигідно продавати її вроздріб чи здавати в оренду. У цьому випадку ключі також зроблять велику послугу. Як це відбувається? Повноцінна робоча копія програми, обмежена за часом роботи, надається клієнту. Після того, як клієнт внесе черговий платіж, термін користування програмою продовжується за допомогою дистанційного перепрограмування пам'яті ключа.

Продаж програми вроздріб

Якщо програма складається з декількох компонентів (наприклад, набір електронних перекладачів – англо-російський, франко-російський і т.п.), то можна включати в комплект постачання всі модулі, але активувати тільки ті з них, за які заплачено. При бажанні клієнт завжди може оплатити компонент програми, що його зацікавив, і який буде активований за допомогою дистанційного програмування ключа.

Відновлення захищеної програми

Виробник випустив нову версію програми. Тепер перед ним виникає проблема відновлення програми в зареєстрованих користувачів. Дистанційне програмування ключів робить цю процедуру швидкою і легкою. При

виході нової версії програми користувачам попередніх версій не потрібно видавати або продавати новий ключ. Потрібно всього лише перепрограмувати ділянку пам'яті наявного ключа і переслати клієнту нову версію (безкоштовно чи за невелику доплату – залежить від маркетингової політики фірми).

Ліцензування в локальних обчислювальних мережах (ЛОМ)

Під ліцензуванням у даному випадку розуміється контроль кількості використовуваних копій програми. Виробникам мережного ПЗ добре знайома ситуація, коли купується одна ліцензійна програма, а в ЛОМ працюють з десятками її копій. У цих умовах електронний ключ стає ефективним засобом, що запобігає запусканню понадлімітних копій програми.

Як здійснюється ліцензування? Припустимо, користувач збирається установити в мережі якусь програму (бухгалтерську, складську і т.п.). При купівлі він вказує число копій програми, що йому необхідно, і одержує відповідну ліцензію. Виробник передає клієнту дистрибутив і належним чином запрограмований ключ. Тепер користувач зможе працювати тільки з тією кількістю копій, за яку заплатив. При необхідності він завжди може докупити відсутні копії, а виробник перепрограмує йому електронний ключ, не виходячи зі свого офісу.

Легко помітити, що сучасна програмно-апаратна система захисту надає безліч сервісних функцій, що дозволяють організувати ефективну маркетингову політику і, природно, одержати додаткову (і дуже відчутну) вигоду.

3.11.9 Огляд та характеристика відомих ключів захисту

HASP (Hardware Against Software Piracy)

Це найбільш розповсюджений ключ захисту від компанії Aladdin Knowledge Systems. В залежності від модифікації може бути:

- мережевим (NetHASP);
- локальним з пам'яттю (MemoHASP);
- локальним без пам'яті (HASP-3);
- з вбудованим годинником (TimeHASP).

Види виконання: LPT, COM, ISA, USB, ADB (Mac), PCMCIA. Обсяг пам'яті – до 496 байтів. Логіка базується на замовленому ASIC-чіпі (логічна матриця).

Оскільки цей ключ, як показала практика, має велику кількість недоліків, він не рекомендується до використання. Крім того, цей ключ відрізняється простотою внутрішньої логіки, і тому для нього досить просто створити емулюючу програмну функцію, навіть не маючи ключа на

руках (оскільки відкриті не тільки алгоритм роботи ключа, але і взаємозв'язок між функцією і системою відкриваючих паролів!).

Типові представники програмного забезпечення, захищені цим типом ключів: всі види програм від фірми "1С", "БЭСТ-4", R-Style Software Labs, ...

Hardlock

Цей тип ключів – один з найкращих ключів, доступних у нас. У 1996 році компанія Aladdin купила невелику німецьку компанію Fast Software Security і тепер продає цей ключ від свого імені.

Випускається в різних модифікаціях, у тому числі у вигляді Hardlock Twin ("непрозорий" ключ: з одного боку – роз'єм LPT-порту, з іншого боку – COM-порту). Пам'ять – до 128 байтів. Логіка базується на власному мікропроцесорі (MCU).

Типові представники програмного забезпечення, захищені цим типом ключів: продукти фірми "ПроИнвестКонсалтинг" (Project Expert 6.xx), "МОТОР-ТЕСТЕР", ER-Mapper 5.x, ...

Обидва ключі вигідно відрізняє від інших конкурентів відмінна документація, розгалужена мережа дилерів по всій Росії, наявність бібліотек практично під усі відомі компілятори для платформ Dos/WinXX.

Sentinel

Виробник – Rainbow Technologies, Inc**.,** найвідоміша компанія в усьому світі, крім СНГ. Контролює близько 55% ринку електронних ключів. Випускає ключі під торговою маркою Sentinel: NetSentinal, USB Sentinal, SentinalSuperPro.

Це звичайні ключі, за своїми параметрами схожі до ключів HASP. Випускаються у виконаннях: LPT, RS232, USB. Мають до 112 байтів енергонезалежної пам'яті, мають до 14 шифрувальних алгоритмів. Базуються на логічному ASIC-чіпі. Є мережні варіанти, варіанти для Macintosh та для Unix-систем.

Досить розвинене і продумане програмне забезпечення дозволяє рекомендувати цей ключ для захистів початкового і середнього рівня.

Типові представники програмного забезпечення, захищені цим типом ключів: "БЕСТ-4", "БЕСТ-ПРО", практично всі західні продукти, систем CAD/CAM.

Guardant

Виробник цих ключів – НВО "Актив". Раніше ключі цієї фірми були більш відомі як ключі "Novex". Ключі "Guardant" є першою вітчизняною розробкою в цій області:

Досить цікаві ключі, представлені трьома видами: Guardant Aptus (Novex), Guardant Stealth (Novex Stealth), Guardant Net (Novex Stealth Net).

Базуються на стандартному PIC 16XXX від Microchip, мають до 256 байтів пам'яті і кілька *перепрограмовуваних* алгоритмів шифрування. Випускаються тільки у варіанті LPT.

З недоліків варто відзначити досить слабке ПЗ і документацію, хоча ключ Novex Stealth за своїми параметрами може легко конкурувати з Hardlock.

Типові представники програмного забезпечення, захищені цим типом ключів: програми "Галактика", "Касатка", "БЕСТ-2+" та інші.

3.11.10 Перспективи розвитку електронних ключів

Поки існує ПЗ і стоїть проблема комп'ютерного піратства, програмно-апаратний захист залишиться актуальним. Яким він буде років через десять, сказати важко. Але вже зараз можна відзначити деякі тенденції, що стають очевидними.

Широкої популярності набувають *USB-ключі* і, швидше за все, вони поступово витиснуть ключі для паралельного порту. Формат ключів змінюється – все більше стають популярними мініатюрні, зручні у використанні ключі USB. У ключах будуть реалізовані більш складні і стійкі алгоритми, збільшиться обсяг пам'яті.

Електронні ключі (трохи інакше улаштовані) починають застосовувати *як засоби ідентифікації* комп'ютерних користувачів, як засіб авторизації при роботі з комп'ютером, для захисту в електронній комерції, для збереження персональних даних і т.п.. Такими ключами-ідентифікаторами в сполученні зі спеціальними програмами можна захищати web-сторінки.

Усе більше будуть використовуватися можливості електронних ключів *для формування маркетингової стратегії* фірм-виробників софта, для просування програмних продуктів.

Народжується безліч компаній (у тому числі і на просторі колишнього СРСР) з альтернативними розробками, що пропонують дуже широкий спектр сервісів.

Поліпшуються використовувані алгоритми, випускаються ключі з вмонтованими шифрувальними системами за відомими, дуже стійкими до криптоанализу і зламу алгоритмами: DES, RSA та іншими.

З іншого боку, при постійно зростаючій кількості ключів і захищених ними програм, стає все складніше їх суміщати, і виробникам потрібно знаходити вихід з цієї ситуації. Хоча цілком можлива ситуація, коли ми в майбутньому побачимо в офісах USB-концентратори, на яких будуть розташовані гірлянди різнобарвних USB-ключів.

3.11.11 Правила використання електронних ключів

1. Не каскадувати ключі на одному комп'ютері (хоча всі виробники в один голос стверджують, що це дуже навіть можна робити) – це не тільки не естетично, але ще і небезпечно як для порту, так і для ключів.
2. У випадку конфліктів рознести режими мережної авторизації по різних протоколах (TCP/IP, NetBios, IPX).
3. Якщо це можливо, то не ставити мережний ключ на сервер – запити до ключа займають багато часу, а стабільність роботи API і мережних менеджерів, у зв’язку з різними антиналагоджувальними і антихакерськими прийомами, залишає бажати кращого. Однак слід пам’ятати – у випадку відключення від локальної мережі машини, що є "сервером ключа", всі інші комп'ютери не зможуть працювати з захищеною програмою (цим особливо страждають продукти серії "1С" через зайві "навороти" у системі захисту).
4. Якщо можна, то розносити ключ і "наворочені" LPT-пристрої типу останніх принтерів від HP, OKI, Lexmark; сканерів і т.п. – “не люблять” вони один одного і заважають один одному.
5. У випадку можливості вибору (наприклад, “БЭСТ-4” поставляється або з ключем HASP, або з ключем Sentinel) – вибирати найбільш розповсюджений ключ – його буде легше і простіше замінити й одержати відповідну підтримку.
6. Не під’єднувати і не від’єднувати ключ при працюючому комп'ютері, не робити це занадто часто (хоча виробники і обіцяють до 100,000 циклів записування/зчитування і до 10 років роботи). Якщо можна, не підключати до Notebook (це знову ж небезпечно для Notebook, у якого система живлення порту досить тендітна).
7. Якщо щось не виходить (особливо з настроюванням роботи в локальній мережі), слід спочатку спробувати почитати документацію на сам ключ від його виробника, оскільки інструкції, що йдуть разом із програмою, захищеною таким ключем, залишають бажати кращого.
8. Слід пам’ятати, що ключ – електронний пристрій і, подібно до інших пристроїв, не любить пилу, статичної електрики, води і т.п., а тому зберігати і використовувати його слід з відповідними запобіжними заходами.

3.12 Захист ПЗ за допомогою опитування довідників

Опитування довідників – ще один спосіб захисту програмного забезпечення від несанкціонованого використання. Він не забезпечує стійкого, довгострокового захисту ПЗ, але може бути легко реалізований і дуже зручний у роботі. При його застосуванні користувач не зобов'язаний працювати з дистрибутивною дискетою, можна і з її копіями.

Робота програм, захищених на основі цього методу, і їх копій поділяється на два етапи. Перший: одержання від користувача інформації з деякого джерела, названого довідником, і порівняння її з еталонною (тобто програма перевіряє, чи дійсно введена інформація знаходиться в довіднику.) Другий: робота самої програми чи її копії. При цьому необхідно зробити все можливе, щоб текст довідника був доступний тільки законному користувачу, оскільки доступ до довідника в цьому методі знімає всі перешкоди для роботи програми. Описана можливість роботи з копіями, недоступна при реалізації інших методів захисту даних, змушує виготовлювачів програмного забезпечення застосовувати цей спосіб у своїх розробках.

У найпростішому випадку довідником є звичайний текстовий файл, що користувач розташовує або на дискеті, або на твердому диску, або має його як роздруківку. Його можна зробити прихованим, якщо він розташований на магнітному носії. Замість текстового файлу можна використовувати службову інформацію, що характеризує визначений клас машин. Однак у цьому випадку прийдеться переборювати додаткові складності при доступі до такої інформації.

Порівняння введеної інформації (пароля) з відповідною інформацією з тексту довідника може бути здійснено двома способами:

1. *Шляхом пошуку інформації в текстовому файлі*, розташованому на магнітному носії, за зазначеними координатами і порівняння її з введеною. Наприклад, програма видає запит: "Введіть п'яте слово третього абзацу сьомої сторінки". Користувач вводить слово. Програма зчитує з тексту довідника п'яте слово третього абзацу сьомої сторінки і порівнює його з введеним.
2. *Шляхом порівняння слова з певної частини тексту*, представленого роздруківкою, *з даними*, що містяться в захищуваній програмі. Цей випадок є як би зворотним до випадку 1. Програма містить інформацію, що в тексті довідника такі-то слова знаходяться в таких-то місцях (наприклад, слово "програма" є п'ятим словом третього абзацу сьомої сторінки, слово "довідник" – перше слово другого абзацу п'ятої сторінки і т.д). У відповідь на запит програми, описаний у першому способі, користувач у тексті довідника за зазначеними координатами шукає слово і вводить його. Програма порівнює отримане слово з інформацією, який вона розташовує.

При невеликому за обсягом довіднику, а також малому запасі опитуваних парольних слів доцільно список усіх парольних слів розмістити зовсім відкрито як файл на спеціальній дискеті. Це дозволить користувачу не тримати в пам'яті інформацію про те, яке слово де знаходиться, і не витрачати час на пошуки потрібного слова в довіднику, а знімати захист самим елементарним образом, випробовуючи кожне слово зі створеного списку. Але якщо ця дискета потрапить у руки до сторонньої людини, то вона буде вдячна нам за можливість просто ознайомитися з вашими секретами.

Однак, перегляд перед кожним запуском програми списку паролів і випробування кожного з них займає досить багато часу. До того ж у таких системах часто виникають помилки через неоднозначність рішення: виділяти чи ні заголовок як окремий абзац. Наприклад, узявши п'яте слово з третього абзацу як пароль, ви не можете запустити програму, оскільки автор виділив заголовок в окремий абзац і необхідне слово знаходиться не в третьому, а в другому абзаці тексту, якщо розглядати його без заголовка.

Через зазначені недоліки описаний вище механізм найчастіше використовують як складову частину багатопрофільних систем захисту від копіювання.

3.13 Введення обмежень на використання ПЗ

В одному зі способів захисту програмного забезпечення від незаконного використання застосовуються обмеження:

- за часом його експлуатації;
- за кількістю запусків;
- за його переміщенням для використання на інших машинах.

Іноді розробники програмного забезпечення накладають обмеження на термін його роботи у визначеного користувача, тобто видають тимчасову ліцензію, після закінчення терміну якої відбувається або самознищення програми, або вона перестає працювати. Ідея, що лежить в основі цього методу, досить прозора – потрібно тільки одержати поточну дату з таймера, а потім, порівнявши з "ліцензійним" терміном використання або дозволити роботу програми, або заборонити. Однак і недолік даного методу захисту ясний – можна просто змінити дату, встановлену в таймері, і тоді програма буде виконуватися незаконно.

Ще одним способом обмежити використання ПЗ є введення лічильника запусків. Звичайно такий лічильник встановлюють:

- у CMOS пам'яті;
- у тілі програми або в спеціальних числових файлах;
- у завантажувальному секторі диска (замість інформації про версію DOS);
- у FAT (у першому елементі таблиці, звичайно заповненому кодом FFFF);

- у резервних секторах;
- у збійних секторах;
- у системному реєстрі;
- після ознаки кінця одного з файлів на локальному диску.

Зменшуючи значення лічильника при кожному запуску контролююча частина програми стежить за ним і при досягненні нульового значення виконує передбачені розроблювачем дії.

Контрольні питання

1. Що таке електронні ключі, в чому доцільність їх використання? Яка будова ЕК?
2. Охарактеризувати види електронних ключів.
3. Дати характеристику способів захисту ПЗ за допомогою електронного ключа.
4. Які методи зламу захистів за допомогою ЕК і способи протидії йому?
5. Як можна підвищити стійкість ЕК до зламу?
6. Охарактеризувати спектр можливостей електронних ключів та перспективи у їх розвитку.
7. Дати коротку характеристику сучасних відомих ключів захисту.
8. Наведіть правила використання електронних ключів та поясніть їх.
9. В чому полягає суть захисту за допомогою опитування довідників і які шліхи його здійснення?
10. Пояснити можливості захисту ПЗ за допомогою введення обмежень на використання програмного продукту.

ЛІТЕРАТУРА

1. Анин Б.Ю. Защита компьютерной информации. – СПб.: БХВ-Петербург, 2000. – 384 с.
2. Вильям Столлингс. Операционные системы. – Москва-Санкт-Петербург-Киев: Вильямс, 2002. – 845 с.
3. Иртегов Д. Введение в операционные системы. – СПБ.: БХВ-Петербург, 2002. – 624 с.
4. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. – М.: Радио и связь, 2001. – 376 с.
5. Михаэль А.Бэнкс. Информационная защита ПК. Пер. с англ. – Киев: Век+, М: Энтроп, СПб.: Корона-Принт, 2001. – 272 с.
6. Домарев В. Безопасность информационных технологий. Методология создания систем защиты. – М.: «Диасофт», 2000. – 368 с.
7. Лукацкий А. Обнаружение атак. – СПб.: БХВ-Петербург, 2000. – 263 с.
8. Щербаков А. Защита от копирования: построение программных средств. – М.: Эдель, 1992. – 334 с.
9. Щербаков А. Разрушающие программные воздействия – М.: Эдель, 1993. – 287 с.
10. Середа С.А. Программно-аппаратные системы защиты программного обеспечения. Материалы Международной конференции аспирантов при Экономической Академии Республики Молдова, 1999.
11. Д.И. Правиков - Ключевые дискеты: разработка элементов защиты от несанкционированного копирования - М.: Радио и связь, 1995.
12. Домашев А.В., Грунтович М.М., Попов В.О., Правиков Д.И., Прокофьев И.В., Щербаков А.Ю. Программирование алгоритмов защиты информации. Учебное пособие. – М.: Нолидж, 2002. – 416 с.
13. Скляров Д. Искусство защиты и взлома информации. – СПб.: БХВ-Петербург, 2004. – 288 с.
14. Глушаков С.В., Хачиров Т.С., Соболев Р.О. – Секреты хакеров. Защита и атака. – Харьков: Фолио, 2004. – 414 с.

Додаток А

Параметри для стандартних типів НМД

Тип	Кількість циліндрів	Кількість головок	Ємність, байт
1	306	4	10.653.696
2 (3)	615	4 (6)	21.411.840 (32.117.760)
4	940	8	65.454.080
5	940	6	49.090.560
6	615	4	21.411.840
7	462	8	32.169.984
8	733	5	31.900.160
9	900	15	117.504.000
10	820	3	21.411.840
11 (12)	855	5 (7)	37.209.600 (52.093.440)
13	306	8	21.307.392
14	733	7	44.660.224
16	612	4	21.307.392
17	977	5	42.519.040
18	977	7	59.526.656
19	1024	7	62.390.272
20	733	5	31.900.160
21	733	7	44.660.224
22	733	5	31.900.160
23	306	4	10.653.696
24	977	5	42.519.040
25	1024	9	80.216.064
26	1224	7	74.575.872
27	1224	11	117.190.656
28	1224	15	159.805.440
29	1024	8	71.303.168
30	1024	11	98.041.856
31	918	11	87.892.992
32	925	9	72.460.800
33	1024	10	89.128.960
34	1024	12	106.954.752
35	1024	13	115.867.648
36	1024	14	124.780.544
37	1024	2	17.825.792
38	1024	16	142.606.336
39	918	15	119.854.080
40	820	6	42.823.680

Навчальне видання

Андрій Веніаминович Дудатьєв
Валентина Аполінаріївна Каплун
Василь Петрович Семеренко

Захист програмного забезпечення

Частина 1

Навчальний посібник

Оригінал-макет підготовлено Каплун В.А.

Редактор О.Д. Скалоцька

Навчально-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК №746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку
Формат 29,7 х 42 1/4
Друк різографічний
Тираж прим.
Зам. №

Гарнітура Times New Roman
Папір офсетний
Ум. друк. арк.

Віддруковано в комп'ютерному інформаційно-видавничому центрі
Вінницького національного технічного університету
Свідоцтво Держкомінформу України
серія ДК №746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ