МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Опорний конспект лекцій
з дисципліни
"Методи та засоби захисту програмного забезпечення"

Тернопіль - 2007

Р.П. Шевчук // Опорний конспект лекцій з дисципліни „Методи та засоби захисту програмного забезпечення”, для студентів напрямку „Комп’ютерні науки”. – Тернопіль, 2007. – 50 с.

Анотація. Особливо важливим аспектом підготовки спеціалістів комп’ютерних наук є успішне засвоєння ними дисципліни “Методи та засоби захисту програмного забезпечення”. 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 програмного забезпечення та автоматизованих систем. Розглянуто моделі розповсюдження програмного забезпечення.

Укладач: **Шевчук Руслан Петрович**, викладач кафедри Комп’ютерних наук ТНЕУ

Відповідальний за випуск: **Дивак Микола Петрович**, д.т.н., професор., завідувач кафедри Комп’ютерних наук ТНЕУ

Рецензенти: завідувач кафедри безпеки інформаційних технологій ТНЕУ, д.т.н., професор **Карпінський М.П.**

доцент кафедри комп’ютерних технологій і систем управління Івано-Франківського національного технічного університету нафти і газу, к.т.н., доцент **Малько О.Г.**

Затверджено на засіданні кафедри Комп’ютерних наук ТНЕУ.
Протокол №___ від _____ 2007 р.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ – Програмне забезпечення
КЗЗ – Комплекс засобів захисту
АС – Автоматизована система
ОС – Операційна система
ЕОМ – Електронно-обчислювальна машина
ЛОМ – Локальна обчислювальна мережа
BIOS – Basic Input/Output System (базова система введення-виведення)
ОЗУ – Оперативний запам’ятовуючий пристрій
ПЗП – Постійний запам’ятовуючий пристрій
ПК – Персональний комп’ютер
CMOS - Complementary Metal Oxide Semiconductor
ОП – Оперативна пам'ять

ВСТУП

Центральною інформаційно-активною ланкою будь-яких комп'ютерних систем є їх математичне, програмне, інформаційне і лінгвістичне забезпечення. Сучасні комп'ютери та комп'ютерні мережі, що володіють потужними обчислювальними, інформаційними і телекомунікаційними можливостями, з своїм складним «внутрішнім технологічним світом», залишаються широким полем діяльності для людини, яка створює і удосконалює самі комп'ютери та завдання, які вони вирішують. При цьому основним технічним інструментом для цього є програмне забезпечення, яке разом з інтелектом людини, його навиками і знаннями, дозволяє створювати складні і часом дивовижні комп'ютерні об'єкти.

Програмне забезпечення сучасних комп'ютерних систем є дуже складним виробом, при створенні і функціонуванні якого активно використовуються автоматизовані засоби його розробки і загальносистемне програмне забезпечення, об'єм і складність якого можуть перевищувати прикладне програмне забезпечення на порядки. Тому в загальному випадку, забезпечення абсолютної якості програмних продуктів є практично нерозв'язним завданням, що є причиною того, що жоден програміст, жодна організація-розробник не гарантує повноцінної надійності створюваного програмного продукту.

Істотних збитків виробникові програмних продуктів наносять такі несанкціоновані дії, як несанкціоноване копіювання програм, їх незаконне розповсюдження та використання. Це завдає значного етичного і матеріального збитку виробникам програмного забезпечення, а часто і легітимним споживачам програмного продукту. Тому багато розробників задаються питанням, чи можна разом з правовим і організаційним забезпеченням процесу розробки і експлуатації програм, здійснити і науково-технічні заходи, що дозволяють захищатися від подібних зловмисних дій.

Таким чином, необхідність внесення до програмного забезпечення захисних функцій на всьому протязі його життєвого циклу від етапу з'ясування задуму на створення програм і їх розробки до етапів випробувань, експлуатації, модернізації і супроводу програм є досить актуальним завданням.

Тому у даному посібнику розглядаються не тільки методологічні основи проблеми захисту програм, але і сучасні методи та засоби забезпечення захисту програм на етапі їх розробки та тестування.

Крім того, досліджуються особливості поведінки програміста – розробника, який може здійснювати широкий набір зловмисних дій. У зв'язку з цим розглянуто сучасні методи та засоби зламу систем захисту програмного забезпечення.

ТЕМА 1. СИСТЕМИ ЗАХИСТУ ІНФОРМАЦІЇ

1.1. Вступ в системи захисту інформації

Захист інформації в системі – діяльність, спрямована на запобігання заподіянню шкоди інтересам власників інформації та її користувачів, а також власників системи.

Криптографічний захист інформації – вид захисту інформації, що реалізується шляхом перетворення інформації з використанням спеціальних (ключових) даних з метою приховування (відновлення) змісту інформації, підтвердження її справжності, цілісності, авторства тощо.

Технічний захист інформації – вид захисту інформації, спрямований на забезпечення за допомогою інженерно-технічних заходів та (або) програмних і технічних засобів конфіденційності, цілісності та доступності інформації, а також унеможливлення її блокування.

Комплексна система захисту інформації – сукупність організаційних, інженерно-технічних заходів, засобів і методів технічного та криптографічного захисту інформації.

Об'єктами захисту в системі є інформація, що обробляється в ній, права власників і користувачів цієї інформації та права власників системи.

Доступ до інформації в системі здійснюється відповідно до порядку доступу до інформації в системі, режиму доступу до неї та інших умов, що визначаються власником інформації.

Власник системи забезпечує захист інформації в системі в порядку та на умовах, визначених у договорі, що укладається ним з власником інформації.

Захист інформації в системі забезпечується:

- запровадженням комплексної системи захисту інформації;
- дотриманням суб'єктами відносин, пов’язаних з обробкою інформації в системі, законодавства України та нормативних документів у сфері захисту інформації в системі;
- використанням засобів електронно-обчислювальної техніки, програмного забезпечення, телекомунікаційного обладнання, а також засобів захисту інформації у системі, які відповідають вимогам законодавства України щодо захисту інформації (наявність сертифіката, експертного висновку тощо).

Для створення комплексної системи захисту інформації використовуються засоби, які мають відповідний сертифікат чи експертний висновок або дозвіл на їх застосування, виданий уповноваженим центральним органом виконавчої влади у сфері криптографічного та технічного захисту інформації.

Комплексна система захисту інформації повинна забезпечувати облік та реєстрацію дій користувачів інформації, пов’язаних з доступом (спробами доступу) до інформації в системі та її обробкою, а також захист від несанкціонованого перекручення або знищення даних про реєстрацію таких дій.

Комплексна система захисту інформації створюється власником системи самостійно за наявності у нього відповідних ліцензій на провадження господарської

діяльності у галузі криптографічного та технічного захисту інформації або із залученням суб'єктів господарювання, які мають такі ліцензії.

Комплексна система захисту інформації створюється органами державної влади та органами місцевого самоврядування для власних потреб за дозволом, що надається уповноваженим центральним органом виконавчої влади у сфері криптографічного та технічного захисту інформації, або із залученням суб'єктів господарювання, які мають відповідні ліцензії.

1.2. Класифікація систем захисту інформації

Мета введення класифікації систем захисту інформації - полегшення задачі співставлення вимог до комплексу засобів захисту (КЗЗ) інформації обчислювальної системи автоматизованої системи (АС).

АС являє собою організаційно-технічну систему, що об'єднує операційну систему (ОС), фізичне середовище, персонал і оброблювану інформацію. Вимоги до функціонального складу КЗЗ залежать від характеристик оброблюваної інформації, самої ОС, фізичного середовища, персоналу і організаційної підсистеми. Вимоги до гарантій визначаються насамперед характером (важливістю) оброблюваної інформації і призначенням АС.

Сьогодні виділяють три ієрархічні класи АС, вимоги до функціонального складу КЗЗ яких істотно відрізняються.

Клас «1» - одномашинний однокористувацький комплекс, який обробляє інформацію однієї або кількох категорій конфіденційності.

Істотні особливості:

- в кожний момент часу з комплексом може працювати тільки один користувач, хоч у загальному випадку осіб, що мають доступ до комплексу, може бути декілька, але всі вони повинні мати однакові повноваження (права) щодо доступу до інформації, яка оброблюється;
- технічні засоби (носії інформації і засоби введення-виведення) з точки зору захищеності відносяться до однієї категорії і всі можуть використовуватись для збереження всієї інформації.

Приклад - автономна персональна електронно-обчислювальна машина (ЕОМ), доступ до якої контролюється з використанням організаційних заходів.

Клас «2» — локалізований багатомашинний багатокористувацький комплекс, який обробляє інформацію різних категорій конфіденційності.

Істотна відміна від попереднього класу — наявність користувачів з різними повноваженнями по доступу і/або технічних засобів, які можуть одночасно здійснювати обробку інформації різних категорій конфіденційності.

Приклад - локальна обчислювальна мережа (ЛОМ).

Клас «3» - розподілений багатомашинний багатокористувацький комплекс, який обробляє інформацію різних категорій конфіденційності.

Істотна відміна від попереднього класу - необхідність передачі інформації через незахищене середовище або, в загальному випадку, наявність вузлів, що реалізують різну політику безпеки.

Приклад - глобальна мережа.

В межах кожного класу АС класифікуються на підставі вимог до забезпечення певних властивостей інформації. З точки зору безпеки інформація характеризується трьома властивостями: конфіденційністю, цілісністю і доступністю. В зв’язку з цим, в кожному класі АС виділяються різні підкласи, для кожного з підкласів кожного класу вводиться деяка кількість ієрархічних стандартних функціональних профілів, яка може бути різною для кожного класу і підкласу АС. Профілі є ієрархічними в тому розумінні, що їх реалізація забезпечує наростаючу захищеність від загроз відповідного типу (конфіденційності, цілісності і доступності). Наростання ступеня захищеності може досягатись як підсиленням певних послуг, тобто включенням до профілю більш високого рівня послуги, так і включенням до профілю нових послуг.

Така класифікація корисна для полегшення вибору переліку функцій, які повинен реалізовувати КЗЗ ОС, проектованої або існуючої АС. Цей підхід дозволяє мінімізувати витрати на початкових етапах створення КЗЗ АС. Проте слід визнати, що для створення КЗЗ, який найповніше відповідає характеристикам і вимогам до конкретної АС, необхідно проведення в повному обсязі аналізу загроз і оцінки ризиків.

1.3. Критерії оцінки захищеності інформації

Критерії є методологічною базою для визначення вимог з захисту інформації в комп'ютерних системах від несанкціонованого доступу; створення захищених комп'ютерних систем і засобів захисту від несанкціонованого доступу; оцінки захищеності інформації в комп'ютерних системах і їх придатності для обробки критичної інформації (інформації, що вимагає захисту).

В процесі оцінки спроможності комп'ютерної системи забезпечувати захист оброблюваної інформації від несанкціонованого доступу розглядаються вимоги двох видів:

- вимоги до функцій захисту (послуг безпеки);
- вимоги до гарантій.

В контексті Критеріїв комп'ютерна система розглядається як набір функціональних послуг. Кожна послуга являє собою набір функцій, що дозволяють протистояти певній множині загроз. Кожна послуга може включати декілька рівнів. Чим вище рівень послуги, тим більш повно забезпечується захист від певного виду загроз. Рівні послуг мають ієрархію за повнотою захисту, проте не обов'язково являють собою точну підмножину один одного. Рівні починаються з першого (1) і зростають до значення *n*, де *n* - унікальне для кожного виду послуг.

Функціональні критерії розбиті на чотири групи, кожна з яких описує вимоги до послуг, що забезпечують захист від загроз одного із чотирьох основних типів.

Конфіденційність. Загрози, що відносяться до несанкціонованого ознайомлення з інформацією, становлять загрози конфіденційності, а саме довірча конфіденційність, адміністративна конфіденційність, повторне використання об'єктів, аналіз прихованих каналів, конфіденційність при обміні (експорті/імпорті).

Цілісність. Загрози, що відносяться до несанкціонованої модифікації інформації, становлять загрози цілісності, а саме довірча цілісність, адміністративна цілісність, відкат і цілісність при обміні.

Доступність. Загрози, що відносяться до порушення можливості використання комп'ютерних систем або оброблюваної інформації, становлять загрози доступності. До доступності відносять такі послуги: використання ресурсів, стійкість до відмов, горяча заміна, відновлення після збоїв.

Спостереженість. Ідентифікація і контроль за діями користувачів, керованість комп'ютерною системою становлять предмет послуг спостереженості і керованості. До спостережності відносять такі послуги: реєстрація, ідентифікація і автентифікація, достовірний канал, розподіл обов'язків, цілісність комплексу засобів захисту, самотестування, автентифікація при обміні, автентифікація відправника (невідмова від авторства), автентифікація одержувача (невідмова від одержання).

Крім функціональних критеріїв, що дозволяють оцінити наявність послуг безпеки в комп'ютерній системі, цей документ містить критерії гарантій, що дозволяють оцінити коректність реалізації послуг. Критерії гарантій включають вимоги до архітектури комплексу засобів захисту, середовища розробки, послідовності розробки, випробування комплексу засобів захисту, середовища функціонування і експлуатаційної документації. В цих Критеріях вводиться сім рівнів гарантій (Г-1, ..., Г-7), які є ієрархічними. Ієрархія рівнів гарантій відбиває поступово наростаючу міру певності в тому, що реалізовані в комп'ютерній системі послуги дозволяють протистояти певним загрозам, що механізми, які їх реалізують, в свою чергу коректно реалізовані і можуть забезпечити очікуваний споживачем рівень захищеності інформації під час експлуатації комп'ютерної системи.

Порядок оцінки комп'ютерної системи на предмет відповідності цим Критеріям визначається відповідними нормативними документами. Експертна комісія, яка проводить оцінку комп'ютерної системи, визначає, які послуги і на якому рівні реалізовані в даній комп'ютерній системі, і як дотримані вимоги гарантій. Результатом оцінки є рейтинг, що являє собою упорядкований ряд (перелічення) буквено-числових комбінацій, що позначають рівні реалізованих послуг, в поєднанні з рівнем гарантій. Комбінації упорядковуються в порядку опису послуг в критеріях. Для того, щоб до рейтингу комп'ютерної системи міг бути включений певний рівень послуги чи гарантій, повинні бути виконані всі вимоги, перелічені в критеріях для даного рівня послуги або гарантій.

1.4. Основні вимоги до розробки систем захисту програмного забезпечення

В залежності від застосування програмного забезпечення (ПЗ) висувають вимоги щодо систем зихисту цього ПЗ, зокрема вимоги до архітектури ПЗ, середовища розробки, послідовності розробки, середовища функціонування, супровідної документації, випробовування.

Вимоги до архітектури забезпечують гарантії того, що КЗЗ у змозі повністю реалізувати політику безпеки і більшою мірою відносяться до архітектури ПЗ. Додержання цих вимог забезпечується Розробником на стадіях проектування КЗЗ. Передусім, вимоги до архітектури покликані забезпечити структурованість КЗЗ відповідно до принципів “хорошого" проектування ПЗ (модульність, інкапсуляція і приховування даних).

Вимоги до середовища розробки забезпечують гарантії того, що процеси розробки і супроводження оцінюваної КС є повністю керованими з боку Розробника.

У процесі розробки ПЗ від Розробника вимагається визначити всі стадії життєвого циклу ПЗ, розробити, запровадити і підтримувати в робочому стані документально оформлені методики своєї діяльності на кожній стадії. Мають бути документовані всі етапи кожної стадії життєвого циклу та їх граничні вимоги (вимоги, що повинні бути виконані раніше, ніж можна приступати до наступного етапу). Крім того, повинні бути документовані стандарти, які використовувались під час розробки ПЗ. Використовувані мови програмування і компілятори мають відповідати вимогам національних, міждержавних або міжнародних стандартів. Розробник повинен розробити, запровадити і підтримувати в дієздатному стані документовані методики керування конфігурацією ПЗ на всіх стадіях її життєвого циклу. При цьому Розробник може розробити і використати систему керування конфігурацією, що найкраще відображає і складність ПЗ.

Вимоги до процесу проектування забезпечують гарантії того, що на кожній стадії розробки (проектування) існує точний опис ПЗ і реалізація ПЗ точно відповідає вихідним вимогам (політиці безпеки). Вимоги до гарантій передбачають наявність чотирьох основних рівнів деталізації ПЗ у процесі її створення: функціональна специфікація, проект архітектури, детальний проект, реалізація.

Вимоги до середовища функціонування забезпечують гарантії того, що ПЗ поставляється замовнику без несанкціонованих модифікацій, а також інсталюється і ініціалізується замовником так, як це передбачається Розробником. Оцінка ПЗ забезпечує гарантії того, що ПЗ правильно реалізує політику безпеки і правильно функціонує, і будується на припущенні, що функціонування ПЗ починається з безпечного стану.

Для того, щоб замовник зміг повною мірою використати послуги безпеки, що надаються ПЗ для реалізації політики безпеки, встановленої в його організації, йому необхідна відповідна документація, в якій були б описані ці послуги і дані вказівки щодо їх використання.

Для демонстрації того, що КЗЗ оцінюваного ПЗ піддавався випробуванням, і доказу повноти цих випробувань Розробник повинен надати документально оформлені результати випробувань.

В плані випробувань повинна бути викладена стратегія випробувань Розробника. План повинен надавати детальний опис всіх тестованих частин КЗЗ. Сюди входять зовнішні інтерфейси КЗЗ, всі політики, привілеї, механізми послуг захисту і специфічних викликів системних функцій, бібліотечного ПЗ і т. ін. План має також відображати середовище випробувань, будь-які особливі умови, що створюються для проведення випробувань, і засоби випробувань. Повинні бути наведені аргументи на користь повноти тестового покриття.

Програма і методика випробувань повинна визначати процедури тестування кожного елемента, визначеного у плані випробувань (наприклад, системних викликів). Для кожного окремого тесту має бути докладно описано використання засобів випробувань, необхідне оточення і особливі умови.

ТЕМА 2. ОСНОВНІ ПОНЯТТЯ ОПЕРАЦІЙНОЇ СИСТЕМИ НЕОБХІДНІ ДЛЯ СТВОРЕННЯ СИСТЕМ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Склад та функції операційної системи

Досить важко дати визначення терміну «операційна система» внаслідок того, що ОС виконує дві мало пов'язані функції: надання користувачу-програмісту розширеної машини і підвищення ефективності використання комп'ютера шляхом раціонального керування його ресурсами. В більшості випадків під ОС розуміють комплекс програм, функціями якого є контроль за використанням і розподілом ресурсів обчислювальної системи.

ОС виникли на певному етапі розвитку ЕОМ. Обсяг їх функцій з часом все збільшувався у відповідності з ростучою потребою по ефективному використанню ЕОМ.

ЕОМ першого покоління (на електронних лампах) не мали ніякого програмного забезпечення, все програмування було на рівні користувача, тобто ЕОМ сприймалася буквально як програмно-керований обчислювальний автомат. З другим поколінням ЕОМ (на дискретних напівпровідникових приладах) народилось і системне програмування, тобто створення бібліотек програм, трансляторів з різних мов програмування і, нарешті, створення моніторних систем, які керують самим процесом проходження задач через ЕОМ і забезпечують той чи інший рівень автоматизації функцій, які на ЕОМ першого покоління виконувала людина-оператор. Ці моніторні системи стали попередниками ОС для ЕОМ третього покоління (на інтегральних напівпровідникових приладах).

До основних функцій ОС відносять:

- ініціація та завершення виконання задач користувачів;
- керування ходом їх виконання;
- обробка різних виключних ситуацій, що виникають в процесі роботи (наприклад, обробка помилок в програмі чи апаратурі);
- розподіл ресурсів ЕОМ між задачами;
- забезпечення можливості використання наявних програмних засобів загального користування (файли, архіви, транслятори і т.п.);
- взаємний захист програм та інформації, які належать різним користувачам;
- оптимізація паралельної роботи пристроїв ЕОМ з метою досягнення найвищої продуктивності;
- реєстрація та облік всієї виконуваної роботи, що дозволяє, зокрема, виконувати фінансові розрахунки з користувачами, вести системний журнал помилок, що полегшує ремонт і т.п.

Це ті функції, які з одного боку не можливо виконати в програмах користувачів, і які, з іншого боку занадто складні, щоб їх можна було реалізувати чисто апаратними засобами. Тому, дещо з іншої точки зору, операційна система - це частина загального програмного забезпечення ЕОМ, яка:

- займає проміжне положення між апаратурою і програмістом-користувачем ЕОМ;

- доповнює апаратуру тими можливостями, які важко або економічно невигідно реалізувати апаратним способом;
- створює умови для ефективної розробки та відлагодження програм;
- знижує вартість використання обчислюваних засобів за рахунок розподілу між користувачами ресурсів ЕОМ та накопиченої інформації.

Таким чином ОС – це основа всієї системи програмного забезпечення сучасної ЕОМ, яка визначає операційне середовище, в якому працюють оператори, програмісти, інженери, адміністратори та власники обчислювальних засобів.

2.2. BIOS - Базова система введення-виведення

Для активізації вузлів комп'ютера завжди використовуються спеціальні програми, які знають, як здійснюється ініціалізація того чи іншого встановленого в комп'ютер пристрою. Конкретна програма може бути дуже короткою, наприклад описуватись парою рядків для активізації пристрою, або дуже великою і складною, такою, що проводить тестування пристрою, налаштування під навколишні елементи і навіть інтерактивне спілкування з користувачем, скажімо, для вибору тієї або іншої функції.

У персональному комп'ютері всі основні програми, призначені для початкового завантаження зібрані в універсальну програму, яка записана в постійному пристрої, який має назву ROM BIOS або, простіше, BIOS, — Basic Input/Output System (базова система введення/виведення). Об'єм сучасної BIOS не меншого 1-2 Мбайт.

Традиційно всі програми, записані в мікросхемі BIOS, можна розділити по виконанню наступних функцій:

- ініціалізація і початкове тестування всіх основних (стандартних) вузлів комп'ютера — розташованих на системній платі, підключених до шини IDE і вставлених в слоти розширення. Для цього використовується програма POST (Power On Self Test), що записана в мікросхемі BIOS. Відзначимо, що "нестандартні" плати розширення, наприклад старі інтерфейси сканерів не тестуються;
- завантаження операційної системи із зовнішнього пристрою — гнучкого диска, вінчестера, компакт-диска або мережевої карти. У найперших персональних комп'ютерах був варіант, коли можна було завантажити інтерпретатор мови Basic, яка знаходилася в додатковій мікросхемі ПЗП;
- обслуговування апаратних переривань, наприклад, від клавіатури і таймера, обробка програмних переривань BIOS, які призначені для управління обміном даними між операційною системою комп'ютера і підключеними до нього периферійними пристроями, виконання базових функцій, наприклад, виведення на екран монітора символів і робота з дисковими пристроями;
- налаштування і конфігурація вузлів системної плати і пристроїв, підключених до неї, що виконується за допомогою програми BIOS Setup.

Практично на всіх ПК встановлено BIOS, яку розробила одна з трьох фірм - AMI, Award або Phoenix. Виробники системних плат не розробляють самі програмне

забезпечення для своїх виробів, а лише іноді допрацьовують стандартні BIOS для конкретної плати. Наприклад, в Росії фірма Ramec встановлює в свої комп'ютери русифіковану версію BIOS.

На сайтах виробників системних плат доступні оновлення BIOS, які користувач може викачати і самостійно встановити. Але слід сказати, що подібну операцію слід робити тільки у тому випадку, коли це життєво необхідно, наприклад, потрібна підтримка нового процесора або інтерфейсу, або треба виправити помилку BIOS (фактично, це означає, що неможлива експлуатація системної плати з тим набором елементів, які є в наявності у користувача). У решті випадків дана операція більш ніж ризикова, оскільки велика вірогідність неправильної прошивки BIOS, що не завжди можливо виправити без допомоги сервісного центру виробника.

Після включення живлення або натиснення кнопки Reset у комп'ютера на адресній шині системної плати встановлюється адреса точки входу в програму BIOS, яка у момент старту знаходиться в самих старших елементах пам'яті, що адресується. Наприклад, в процесорах 8086/8088 після виникнення сигналу RESET припиняються всі поточні процедури, а після закінчення дії цього сигналу управління передається інструкції за адресою 0FFFF0h, в процесорах 386 — за адресою OFFFFFFOh і т.д.

Слід відмітити, що первинна адреса завантаження штучно формується чіпсетом системної плати, який примусово встановлює всі адресні лінії, окрім перших чотирьох, в одиничний стан. Після передачі управління BIOS точка входу стає доступною за стандартною адресою 0FFFF0h, де нею може скористатися будь-яка програма.

Розробники IBM PC сумісних комп'ютерів завжди вимушені озиратися на найперший персональний комп'ютер IBM PC. У ньому стандартне місце BIOS, як і у решти всіх ПК, знаходиться в області за адресами від 0F0000h до 0FFFFFh. Відповідно, доводиться вимушено створювати умови, щоб образ BIOS відображався в двох місцях — в стандартному для IBM PC і в кінці можливої фізичної пам'яті (у ряді BIOS є можливість вказати, що він знаходиться під межею 16 Мбайт, що приводить до неможливості використання більше 16 Мбайт ОЗУ.)

Сама точка входу є такою, що займає п'ять елементів пам'яті (один байт — команда і 4 байти — адреса) команду безумовного переходу (JMP-адреса) на підпрограму початкового запуску комп'ютера.

Окрім стандартної точки старту системи, в BIOS визначено ще декілька елементів пам'яті. З них найбільш важливі знаходяться на самому початку і кінці BIOS — ознаки коректних даних в ПЗП. На початку йдуть два байти з вмістом 55AAh, а в кінці - число 0, яке повинне вийти, якщо скласти значення всіх байтів в ПЗП (по модулю 256). Такими самими ознаками повинні володіти і інші ПЗП, які можуть бути встановлені на платах розширення, наприклад, відеоадаптера чи мережевої плати.

Програми, що знаходяться в BIOS, використовують ряд ресурсів комп'ютера для зберігання даних, отриманих в ході ініціалізації пристроїв ПК, тестування і для роботи службових підпрограм. Найбільш важлива службова зона адрес розміром в 1 Кбайт починається з нульової адреси. У ній знаходяться вектори апаратних і програмних переривань, з якими працюють процесор і програмне забезпечення. Самі

вектори є інструкцією безумовного переходу на підпрограму обробки переривання. Кожен вектор займає 4 байти, відповідно до цього може бути 256 переривань.

2.3. CMOS - Complementary Metal Oxide Semiconductor

Робота таких стандартних пристроїв, як клавіатура, може обслуговуватися програмами BIOS, але такими засобами неможливо забезпечити роботу з усіма можливими пристроями (у зв'язку з їх величезною різноманітністю та наявністю великої кількості різних параметрів). Але для своєї роботи програми BIOS вимагають всю інформацію про поточну конфігурацію системи. З очевидних причин цю інформацію не можна зберігати ні в оперативній пам'яті, ні в постійній.

Спеціально для цих цілей на материнській платі є мікросхема енергонезалежної пам'яті, яка по технології виготовлення називається CMOS. Від оперативної пам'яті вона відрізняється тим, що її вміст не зникає при вимкненні комп'ютера, а від постійної пам'яті вона відрізняється тим, що дані можна заносити туди і змінювати самостійно, у відповідності з тим, яке обладнання входить до складу системи. Мікросхема пам'яті CMOS постійно живиться від невеликої батарейки, що розташована на материнській платі. У цій пам'яті зберігаються дані про гнучкі та жорсткі диски, процесори і т.д. Той факт, що комп'ютер чітко відслідковує дату і час, також пов'язаний з тим, що ця інформація постійно зберігається (і обновлюється) у пам'яті CMOS. Таким чином, програми BIOS зчитують дані про склад комп'ютерної системи з мікросхеми CMOS, після чого вони можуть здійснювати звертання до жорсткого диска та інших пристроїв.

Стандартна статична пам'ять CMOS має 64 регістри, до яких звертаються за адресами введення/виведення від 0 до 3Fh. При виключенні живлення елементу пам'яті, CMOS живиться від батареї напругою 3 В, яка встановлена на системній платі. У CMOS зберігається конфігурація комп'ютера і поточний системний час.

2.4. Переривання, їх роль та процедура звернення в програмах

ЕОМ - цифровий автомат, який функціонує відповідно до програми яка знаходиться в його оперативній пам’яті (ОП). Програміст в своїй програмі приписує ЕОМ послідовність змін своїх станів у відповідності з деяким алгоритмом досягнення мети. В режимі мультипрограмування є необхідними засоби оперативного втручання в роботу ЕОМ. Найбільш яскраво ця необхідність проявляється в керуючих обчислювальних системах, де до ЕОМ безперервно надходить інформація від об’єктів. Частина цієї інформації є періодичною з відомим наперед періодом і цю інформацію ЕОМ може сприйняти шляхом періодичного опитування тих чи інших давачів. Інша частина інформації від об’єктів надходить не періодично, а деякі з цих інформаційних посилок вимагають негайної реакції від ЕОМ, оскільки затримка в обробці цих заходів у прийнятті відповідних рішень може мати катастрофічні наслідки.

Оперативне втручання зовні необхідне також і для нормальної роботи ЕОМ, для сприймання повідомлень про завершення обмінів з зовнішніми пристроями, для

забезпечення можливості людині-оператору втручатись в хід обчислювального процесу. Ці можливості забезпечує система переривань ЕОМ.

Переривання - це припинення послідовного виконання команд активної програми (тої, чиї команди виконує в даний момент процесор) і перехід до спеціальної підпрограми. Цей перехід є тимчасовим - по закінченню підпрограми виконання перерваної програми відновлюється з того місця, де вона була призупинена.

Переривання діляться на 5 типів:

- зовнішні переривання (клавіатура, таймер, лінії зв'язку);
- виклик супервізора з програми користувача для виконання якоїсь з його функцій (поява в програмі команди INT-SVC);
- переривання через особливі ситуації, що виникають в ході виконання програми (порушення захисту пам'яті, помилка в коді операції, арифметичне переповнення та ін.);
- переривання через збій ЕОМ (від схем оперативного контролю ЕОМ).
- переривання від пристроїв введення-виведення.

Розглянемо послідовність дій, викликаних сигналом переривання:

1. Перехід до підпрограми. Виконується апаратно. Він може реалізовуватись шляхом занесення в лічильник команд адреси спеціальної підпрограми (відповідно до причини переривань) або шляхом позачергового запису в регістр дешифратора команд спеціальної команди переривання, першими діями якої є організація переходу до цієї спеціальної програми.

2. Заборона інших переривань. Під час обробки переривання встановлюється апаратно або програмно декілька масок або ключів, які закривають деякі види переривань, що захищає виконувану підпрограму від переривань з нижчим пріоритетом.

3. Запам'ятовування інформації, необхідної для відновлення виконання перерваної програми. Оскільки метою переривання є виконання спеціальної підпрограми з наступним поверненням до перерваної програми, необхідно запам'ятати всю інформацію, зв'язану з виконанням перерваної програми для наступного відновлення її роботи. Як правило, ця інформація містить такі параметри:

- лічильник команд, який містить адресу, на якій програма була перервана і до якої повинно відбутись повернення;
- внутрішні регістри процесора: суматор, РЗП та ін;
- регістри захисту пам'яті та переадресації;
- деякі робочі комірки пам'яті;
- індикатори типу індикаторів переповнення та переносу;
- умови виконання програми, які зберігаються, як правило, в слові стану програми.

Запам'ятовування перерахованих комірок пам'яті регістрів та індикаторів виконується апаратно або програмно, найчастіше використовуються обидва способи.

4. Виконання підпрограми переривань. Ця частина містить фазу діагностики для виявлення точної причини переривання і наступне виконання необхідних у

цьому випадку дій. Наприклад, переривання мало місце з тої причини, що прийшов сигнал про закінчення деякого обміну з зовнішнім пристроєм пам’яті. ЦП переводить програму, яка зажадала цього обміну і тому була заблокована, в стан готовності до подальшої роботи.

5. Відновлення інформації і поновлення виконання перерваної програми. Ця фаза є протилежною до фаз запам’ятовування інформації, заборони переривань та переходу до підпрограми. Відновлюються параметри перерваної програми. Заборони з переривань знімаються і виконується перехід до того місця, де програма була перервана.

ТЕМА 3. МОДЕЛІ РОЗПОВСЮДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Безкоштовне програмне забезпечення

Під моделлю безкоштовного ПЗ (Freeware) розуміється відсутність будь-якої оплати за користування ПЗ. Дуже часто за таким принципом розповсюджуються невеликі утиліти, які, на думку авторів, можуть виявитися корисними широкому кругу користувачів, але не матимуть попиту, якщо призначити плату за їх використання. У більшості випадків безкоштовне ПЗ розробляється одним програмістом, ентузіастом своєї справи несхильним до зайвої комерціалізації сучасних інформаційних технологій. Часто декілька добровольців організовують команду, що розробляє і підтримує достатньо складну програмну систему.

Багато безкоштовних програм розповсюджуються з відкритими кодами. Однак програми з відкритими кодами і безкоштовні програми - це зовсім не одне і те ж. Комерційні продукти теж можуть поставлятися з відкритими кодами. Досить часто зустрічається ситуація, коли програма або бібліотека безкоштовна для особистого використання, але вимагає ліцензії (іноді досить дорогої) для комерційного застосування.

Не рідкісні випадки, коли безкоштовне програмне забезпечення розробляється великими комерційними компаніями для зміцнення своєї долі на ринку ПЗ. Так, наприклад, документи у форматі PDF навряд чи мали б сьогодні таку популярність, якби ніколи не існувала безкоштовна програма для їх перегляду - Adobe Acrobat Reader, доповнююча лінійку платних продуктів для створення PDF-документів (Acrobat Exchange, Distiller, Business Tools, Approval і т.д.). Аналогічно, для документів, створених в комерційній програмі Microsoft Word, існувала безкоштовна програма перегляду Microsoft Word Viewer, розроблена корпорацією Microsoft.

Іноді автор безкоштовного проекту продає кому-небудь всі права на свою розробку, а новий власник починає поширювати ту ж саму програму за гроші, іноді наймаючи колишнього автора для продовження розробки.

3.2. Умовно безкоштовне програмне забезпечення

Наявність можливості оцінити програму до покупки ("try before you buy") є однією з головних рис умовно безкоштовних продуктів (share). У більшості випадків автори умовно безкоштовного ПЗ надають користувачам незареєстровані версії своїх розробок, які можна використовувати в незміненій формі. Умовно безкоштовне програмне забезпечення розробляється з метою одержання якогось прибутку чи вигоди. Однак, перед одержанням прибутку від свого ПЗ, потенційному користувачеві обов'язково надається можливість апробувати програмний продукт у дії протягом деякого періоду часу з невеликими функціональними обмеженнями. Після закінчення тестового періоду потенційний покупець повинен ухвалити рішення про придбання програми. Якщо вирішено відмовитися від покупки, то треба припинити використовувати програму і видалити її з комп'ютера. Інакше, необхідно сплатити

ліцензію на програму чи надати авторові чи виконати іншу запропоновану автором умову. Після цього автор надає можливість отримання повно функціональної версії ПЗ.

Зазвичай умовно безкоштовні програми доставляються через Інтернет і мають невеликий розмір (декілька мегабайт). Також дуже часто для перетворення обмеженої версії на повно функціональну не потрібні ніякі додаткові файли - досить ввести правильний реєстраційний код, отриманий від автора.

Сьогодні умовно безкоштовні продукти дуже популярні. Багато відомих розробників ПЗ беруть на озброєння концепцію "Try before you buy", щоб зацікавити покупців. По суті, обмежені ознайомлювальні версії комерційних продуктів є всього лише одній з модифікацій ідеї Shareware. Навіть корпорація Microsoft безкоштовно поширює 120-денні версії Windows 2003 Server і Visual Studio .NET. Правда, невелика відмінність полягає в тому, що для перетворення 120-денної версії на повноцінну доведеться отримати диск з новою версією і виконати процедуру оновлення, а для "класичних" умовно безкоштовних програм перехід до повної версії відбувається відразу ж після введення правильного реєстраційного коду.

Іноді автори умовно безкоштовного ПЗ вибирається один з наступних методів розповсюдження:

- Cardware - кожен користувач програми, який хоче її зареєструватися, повинен надіслати авторові програми поштову листівку з виглядом місцевості, де він проживає;
- Mailwam - сучасніший варіант Cardware, у якому авторові програми надсилається електронний лист. Як правило, у відповідь автор присилає реєстраційний код, що дає можливість працювати з програмою;
- Donationware - коли автор не вимагає ніякої оплати, але пропонує всім, кому сподобалася програма, пожертвувати довільну суму, щоб підтримати розробку;
- Gifrware - майже те ж саме, що і Donationware, але автор готовий приймати не тільки грошові пожертвування, але і інші подарунки;
- Beerware - подяка за програму приймається у вигляді пива;
- Vegeware - автор збирає з користувачів плату за програму у формі рецептів вегетаріанських блюд.

3.3. Комерційне програмне забезпечення

Комерційне програмне забезпечення створюється з метою одержання прибутку у вигляді матеріальної винагороди. Комерційне ПЗ більше всього схоже на звичайний товар, який люди звикли купувати в магазинах. Перш за все, для програмного забезпечення, що поширюється як комерційне, застосовується принцип "гроші - вперед", тобто користувач отримує програму тільки після повного внесення оплати за неї. Дуже багато програм, поширюється у такий спосіб. У більшості випадків при покупці комерційного ПЗ користувач отримує коробку, в якій містяться носії інформації (наприклад DVD або компакт-диски), документація, реєстраційна картка та інше, на розсуд продавця.

Зрозуміло, що автор (або правовласник) зацікавлений в тому, щоб зібрати плату з кожного користувача програми. Для досягнення цього необхідно застосовувати технологічні методи, що обмежують розповсюдження нелегальних (не ліцензованих) копій програми. До популярних технологічних методів відносяться різні апаратні захисти, системи реєстрації і активації, перевірка ліцензії через Інтернет при кожному запуску програми і т.д.

Проте чисте комерційне програмне забезпечення володіє однією дуже важливою особливістю. Кінцевий споживач зможе отримати уявлення про те, що він купує, тільки після здійснення покупки, і, отже, достатньо висока вірогідність того, що він буде розчарований і захоче позбавитися від програми та повернути назад заплачені гроші. Для того, щоб не відлякати покупців, продавці часто вимушені обіцяти повернення грошей, наприклад, після двох тижнів з моменту покупки, якщо програма не сподобається.

Останніми роками XX століття, разом із бурхливим розвитком Інтернет технологій набула поширення модель розповсюдження програмного забезпечення, що демонструє рекламу (Adware), яку також можна віднести до комерційної. Основна ідея полягає в тому, що розробник отримує плату за використання програми не від кінцевого споживача, якому програма дісталась безкоштовно, а від рекламодавців. При цьому користувач комерційної програми вимушений дивитися картинки, що підкачуються з Інтернету. Зрозуміло, що цей підхід актуальний тільки для ПЗ, робота якого прямо пов'язана з доступом в Інтернет. Проте з часом ефективність реклами такого роду значно знизилася, і знайти охочих платити за неї справжніми грошима стало набагато важче. Однак все ще продовжують існувати спонсоровані програмні продукти (як правило, інформаційній спрямованості), розробка яких ведеться на гроші рекламодавців в обмін на розміщення їх інформації в програмі.

Розробники комерційних програм в рекламних цілях випускають обмежені ознайомлювальні версії своїх продуктів. Такі версії зазвичай не дозволяють плідно працювати, але створюють правдиве враження про функціональність програми. Можна виділити декілька основних типів обмежених комерційних програмних продуктів:

- Demoware - це коли в програмі присутні функціональні обмеження. Наприклад, можна обробляти файли не більші заданого розміру, не можна виконувати збереження і т.д. Такі програми іноді називають Crippleware - "урізане" програмне забезпечення;
- Trialware - наявність обмежень за часом використання ПЗ. Обмеження можуть виражатися у вигляді тривалості періоду часу, впродовж якого можна користуватися програмою (наприклад 30 днів з моменту інсталяції) або у вигляді фіксованої дати закінчення тестового періоду. Може обмежуватися число запусків програми або число процесів обробки;
- Nagware - користувачу регулярно нагадується про те, що дана версія програми не є повноцінною комерційною версією. Таке нагадування може виглядати як діалогове вікно, що з'являється при запуску програми і з деякою періодичністю під час роботи ПЗ, додаткові написи, що виводяться на принтер або екран, і т.д.

ТЕМА 4. МЕТОДИ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Мотиви захисту програмного забезпечення

Під мотивами захисту ПЗ розуміється «зміст захисту ПЗ» або «від чого захищають ПЗ».

Комерційні програми зазвичай захищають від несанкціонованого тиражування. Наявність доступу тільки до носія інформації з дистрибутивом (набором інсталяційних файлів) програмного продукту не повинна давати можливості встановити працездатну копію програми. Тобто даних дистрибутива, який можна скопіювати або непомітно взяти на декілька днів, не повинно вистачати для створення працездатної копії програми. Подібні обмеження можуть бути реалізовані різними способами. Наприклад, дуже багато комерційних програм при інсталяції вимагають ввести серійний номер, надрукований на коробці або вказаний в одному з документів, що додаються до програмного продукту (у Microsoft - в сертифікаті автентичності).

Також часто виникає потреба обмежити число користувачів, що одночасно працюють з програмою. Тобто людина, яка придбала ліцензію на одне робоче місце, не повинна мати можливості створити 2 робочих місця, що функціонують одночасно. Це досягається за рахунок використання апаратних ключів, менеджерів ліцензій і процедури активації.

Для деяких програмних продуктів (зокрема ігор) часто використовується прив'язка до носія інформації, наприклад компакт-диску. Тобто для запуску гри потрібна наявність в приводі оригінального компакт-диска, який захищений від копіювання стандартними засобами.

Для комерційних Trialwre програмних продуктів, обмежених за часом або числом запусків, необхідно правильно реалізувати зберігання лічильників, щоб зловмисник не зміг примусити працювати програму, просто перевівши годинник або видаливши файл, в який записується кількість запусків програми або число оброблених файлів.

Умовно безкоштовні продукти, на відміну від обмежених по функціональності версій комерційних програм, після введення реєстраційного коду повинні надавати доступ до всіх функцій, передбачених в повній версії програми. Тобто в безкоштовно поширюваній версії програми повинні бути реалізовані всі функції повної версії. Отже для цієї моделі розповсюдження, бажано так організувати захист, щоб зловмисник не зміг дістатися до функцій, властивих тільки повній версії, поки в його розпорядженні не буде правильного реєстраційного коду чи іншої необхідної інформації.

Процедури перевірки правильності серійних номерів, а також реєстрації віконних кодів і кодів активації повинні будуватися так, щоб зловмисник не міг самостійно генерувати правильні коди і, в той же час, довжина кодового рядка не була дуже великою.

Також може виникнути потреба захищати будь-які виконувані файли від внесення змін, дизасемблювання, дослідження під відлагоджувачем та інше.

4.2. Реєстраційні коди

Процедура реєстрації ПЗ полягає у введені певної інформації про споживача ПЗ (заповнення своєрідної реєстраційної картки) та надсилання її виробникові через електронну пошту, Інтернет... Після цього споживачеві надсилається реєстраційний код, після введення якого він стає зареєстрованим користувачем і отримує передбачувані привілеї (технічну підтримку, гарантійне обслуговування та інше). У свою чергу виробник ПЗ поповнює статистичну інформацію про своїх клієнтів. Оскільки ім'я користувача не є унікальним, кожен екземпляр продукції, що продається, доцільно пов'язувати з деяким значенням, що не повторюється - серійним номером. Цей номер вказується користувачем при заповненні реєстраційної картки і надалі використовується при спілкуванні з виробником. А в додатку до програмних продуктів серійний номер цілком може виконувати і допоміжну функцію - обмежувати нелегальне копіювання. Якщо програма при встановлені вимагає ввести правильний серійний номер, вкравши (скопіювавши) носій з дистрибутивом програми, який однаковий у всіх користувачів, отримати робочу копію програми не вдасться. А розповсюдження серійного номера дозволяє знайти і покарати асоційованого з цим номером користувача.

В деяких випадках після встановлення програми (неважливо, з введенням серійного номера або без) для одержання доступу до всіх функцій програми користувачеві необхідно виконати ще одну процедуру — реєстрацію або, як це тепер називає Microsoft, активацію. Така поведінка характерна для більшості Shareware-продуктів, а також для програм, розробники яких вважають, що користувач не має права працювати, поки не повідомить про себе всі необхідні відомості, навіть якщо він вже придбав ліцензію.

При введені одержаного реєстраційного коду спрацьовує механізм захисту ПЗ, який перевіряє вірність введеного коду. При цьому можливість обчислювати вірні коди повинна завжди залишатися тільки в руках розробника. Для того, щоб супротивник, виправивши декілька байт, не зміг заставити ПЗ працювати так, як ніби він був коректно зареєстрований чи активований, необхідно частину програмного коду або даних, доступ до яких дозволений тільки легальним користувачам зашифрувати стійким алгоритмом, а ключ шифрування обчислювати, використовуючи реєстраційний код. Тоді без знання реєстраційного коду отримати повноцінну версію програми не вдасться. Подібну функціональність забезпечують, програми ASProtect (ASPack Software) і EXECryptor (SoftCompIete Development).

Визначимо декілька критеріїв, по яких можна порівнювати властивості різних методів генерації і перевірки кодів:

- можливість пов'язати код з ім'ям користувача або характеристиками комп'ютера;
- неможливість обчислити будь-який правильний код, маючи в розпорядженні тільки алгоритм його перевірки;
- неможливість обчислити код для певного користувача, знаючи алгоритм перевірки і правильний код іншого користувача;
- неможливість розшифрування програми (отримання ключа шифрування) за наявності заблокованого (занесеного в чорний список) реєстраційного коду;

- довжина ключового рядка (зручність користувача).

Всі методи перевірки правильності кодів можна, умовно, розділити на три категорії:

- алгоритмічні, що базуються на принципі "чорного ящика";
- алгоритмічні, що базуються на математично складному завданні;
- табличні.

Будь-які алгоритмічні методи дозволяють пов'язати код з ім'ям користувача або інформацією про його комп'ютер, тим самим ускладнивши отримання декількох копій програми, що виглядають як легальні. При використанні "чорного ящика" розробник має намір заплутати алгоритм перевірки, щоб його було важче зрозуміти і обернути. Такий підхід, напевно, використовується частіше за всіх інших. Якщо процедура перевірки написана без грубих помилок, то отримати з неї правильний код неможливо, але, знаючи один правильний код і обернувши процедуру перевірки, зламувач може обчислити будь-які нові коди.

Алгоритмічні методи перевірки реєстраційного коду, що базуються на складному математичному завданні, не потребують приховування деталей реалізації. Їх особливість в тому, що для генерації коду і перевірки його правильності використовуються два різних алгоритми і отримання алгоритму генерації з алгоритму перевірки є математичним завданням, що не має на теперішній час ефективного рішення. Найчастіше для цих цілей використовуються криптографічні алгоритми з відкритим ключем. Проте у асиметричної криптографії є одна особливість: розмір блоку, над яким проводяться операції, досить великій. Так, для RSA-1024 розмір блоку (а значить, і мінімальний розмір реєстраційного коду) складає 128 байт. А щоб двійкові дані можна було ввести з клавіатури, їх кодують, наприклад, алгоритмом MIME64, який збільшує розмір блоку на третину. Тобто довжина кодового рядка складе як мінімум 172 символи. Очевидно, що ввести без помилок безглузду послідовність такої довжини, що складається з букв, цифр і розділових знаків, майже неможливо. Це робить дану схему неприйнятною для реєстрації, наприклад, по телефону або факсу. Сьогодні проводяться спроби створити стійку систему реєстрації, що використовує короткі коди і базується на складному математичному завданні. Так, компанія SoftComplete Development в своєму продукті HardKey System версії 2.0 (квітень 2002) використовувала алгоритм, для зламу якого треба було багато разів обчислити дискретний логарифм, що є складним завданням. У будь-якому випадку, при правильній реалізації алгоритмічні методи, що базуються на математично складному завданні, не дозволяють зламнику, що знає один правильний реєстраційний код, генерувати нові коди. Однак єдиний спосіб заблокувати вкрадений код полягає в тому, щоб занести цей код в так званий "чорний список". І очевидно, що обхід блокування вимагає не рішення математичної задачі, а виправлення логічної умови. Тобто при бажанні зламник може отримати повністю працездатну версію розшифрованої програми, маючи тільки заблокований код.

У табличних методах генерується задана кількість реєстраційних кодів (по числу можливих користувачів), і в програмі зберігаються таблиці, побудовані на основі цих кодів. Зрозуміло, коди не можуть залежати від імені користувача або характеристик системи, оскільки генеруються до того, як з'являються перші

зареєстровані користувачі. Самий простій спосіб - зберігати в програмі результат обчислення криптографічної хеш функції від кожного коду. При цьому легко перевірити правильність ключа, обчисливши його хеш, однак, маючи значення хеш функції, обчислити ключ практично неможливо. Якщо частину реєстраційного коду зробити статичною (однаковою для всіх кодів), то її можна використовувати як ключ для шифрування програми. Але при такому підході заблокований код легко може бути використаний для розшифрування, якщо його хеш додати у таблицю, що зберігається всередині програми, або взагалі відключити перевірку правильності хеша. Тому правильніше для кожної нової публічної версії продукту випадковим чином генерувати ключ шифрування програми, при цьому кожен запис таблиці повинен одержуватись шляхом зашифрування ключа програми на ключі, отриманому з реєстраційного коду. І, зрозуміло, кожен запис повинен містити деяку контрольну інформацію, що дозволяє оцінити правильність розшифрування.

У кожного з описаних методів перевірки правильності кодів є свої переваги і недоліки.

Так "чорний ящик" порівняно простий в реалізації і дозволяє використовувати короткі коди, прив'язані до імені користувача. Але майже завжди при використанні цього підходу можливе створення генератора ключів.

Методи, що базуються на стійкій криптографії, досить складні для самостійної реалізації і дуже часто вимагають довгого кодового рядка. Крім того, багато алгоритмів на сьогодні запатентовані.

Табличні методи дають можливість повністю блокувати зламані реєстраційні коди, однак не дозволяють прив'язувати код до імені користувача. Крім того, якщо число користувачів дуже велике, таблиці можуть займати великий об'єм.

4.3. Апаратні ключі

Одним із сучасних методів захисту ПЗ є використання апаратних ключів, які поставляються разом з ліцензійним ПЗ і виконуються функцію його захисту від несанкціонованого використання. Апаратні ключі виконуються у більшості випадків у вигляді флеш пристрою та підключаються до одного з портів ПК. При запуску ПЗ із методом захисту на основі апаратних ключів відбувається звернення ПЗ до порту з ключем та виконується ряд запитів до ключа, який функціонує на базі певного алгоритму. Сьогодні відомо багато алгоритмів на основі яких працюють апаратні ключі.

Наведемо найбільш розповсюдженні алгоритми захисту з використанням апаратних ключів:

- Ключі з пам'яттю – один з найпростіших типів ключів. Ключі з пам'яттю мають певне число комірок пам’яті, з яких дозволено зчитування. У деякі з цих комірок також може проводитися запис. Зазвичай в комірках пам’яті недоступних для запису, зберігається унікальний ідентифікатор ключа. Ключі з пам'яттю не здатні протистояти емуляції. Достатньо один раз прочитати всю пам'ять і зберегти її в емуляторі. Після цього правильно емулювати відповіді на всі запити до ключа не складе великих труднощів.

Таким чином, апаратні ключі з пам'яттю в заданих умовах не здатні дати ніяких переваг в порівнянні з чисто програмними системами.

- Ключі з невідомим алгоритмом. Багато сучасних апаратних ключів містять секретну функцію перетворення даних, на якій і ґрунтується секретність ключа. Іноді програмістові надається можливість вибрати константи, перетворення, що є параметрами, але сам алгоритм залишається невідомим. Перевірка наявності ключа повинна виконуватися таким самим чином. При розробці захисту програміст робить декілька запитів до алгоритму і запам'ятовує отримані відповіді. Ці відповіді в якійсь формі кодуються в програмі. Під час виконання програма повторює ті ж запити і порівнює отримані відповіді із збереженими значеннями. Якщо виявляється неспівпадання, значить, програма отримує відповідь не від оригінального ключа. Ця схема має один істотний недолік. Оскільки захищена програма має кінцевий розмір, тому і кількість правильних відповідей, які вона може зберігати є обмеженою. А це означає, що існує можливість побудови табличного емулятора, який знатиме правильні відповіді на всі запити, результат яких може перевірити програма.
- Ключі з таймером. Деякі виробники апаратних ключів пропонують моделі, що мають вбудований таймер. Але для того, щоб таймер міг працювати в той час, коли ключ не підключений до комп'ютера, необхідне вбудоване джерело живлення. Середній час роботи батареї, що живить таймер, складає 4 роки, і після її розрядки ключ перестане правильно функціонувати. Можливо, саме із-за порівняно короткого часу життя ключі з таймером застосовуються досить рідко. Прикладом таких ключів є ключі HASP Time, що надають можливість дізнаватися поточний час, встановлений на вбудованому в ключ годиннику. І захищена програма може використовувати ключ для того, щоб відстежити закінчення тестового періоду. Але очевидно, що емулятор дозволяє повертати будь-які покази таймера, тобто апаратна частина ніяк не підвищує стійкість захисту.
- Ключі з відомим алгоритмом. У деяких ключах програмістові, що реалізовує захист, надається можливість вибрати з безлічі можливих перетворень даних, що реалізовуються ключем, одне конкретне перетворення. При цьому програміст знає всі деталі вибраного перетворення і може повторити зворотне перетворення в чисто програмній системі. Наприклад, апаратний ключ реалізує симетричний алгоритм шифрування, а програміст має можливість вибирати використовуваний ключ шифрування. Зрозуміло, ні в кого не повинно бути можливості прочитати значення ключа-шифрування з апаратного ключа. У такій схемі програма може передавати дані на вхід апаратного ключа і отримувати у відповідь результат шифрування на вибраному ключі. Але тут виникає дилема. Якщо в програмі відсутній ключ шифрування, то повернені дані можна перевіряти тільки табличним способом, а значить, в обмеженому об'ємі. Фактично маємо апаратний ключ з невідомим програмі алгоритмом. Якщо ж ключ шифрування відомий програмі, то можна перевірити правильність обробки будь-якого об'єму даних, але при цьому існує

можливість витягання ключа шифрування і побудови емулятора. А якщо така можливість існує, супротивник обов'язково спробує нею скористатися.

- Ключі з програмованим алгоритмом. Дуже цікавим рішенням з погляду стійкості захисту є апаратні ключі, в яких може бути реалізований довільний алгоритм. Складність алгоритму обмежується тільки об'ємом пам'яті і системою команд ключа. В цьому випадку для захисту програми важлива частина обчислень переноситься в ключ, і у супротивника не буде можливості запротоколювати правильні відповіді на всі запити або відновити алгоритм по функції перевірки. Адже перевірка, як така, може взагалі не виконуватися — результати, повернені ключем, є проміжними величинами в обчисленні якоїсь складної функції, а значення, що подаються на вхід, залежать не від програми, а від оброблюваних даних.

4.4. Навісні захисти (протектори)

Протектором називають програмних код, який вмонтовується у виконуваний файл та відповідає за правильне завантаження всіх змінних цього файлу в пам’ять. Практично для всіх форматів виконуваних файлів були розроблені алгоритми, що дозволяють додавати новий код так, щоб він виконувався до основної програми, не порушуючи при цьому її функціональності. Швидше за все, основні дослідження в цій області були виконані авторами вірусів, оскільки додавання тіла вірусу до програми є одним з основних методів зараження. Код, дані і ресурси зазвичай захищаються за допомогою шифрування. Алгоритм, що використовується не обов'язково повинен бути криптографічно стійким, оскільки ключ шифрування все одно неможливо зберегти в цілковитій таємниці. Дуже часто до шифрування застосовується стиснення даних, що дозволяє компенсувати збільшення розміру виконуваного файлу, що відбувається внаслідок додавання коду протектора. А іноді результуючий захищений файл навіть зменшується в розмірі в порівнянні з початковим файлом.

При запуску захищеної програми управління відразу отримує код протектора, який виконує передбачені перевірки і розшифровує у пам'яті всі необхідні області, а також проводить налаштування таблиці адрес функцій, що імпортуються. Після успішного завершення процедури налаштування протектор передає управління на оригінальну точку входу (Original Entry Point, OEP) і починається виконання основної програми.

Перевірки, що виконуються протектором до початку роботи програми, можуть бути різного роду. Це можуть бути перевірки, що стосуються наявності ліцензії (щоб без ліцензії програма просто не запускалася) або порівняння поточної дати із значенням, після якого програма повинна перестати працювати, а також спроби визначити наявність запущеного відлагоджувача.

Протектори розроблялись з метою забезпечення захисту вмісту виконуваного файлу від дослідження і модифікації. Але часто виходить, що захищена програма по деяких характеристиках виявляється для користувача гірше, ніж та ж програма без захисту.

4.5. Захист від несанкціонованого копіювання

При захисті ПЗ від несанкціонованого копіювання використовуються методи, що дозволяю реалізовувати в ПЗ функції прив'язки процесу виконання коду програми до ПК на яких дана програма виконується. Інстальована програма для захисту від копіювання при кожному запуску повинна виконувати наступні дії:

- аналізувати програмно-апаратне середовище ПК, на якому вона запущена та формувати на основі цього аналізу характеристики свого середовища виконання;
- провіряти ідентичність середовища виконання шляхом порівняння поточних характеристик з еталонними, що зберігаються на жорсткому диску ПК;
- блокувати свій запуск у випадку неспівпадання поточних характеристик з еталонними.

До основних методів захисту від копіювання можна віднести:

- криптографічний метод. Для цього методу інсталятор програми повинен виконувати: 1. аналіз програмно-апаратного середовище ПК, на якому він інстальована програма та формувати на основі цього аналізу еталонні характеристики середовища виконання програми; 2. проводити запис криптографічних перетворень еталонних характеристик програмно-апаратного середовище ПК на жорсткий диск.
- метод прив'язки до ідентифікатора. Зміст даного методу заключається в тому, що на жорсткому диску при інсталяції захищеної від копіювання програми формується унікальний ідентифікатор, який звіряється при кожному запуску програми. При відсутності чи неспівпадінні цього ідентифікатора програма блокує своє виконання.
- метод маніпуляції з кодом програми. Є два способи реалізації даного методу: 1. включення в тіло програми пустих модулів, на які імітується передача керування; 2. зміна початку захищеної програми таким чином, щоб стандартний дизасемблер не зміг її правильно дизасемблювати.

4.6. Стеганографічний захист даних

Стеганографія (від грецького "тайнопис") має багатовікову історію. Мета стеганографії – приховати сам факт існування повідомлення. Такі приховані повідомлення можуть включатися в різноманітні зовнішні "невинні" дані і передаватися разом з ними без будь-якої підозри збоку.

Базові принципи комп'ютерної стеганографії такі:

1. Захист має ґрунтуватися на припущенні, що зловмисник має повне уявлення про стеганографічну систему та деталі її реалізації. Єдиною інформацією, яка залишається невідомою потенційному зловмиснику є ключ, за допомогою якого лише його власник може встановити факт присутності та зміст прихованого повідомлення.
2. Якщо зловмисник якимось чином дізнається про факт існування прихованого повідомлення, це не повинно дозволити йому довести цей факт третій особі і

тим більше виявити подібні повідомлення в інших даних доти, поки ключ зберігається в таємниці.

3. Потенційний зловмисник повинен бути позбавлений будь-яких технічних та інших переваг у розпізнаванні або розкритті змісту таємного повідомлення.

Розглянемо кілька методів приховування повідомлень у цифрових сигнатурах та інших добре визначених, але малоінформативних компонентах цифрового зв'язку, які сьогодні є вже досить поширеними.

Приховування даних у цифрових комунікаціях. Левова частка комп'ютерної інформації "шумить" (наявність помилок у даних, завад та інших випадкових сигналів у каналах зв'язку). Шум є практично в будь-якому масиві результатів вимірювань, графічному образі, звуковому файлі тощо. Практичні алгоритми стеганографії якраз і засновані на ідеї заміни за певними законами шумових компонент інформації початковим текстом. Називатимемо таку інформацію, що "шумить" і призначена для приховування таємних повідомлень, контейнером, а біти, що "шумлять" – бітами контейнера. Біти контейнера, замінені бітами приховуваного повідомлення, дістали назву прихованих бітів. Дані контейнера мають бути досить "шумними", щоб невеликі зміни в їх безладді не могли стати помітними. Такий метод відомий як сурогатна стеганографія.

Припускається, що кодування прихованого повідомлення має відтворювати характеристики шуму контейнера, що є важко досяжною, але реальною метою. Одна з можливостей полягає в генерації великої кількості альтернативних контейнерів, для того щоб вибрати з них найбільш придатний для зберігання таємного коду. Такий підхід називається селектуючою стеганографією. Єдина пов'язана з ним проблема полягає в тому, що навіть оптимально організований, він дає змогу приховати незначну кількість даних при дуже великій обчислювальній складності.

Ще один варіант – моделювання характеристик шуму контейнера. Наслідувана функція має бути побудована так, щоб не тільки кодувати приховувані повідомлення, а й дотримуватися моделі початкового шуму. У граничному випадку ціле повідомлення може конструюватися згідно з моделлю шуму. Подібний підхід можна назвати конструюючою стеганографією. Така стратегія має ряд недоліків: її проблематично з'єднати з сильним алгоритмом шифрування, а моделювання шуму – заняття не з легких. Більше того, реальні зразки, створенні на основі цієї моделі, іноді можуть навіть сприяти виявленню таємного повідомлення замість того, щоб збільшувати його безпеку.

4.7. Криптографічний захист програмного забезпечення

Сучасна комп'ютерна система надає своїм користувачам функції захисту інформації, яка у ній зберігається або обробляється. До переліку цих функцій входять: аутентифікація користувача, розмежування доступу до інформації, забезпечення цілісності, конфіденційності інформації, її захист від модифікації або знищення, електронний цифровий підпис та інше. Частина перелічених функцій традиційно реалізується за допомогою криптографічних алгоритмів перетворення інформації у вигляді програмних модулів для універсальних процесорів.

Як за рубежем, так і в Україні запроваджені відповідні стандарти на криптографічні алгоритми захисту інформації. До переліку алгоритмів входять: алгоритми цифрового підпису, алгоритми обчислення хеш-функцій, алгоритми симетричного шифрування та інші. Більшість захищеного програмного забезпечення підтримує, як правило, виконання симетричних блокових та асиметричних алгоритмів шифрування.

До переліку симетричних блокових шифрів входять: ГОСТ28147-89, DES, IDEA та інші. Ці алгоритми використовують невеликий набір базових операцій: додавання (віднімання) за заданим модулем частини блоку із ключем, множення за модулем, зсув на задану кількість біт, перестановка біт, заміна частин блоку у відповідності із таблицею чи функцією. Алгоритм перетворення даних складається із декількох раундів, кількість яких строго визначена для конкретного алгоритму, у кожному з яких виконуються перелічені операції. Також у перетворенні даних бере участь ключ чи його елементи, обчислені за відповідним алгоритмом. Розмір вхідного блоку складає, як правило, 64 біт, розмір ключа змінюється від 56 до 256 біт.

Асиметричні алгоритми шифрування RSA, DSA, ГОСТ Р34.10-94 використовують такі базові операції: множення, додавання, піднесення до степеня за модулем. Як особливість цього класу алгоритмів необхідно зазначити виконання перелічених операцій над числами із розрядною сіткою порядку 160 – 2048 біт.

Криптографічні хеш-функції дозволяють перетворити вхідну послідовність будь-якого розміру в вихідне значення фіксованої довжини. Хеш-функції використовуються в криптографічних протоколах аутентифікації, алгоритмах електронно-цифрового підпису, генераторах псевдо-випадкових послідовностей. Криптографічні хеш-функції обов'язково є однонаправлені, для того щоб противних не зміг розкрити початкове повідомлення. Більше того, не повинно бути ефективного способу знаходження повідомлення, обчислення хеш-функції від якого зможе бути одержане необхідне значення хеша.

Математичні схеми, що використовуються в алгоритмах та реалізують електронний цифровий підпис, ґрунтуються на однонаправлених функціях. На практиці, як правило, в схемах ЕЦП замість документа x розглядають його хеш-функцію $h(x)$, яка володіє рядом спеціальних властивостей, найважливіша з яких – відсутність "колізій" (тобто практична неможливість створення двох різних документів з однаковим значенням хеш-функції). Найбільш відомі математичні схеми ЕЦП такі: RSA (R.L.Rivest, A.Shamir,L.Adleman), OSS (H.Ong, C.P.Schnorr, A.Shamir), Ель-Гамаля (T.ElGamal), Рабіна (M.Rabin), Окамото-Сараісі (T.Okamoto, A.Shiraishi), Мацумото-Імаі (T.Matsumoto,H.Imai).

ТЕМА 5. ЗАСОБИ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1. Засоби захисту вмісту файлів та папок

Захисту вмісту файлів та папок дозволяє насамперед зберегти цілісність, конфіденційність та не ушкодження інформації, що у них зберігається. Захист вмісту файлів та папок може здійснюватися засобами операційної системи, а також спеціалізованими програмними продуктами. У випадку захисту на рівні ОС, доцільно згадати файлову систему NTFS, яка дозволяє розмежовувати права доступу до папки чи файла, а також встановлювати дозволи на їх читання, модифікацію чи видалення.

Наведемо перелік та коротку характеристику найбільш розповсюджених спеціалізованих програмних продуктів для захисту вмісту файлів та папок.

Folder Guard - програма для захисту файлів та папок. Дозволяє "заховати" папки і файли, встановити паролі на папки та інше. Може працювати в мережі.

ORiEN - програма для захисту "exe" файлів за допомогою пароля або ключового файлу. Може обмежувати запуск програми певною операційною системою або диском. Може захистити від відлагоджувачів і вірусів.

Hide Folders XP дозволяє приховати найбільш важливі папки і захистити дані від випадкового стирання. Підтримує файлові системи FAT, FAT32 і NTFS.

Password Door дозволяє заборонити паролем запуск будь-якої програми встановленою на комп'ютері.

Advanced Hide Folders - обмежує доступ не тільки на запуск програм, але і на відкриття файлів і папок. Може приховувати папки і файли з можливістю розблокування доступу тільки після введення пароля.

Files_Protect - програма для захисту файлів від видалення. Після обробки файлу, ОС не може з ним нічого зробити.

Access Administrator - захист паролем файлів і папок. Дозволяє встановити захист на перегляд, відкриття, зміну, видалення файлів і папок для певного користувача у часовому просторі.

5.2. Засоби захисту вмісту локальних дисків

У цьому питанні наводиться перелік та коротка характеристика програмних засобів для захисту локальних дисків жорсткого диску та програмних продуктів для створення віртуальних логічних дисків з зашифрованою інформацією користувача.

BestCrypt - популярна програма для шифрування даних. Best Crypt створює додатковий логічний диск для зберігання секретних даних, доступ до якого закритий паролем. Вся інформація про логічний диск зберігатиметься в деякому файлі, так званому контейнері. Модуль під назвою SHA-1 Key Generator дозволяє використовувати декілька паролів на один логічний диск. У майбутньому з цим диском можна буде працювати як з реальним логічним диском - запускати Scandisk, форматувати і ставити будь-яку таблицю розміщення файлів, призначати мітку і т.д. Скачати програму BestCrypt можна з сайту http://www.securitylab.ru.

Disk Password Protection - це комплексний захист комп'ютера, дисків, розділів і конфіденційної інформації від несанкціонованого доступу. Дозволяє захистити локальні диски і розділи паролем, приховати і захистити свою інформацію від сторонніх осіб, обмежити перегляд і запуск файлів, захистити паролем завантаження операційних систем на комп'ютері. Password Protection надає користувачу вибір захисту, який йому більше підійде. Перший вид захисту – парольний захист завантаження. Наступний вид захисту - захист розділів – дозволяє сховати розділ від стороннього ока, відображаючи лише вільне місце диска. Захист низького рівня дозволяє заблокувати жорсткий диск від будь-яких спроб читання/запису. Цей захист працює тільки з ATA-сумісними жорсткими дисками і може бути активована тільки з MS-DOS.

StrongDisk – програма для створення логічних дисків з будь-яким ім'ям (скажімо, Z:), з яким користувач зможете працювати як з будь-яким іншим диском файлової системи. Фізично, створений диск буде файлом з довільним ім'ям, який може знаходитися де завгодно, - від жорсткого диска до CD-ROM. Причому цей файл-диск користувач зможете відкрити на будь-якому іншому комп'ютері, після встановлення на нього програми StrongDisk. Таким чином, файл є зашифрованим образом диска. Для шифрування використовуються криптографічні алгоритми з відкритими початковими текстами, що забезпечує надійність захисту. Дістати доступ до захищеної інформації не зможуть ні хакери, ні спецслужби. Для захисту можна використовувати пароль, файл-ключ або електронний USB ключ. Можлива довільна комбінація цих методів - наприклад, використання файл-ключа і пароля. Така комбінація підвищує ступінь захисту, оскільки володіння тільки одним елементом, файл-ключом або паролем, не дає доступу до інформації. Файл-ключ можна потім записати на дискету і носити з собою. Але слід бути обережним - при втраті файл-ключа або пароля ніхто, включаючи розробників програми, не зможе відновити ваші дані. StrongDisk випускається в двох версіях - Pro і Server, перша призначена для звичайних користувачів, а друга розширює можливості StrongDisk для використання в мережі, коли конфіденційна інформація зберігатися на сервері.

5.3.Засоби захисту ПЗ від несанкціонованого копіювання

Програми захисту від копіювання відрізняються певною специфікою, оскільки, з одного боку, повинні дозволяти читання програми для її виконання, а з іншого - забороняти операцію читання для усунення несанкціонованого читання. Таку задачу можна розв'язати двома шляхами:

- дозволити операцію читання (копіювання), але зробити скопійовані програми неробочими (невпізнаними);
- зробити інформацію “важкою для читання” стандартними засобами, але доступною для спеціальних програмних засобів.

З цією метою можна використати “ключову” дискету. На ній зберігаються спеціальні програмні засоби, необхідні для успішного читання (копіювання) файлів, що знаходяться на жорсткому диску. На ключовій дискеті для сприйняття індивідуальності позначають деякі сектори “поганими” (bad) і змінюють структуру запису системної інформації. Одночасно в спеціальні програми керування читанням

(копіюванням) повинні бути закладені функції перевірки цих “унікальних” особливостей.

До таких спеціальних програм можна віднести:

1. PROKOP - складається із програм Codprtct.com, Floprtct.com, Hardinst.com, які встановлюються на вінчестері. В процесі захисту програм користувача від копіювання відбувається прив’язка завантажуваних програм до ключової дискети.
2. PP - система захисту програм від несанкціонованого копіювання складається з модулів: I_and_r.exe, Ptoolf.exe, Ptooli.exe. Захист полягає в зміні EXE-файлів шляхом переміщення з них блока обсягом 512 байт і внесення на це місце спеціальної програми, яка розпізнає умови, задані в момент інсталяції.
3. TEXT PROTECTION - складається з модулів Decod.exe, Deshifr.exe, Pold.exe, які призначені для захисту текстових даних від несанкціонованого копіювання. Ключова дискета створюється шляхом фізичного пошкодження частини секторів - подряпинами, нанесеними голкою.
4. ANTYCOP - має в своєму складі файли: Antycop.com, Antycop.ust, Antycop.zag. Це система захисту файлів, які розміщені на жорстких і гнучких дисках, від несанкціонованого копіювання. Захищені файли прив’язуються до носія і під час копіювання стають непрацездатними. При встановленні захисту враховуються такі параметри, які вводяться користувачем: ключ захисту, кількість інсталяцій (всі копії, які зроблені понад встановлений ліміт, будуть непрацездатними).
5. ЗАХИСТ-МІКРО - функціонує на ПК типу IBM PC/XT або сумісних з ним під керуванням MS DOS. Призначений для захисту файлів від несанкціонованого копіювання і від розповсюдження програмних продуктів, записаних на гнучких магнітних носіях. Для цього використовуються дискети спеціального виготовлення.

Програми, які захищені описаним вище методом, захищаються від копіювання за допомогою стандартних засобів (Copy, Xcopy, Diskcopy, Norton Utilities тощо). Однак, програми захисту, а, значить, і їх “унікальні” характеристики піддаються тиражуванню. Внаслідок цього низка розробників ПЗ відмовляється від продажу програм на дискетах. Вони розміщують своє ПЗ безпосередньо в ОС під час покупки ПК, забезпечуючи тим самим відповідний захист.

5.4. Антивіруси

Найбільш поширеним засобом нейтралізації вірусів є антивірусні програми (антивіруси). Антивіруси, виходячи з реалізованого в них підходу до виявлення і нейтралізації вірусів, прийнято ділити на наступні групи:

- детектори;
- фаги;
- вакцини;
- щеплення;

- ревізори;
- монітори.

Детектори забезпечують виявлення вірусів за допомогою проглядання виконуваних файлів і пошуку так званих сигнатур - стійких послідовностей байтів, наявних в тілах відомих вірусів. Наявність сигнатури в якому-небудь файлі свідчить про його зараження відповідним вірусом. Антивірус, що забезпечує можливість пошуку різних сигнатур, називають полідетектором.

Фаги виконують функції, властиві детекторам, але, крім того, «виліковують» інфіковані програми за допомогою «вирізування» вірусів з їх тіл. По аналогії з полідетекторамі, фаги, орієнтовані на нейтралізацію різних вірусів мають назву поліфагамі.

На відміну від детекторів і фагів, вакцини за своїм принципом дії подібні до вірусів. Вакцина імплантується в програму, що захищається, і запам'ятовує ряд кількісних і структурних характеристик програми. Якщо вакцинована програма не була до моменту вакцинації інфікованою, то при першому ж після зараження запуску відбудеться наступне. Активізація носія віруса приведе до отримання вірусом керування, який, виконавши свої цільові функції, передасть керування вакцинованій програмі. У останній, в свою чергу, спочатку керування отримає вакцина, яка виконає перевірку відповідності запам’ятованих нею характеристик аналогічним характеристикам, отриманим у певний момент. Якщо вказані набори характеристик не співпадають, то робиться висновок про зміну тексту вакцинованої програми вірусом. Характеристиками, використовуваними вакцинами, можуть бути довжина програми, її контрольна сума і т.д. Принцип дії щеплень базується на обліку тієї обставини, що будь-який вірус, як правило, позначає програми, що інфікуються, якою-небудь ознакою з тим, щоб не виконувати їх повторне зараження. У іншому випадку мало б місце багатократне інфікування, супроводжується істотним і тому таким, що легко виявляється збільшенням об'єму заражених програм. Щеплення, не вносячи ніяких інших змін в текст програми, що захищається, позначає її тією ж ознакою, що і вірус, який, таким чином, після активізації і перевірки наявності вказаної ознаки, вважає її інфікованою і «залишає у спокої».

Ревізори забезпечують стеження за станом файлової системи, використовуючи для цього підхід, аналогічний реалізованому у вакцинах. Програма-ревізор в процесі свого функціонування виконує до кожного виконуваного файлу порівняння його поточних характеристик з аналогічними характеристиками, отриманими в ході попереднього перегляду файлів. Якщо при цьому виявляється, що, згідно наявної системної інформації, файл з моменту попереднього перегляду не оновлювався користувачем, а порівнювані набори характеристик не співпадають, то файл вважається інфікованим. Характеристики виконуваних файлів, отримувані в ході чергового перегляду, запам’ятовуються в окремому файлі (файлах), у зв'язку з чим збільшення довжин виконуваних файлів, що має місце при вакцинації, в даному випадку не відбувається. Інша відмінність ревізорів від вакцин полягає в тому, що кожен перегляд виконуваних файлів ревізором вимагає його повторного завантаження.

Монітор є резидентною програмою, що забезпечує перехоплення потенційно небезпечних переривань, характерних для вірусів, і що вимагає у користувачів підтвердження на виконання операцій, наступних за перериванням. У разі заборони або відсутності підтвердження монітор блокує виконання призначеної для користувача програми.

Антивіруси розглянутих типів істотно підвищують вірусозахищеність окремих ПК і обчислювальних мереж в цілому. Відомо ряд недоліків при використанні антивірусів. У зв'язку з цим необхідна реалізація альтернативних підходів до нейтралізації вірусів: створення операційних систем, що володіють високою вірусозахищеністю в порівнянні з найбільш «дружньою до вірусів» MS DOS, розробка апаратних засобів захисту від вірусів і дотримання технології захисту від вірусів.

5.5. Міжмережеві екрани (браундмауери)

Міжмережний екран - це набір пов’язаних між собою програм, що встановлюються на комп’ютері, який містить ресурси власників та захищає їх від користувачів із зовнішньої мережі, наприклад Internet. Власник комп’ютера, що має вихід в Internet, встановлює міжмережний екран, щоб запобігти одержанню сторонніми конфіденційних даних, котрі зберігаються на комп’ютері, а також для контролю за зовнішніми ресурсами, до яких мають доступ інші користувачі даної комп’ютерної системи.

Ряд завдань стосовно захисту від найбільш імовірних атак для внутрішніх мереж здатні вирішувати тільки міжмережні екрани. У вітчизняній літературі частіше зустрічаються терміни іноземного походження: брандмауер і firewall. Поза комп’ютерною сферою брандмауером (чи firewall) називають протипожежну стіну, зроблену з вогнестійких матеріалів, щоб перешкодити поширенню пожежі. У сфері використання комп’ютерних технологій міжмережний екран становить собою бар’єр, що захищає від умовної пожежі - спроб зловмисників несанкціоновано вторгнутися у внутрішню мережу для вчинення протиправних дій. Міжмережний екран покликаний створити безпечний доступ до зовнішньої мережі та обмежити доступ зовнішніх користувачів до внутрішньої мережі.

Вибір оптимальних міжмережних екранів (firewalls) – це, головним чином, питання правильного співвідношення між вимогами користувачів стосовно доступу та вірогідності несанкціонованого доступу. В ідеалі система має запобігати будь-якому несанкціонованому вторгненню. Однак, враховуючи широкий спектр Web-сервісів, необхідних користувачам, ftp, telnet, SNMP, Network File System, IP телефонія, електронна пошта тощо, досягнути певного рівня запобігання несанкціонованому втручанню дуже важко.

Основна мета firewall - не допустити несанкціонованого доступу в локальну мережу через Internet шляхом перегляду пакетів даних і використання спеціальних засобів підтвердження повноважень для додатків. Перегляд пакетів проводиться з метою блокування підозрілих видів трафіка. Функції підтвердження повноважень, які орієнтовані на прикладні програми, здійснюють повний контроль і перевірку на допустимість усіх вхідних і вихідних даних. Усе більше фахівців у галузі захисту

інформації приходить до розуміння, що використання firewall для захисту локальної мережі значно зменшує ризик несанкціонованого втручання через Internet.

Ряд firewall дозволяють також організовувати віртуальні корпоративні мережі Virtual Private Network (VPN), що об'єднують декілька локальних мереж, включених у Internet в одну віртуальну мережу. Така система дозволяє організувати прозоре для користувачів з'єднання локальних мереж, зберігаючи секретність і цілісність інформації, що передається за допомогою шифрування. При цьому під час передачі даних по Internet шифрується не лише інформація, призначена для користувача, але і мережна - мережні адреси, номери портів тощо. VPN створює захищене з'єднання через Internet. Зараз подібне використовується в багатьох технологіях.

5.6. Засоби стеганографічного захисту

Наведемо перелік та коротку характеристику найбільш розповсюджених засобів стеганографічного захисту:

1. Steganos v.1.4. – програма, яка може приховувати інформацію, використовуючи стеганографічні методи, і шифрувати її за допомогою технології криптографії. Призначена для роботи в середовищі DOS. Дає можливість приховувати всі види файлів у графічних файлах формату BMP, у звукових файлах формату WAV і VOC, текстових ASCII. Може не вилучати файл "повідомлення" і створювати резервну копію файла "контейнера". Отриманий в результаті цього перетворення файл формату BMP можна перевести в інші графічні формати, не руйнуючи структуру зображення, наприклад, GIF, і навпаки без втрати закодованої інформації.
2. Hideseek – програма може приховувати файл "повідомлення" лише у графічних файлах GIF, а найбільше розширення екрану, з яким вона може працювати 320*480 пікселів.
3. Hide 4PGP v.1.0 – програма, призначена для роботи в середовищі DOS. Може приховувати всі види файлів у графічних файлах формату BMP (256-кольорове або 24-бітне зображення не повинно бути стиснутим) і звукових файлах формату WAV і VOC.
4. PGE v.1.0 – програма, що також працює в середовищі DOS. Може приховувати всі види файлів у графічних файлах формату GIF (87,89) і JPG (JFIF).
5. S-Tools v.4 for Windows – програмний засіб, що дає можливість приховувати всі види файлів у графічних файлах формату BMP і GIF і звукових файлах формату WAV. При роботі створює новий файл з закодованою інформацією. Має багатовіконний режим роботи і може одночасно кодувати кілька файлів. Використовує кілька стеганографічних алгоритмів, і користувач може вибрати найбільш придатний.
6. White Noise StormTM – призначений для роботи в середовищі DOS. Може приховувати всі види файлів у графічних файлах формату BMP і GIF і звукових файлах формату VOC. У ході тестування найпереконливіші результати продемонстрували Steganos for Windows 95 і S-Tools v.4.

ТЕМА 6. ЗАСОБИ ЗЛАМУ СИСТЕМ ЗАХИСТУ

6.1. Проблеми існування засобів зламу захистів ПЗ

Проблеми існування засобів зламу захистів ПЗ породжені багатьма причинами. При цьому практично всі ці причини ґрунтуються на простих людських бажаннях: прагненню до влади, грошам, безпеці, славі, отриманню задоволення і відновленню справедливості. Розберемо ці бажання докладніше.

Влада і гроші. Якщо яка-небудь людина або компанія використовували засіб захисту інформації, а супротивникові вдалося знайти у ньому «диру», супротивник отримує над користувачами певну владу. Ця влада може виражатися по-різному: як можливість читати засекречені повідомлення, підроблювати чужий підпис або, наприклад, шантажувати того, хто використовує зламану систему. Зрозуміло, супротивник піддає аналізу систему захисту, до розробки якої він сам не має ніякого відношення - основна мета аналізу полягає не в перевірці стійкості захисту, а в отриманні хоч якого-небудь способу тиску на сторону, що застосовує вразливу технологію. Тому зазвичай шукається найбільш простій спосіб атаки, що приносить плоди за мінімальний час і при мінімальних матеріальних витратах.

Зараз, коли технології захисту інформації увійшли до повсякденного життя і застосовуються в тій або іншій формі майже всіма людьми (введення PIN-коду або пароля - це вже використання засобів захисту), можливостей для атаки стало значно більше. Це обумовлено як існуванням найрізноманітніших (і далеко не завжди безпечних) засобів захисту, так і порівняно легким виявленням об'єктів, які можна спробувати атакувати, - з появою Інтернету захищені системи і дані можна зустріти буквально на кожному кроці.

Якщо супротивникові вдалося знайти слабке місце в системі захисту, його подальші дії можуть бути самими різними залежно від характеру виявленої вразливості. Мабуть, найстрашніші наслідки можуть виникнути, якщо супротивник навчився читати зашифроване листування. При цьому він не надає активної дії, і його присутність ніяк не виявляється. Користувачі можуть ніколи не дізнатися, що вміст їх повідомлень став відомий супротивникові. Якщо супротивник зможе, наприклад, підроблювати чужий цифровий підпис, то перший раз подібні дії цілком можуть залишитися непоміченими. Але якщо дійсний власник підпису виявить, що його підпис знаходиться на документі, який він бачить перший раз в житті, найправильнішим рішенням буде відкликати ключ підпису, після чого супротивник втрачає всю отриману в результаті зламу перевагу.

Достатньо поширений спосіб використання інформації про недоліки в організації безпеки деякої системи - продемонструвати вразливості і запропонувати допомогу в їх усуненні (зрозуміло, за винагороду). Проте в подібній ситуації супротивник сильно ризикує - при численних контактах з власниками зламаної системи досить складно зберігати анонімність, а значить, неабияк зростають шанси бути спійманим і отримати звинувачення в неправомірному доступі до інформації і шантажі.

Нерідко зустрічається і ситуація, коли супротивник, що не має можливості виконати аналіз захищати самостійно (наприклад через нестачу технічних знань), за гроші наймає фахівців, які і проводять всі необхідні дослідження.

Існує велика категорія людей, що мріють прославитися будь-яким доступним ним способом. Цих способів велика кількість, і вони дуже сильно відрізняються один від одного.

Більшість людей, що увійшли до історії, впродовж всього життя дуже багато трудилися і саме цим здобули собі світове визнання. Так можна почати з розробки власної версії інтерпретатора мови BASIC або емулятора терміналу і через декілька років стати лідером гігантської софтверної (що проводить програмне забезпечення) імперії або найкрупнішого відкритого програмного проекту в історії. Але цей шлях не дуже надійний - мільйони людей день за днем працюють не покладаючи рук, а слава приходить лише до одиниць.

Ось і виходить, що людині або компанії, що виявила слабкість в широко поширеному засобі захисту, досить залишити повідомлення про своє відкриття в Інтернеті. Це повідомлення в лічені години виявиться рознесеним по всьому світу, з'явиться на тисячах сайтів, потрапить через поштові розсилки мільйонам користувачів, а то і виплеснеться в пресі і на телеекранах. І якщо автор повідомлення не забуде вказати своє ім'я у нього є багато шансів залишитися в пам'яті людей, а значить, і в історії.

Проте у жадання слави є і інші прояви. У комп'ютерному світі існує досить численне співтовариство людей, що займаються зломом комерційного програмного забезпечення. Дуже часто ці люди об'єднуються в групи, вибирають собі псевдоніми, а групам - назви, і починають змагатися з іншими подібними командами.

Мета змагання, як правило, - це викласти у вільний доступ зламану нову версію програмного продукту раніше, ніж це зробить інша група. Зламом вважається те, що програма може використовуватися без реєстрації або генератор реєстраційних кодів додається до програми. Зрозуміло, члени групи не залишають своїх реальних імен або адрес, і фактично всю популярність вони створюють і закріплюють виключно в рамках свого співтовариства.

Але окрім людей, що рвуться до слави, влади або грошей, є і просто цікаві. В більшості своїй вони цікавляться не результатом, а процесом дослідження. Для них не важливо, чи вдасться виявити вразливість. Адже сам процес дослідження може приносити величезне задоволення. Саме з таких цікавих, прагнучих проникнути в саму суть, і виходять дуже хороші вчені. А наукове дослідження засобів захисту дозволяє збирати інформацію, необхідну для всестороннього розгляду сукупності технологій безпеки, виявлення їх переваг і недоліків, а також вироблення рекомендацій по побудові надійніших систем.

Іноді дослідження або навіть злам захисту проводиться з метою забезпечення безпеки держави або відновлення законності. Наприклад, якщо інформація на комп'ютері особи, підозрюваної в скоєнні злочину, повністю або частково зашифрована, то суд може видати дозвіл на аналіз використаного засобу захисту і на витягання прихованих даних.

Дуже схожа ситуація і з дослідженням шкідливих програм: вірусів, троянських коней. Розробники антивірусних програм проводять аналіз внутрішнього коду

вірусів для розробки ефективних методів протидії та інші. Це не є дослідженням захисту в чистому вигляді, але, безумовно, відноситься до інформаційної безпеки.

6.2. Класифікація засобів зламу захистів ПЗ

Всі засоби дослідження ПЗ можна розбити на 2 класи: статичні і динамічні. Перші оперують початковим кодом програми як даними і будують її алгоритм без виконання, другі - вивчають програму, інтерпретуючи її в реальному або віртуальному обчислювальному середовищі. Звідси витікає, що перші є більш універсальними в тому сенсі, що теоретично можуть отримати алгоритм всієї програми, у тому числі і тих блоків, які ніколи не отримають управління. Динамічні засоби можуть будувати алгоритм програми тільки на підставі її трасування, отриманого при певних вхідних даних. Тому завдання отримання повного алгоритму програми в цьому випадку еквівалентне побудові вичерпного набору текстів для підтвердження правильності програми, що практично неможливе, і взагалі при динамічному дослідженні можна говорити тільки про побудову деякої частини алгоритму.

Два найбільш відомих типи програм, призначених для дослідження ПЗ, якраз і відносяться до різних класів: це відлагоджувач (динамічний засіб) і дизасемблер (засіб статистичного дослідження). Якщо перший широко застосовується користувачем для відлагоджування власних програм і завдання побудови алгоритму для нього вторинні і реалізуються самим користувачем, то другим призначений виключно для їх вирішення і формує на виході асемблерний текст алгоритму.

Крім цих двох основних інструментів дослідження, можна використовувати:

- дискомпілятори, програми, що генерують з виконуваного коду програму на мові високого рівня;
- трасувальники, програми що спочатку запам'ятовують кожну інструкцію, що проходить через процесор, а потім переводять набір інструкцій у форму, зручну для статичного дослідження, автоматично виділяючи цикли, підпрограми і інше;
- «системи що стежать» - запам'ятовують і аналізують трасу вже не інструкції, а інших характеристик, наприклад викликаних програмою переривання.

Крім класифікації засобів подолання захисту ПЗ на статичні та динамічні існують інші класифікації. Наприклад класифікація, на пасивні і активні.

Пасивні інструменти не надають ніякої дії ні на саму досліджувану програму, ні на її оточення. Активні інструменти, навпаки, взаємодіють з програмою під час її виконання. З цього випливають два важливі зауваження:

- активний інструментарій може дати значно більше інформації, чим пасивний, оскільки дозволяє оцінювати полягання програми в динаміці;
- присутність активних інструментів може бути виявлена захистом і може привести до відповідних дій. Виявити або запобігти застосуванню пасивних засобів програма не в змозі.

Активні інструменти, у свою чергу, можуть використовуватися тільки для протоколювання (моніторингу) ходу виконання програми або для явної дії на хід

виконання програми: підміни даних, виправлення результатів перевірки умов і т.д. Також активний інструментарій може проводити віртуалізацію середовища виконання програми. Тобто програма знаходиться в повній впевненості, що виконується в звичайних умовах, а насправді кожен її крок, включаючи дії по пошуку активних засобів аналізу, знаходиться під повним контролем дослідника.

Ще один спосіб класифікації інструментів - по області застосування, тобто що саме піддається дослідженню:

- виконуваний код;
- ресурси ПЗ;
- дискові файли;
- записи в реєстрі;
- інформація в оперативній пам'яті;
- інформація, що отримується від пристроїв введення;
- інформація, що посилається на пристрої введення;
- повідомлення і дані, що пересилаються між процесами і всередині процесу;
- дані, що передаються по мережі;
- виклики бібліотечних функцій.

6.3. Програмний інструментарій для зламу програмного забезпечення

При дослідженні програмного фахівцеві доводиться користуватися різними інструментами, що дозволяють ефективніше виконувати поставлені завдання. Ці інструменти здатні підняти продуктивність праці аналітика у декілька разів. Про існуючі інструменти і їх можливості корисно знати і розробникам засобів безпеки, щоб ефективніше створювати труднощі аналітикам, які намагатимуться знайти дірки в захисті.

Два основні способи дослідження програмного коду - це дизасемблювання та відлагодження.

Використовуючи дизасемблер, можна подивитися, як написана програма, які команди і в якій послідовності повинні виконуватися, до яких функцій йде звернення і т.д. В загальному випадку дизасемблер не здатний відновити початковий текст програми, написаної на мові високого рівня, такому як C або Pascal. Результатом роботи дизасемблера є (як можна здогадатися з назви) еквівалентний текст на мові асемблера. Для осмислення асемблерного тексту аналітик, зрозуміло, повинен бути добре знайомий з мовою асемблера і з особливостями того середовища, в якому повинна виконуватися програма, що дизасемблюється. Дизасемблер є пасивним інструментом - він ніяк не впливає на програму. Найпотужнішим дизасемблером з тих, що існують на сьогоднішній день є IDA Pro (Interactive DisAssembler), розроблений компанією DataRescue.

Для захисту від дизасемблерів застосовуються різні методи. Наприклад, якщо код програми запакований або зашифрований, дизасемблер не зможе побачити в досліджуваному файлі справжні інструкції програми. Але захищену таким чином програму можна спочатку розшифрувати і розпакувати, а потім скористатися дизасемблером.

Для більшості популярних засобів упаковки і шифрування коду виконуваних модулів давно розроблені автоматичні або напівавтоматичні пакувальники. А для того, щоб дізнатися, як саме запакований той або інший модуль, можна скористатися спеціальними програмами-ідентифікаторами, які за деякими характерними ознаками здатні ідентифікувати назву і версію використовуваного засобу захисту, а також версію компілятора, що застосовувався при розробці програми.

Отже реальну складність для дизасемблювання представляють тільки програми, які розшифровують фрагменти коду динамічно, не допускаючи одноразової присутності в пам'яті розшифрованого коду цілком.

В деяких випадках дизасемблер відмовляється працювати з виконуваним файлом, якщо якісь заголовки файлу сформовані з порушенням специфікації, але даний спосіб також не є надійним.

Іноді код програми модифікується так, щоб дизасемблерну послідовність команд було дуже важко аналізувати. Наприклад, сусідні команди розносяться в різні місця, а правильність виконання організовується за рахунок великого числа безумовних переходів. Або між командами вставляються довільні фрагменти коду, що не впливають на результати обчислень, але що віднімають у людини, що виконує аналіз багато часу.

Правда, варто відзначити, що, наприклад, дизасемблер IDA Pro має досить потужні додаткові засоби (модулі, що підключаються та мова сценаріїв), надаючи тим самим можливість нейтралізувати всі спроби протидії дизасемблюванню і подальшому аналізу.

Відлагоджувач, на відміну від дизасемблера, є активним інструментом і дозволяє прослідкувати процес виконання по кроках, отримуючи у будь-який момент всю інформацію про поточний стан програми або вносити зміни в порядок її виконання. Зрозуміло, відлагоджувач здатний показувати дизасембльовані інструкції, стани регістрів, пам'яті і багато що інше. Але наявність відлагоджувача, через його активність, може бути виявлена програмою або тією її частиною, яка відповідає за захист..

Відлагоджувачі бувають трьох основних типів: рівня користувача, рівня ядра і ті що емулюють.

Відлагоджувачі призначеного для рівня користувача (User-level Debuggers) мають практично ті ж можливості, що і відлагоджувана програма. Вони використовують Debugging API, що входить до складу операційної системи і з його допомогою здійснюють контроль над об'єктом відлагодження.

Відлагоджувачі призначеного для рівня користувача входять до складу багатьох середовищ розробки, таких як Visual Studio. Вони використовуються для дослідження незахищених програм, але можуть бути легко виявлені.

Відлагоджувачі рівня ядра (Kernel-mode Debuggers) вбудовуються всередину операційної системи і мають значно більше можливостей, чим відлагоджувачі призначені для рівня користувача. З ядра операційної системи можна контролювати багато процесів, не доступних іншими способами. Одним з найпотужніших і часто використовуваних відлагоджувачів рівня ядра є Softlce, розроблений в компанії NuMega Labs (Compuware Corporation). Але і відлагоджувачі рівня ядра майже завжди можуть бути виявлені з програми, що не має доступу до ядра. Хоча для

Softlce, наприклад, був розроблений модуль розширення IceExt, що дозволяє, серед іншого, непогано приховувати наявність відлагоджувача в пам'яті.

Емулюючі відлагоджувачі, мабуть, є найкращим засобом дослідження коду програм. Такі відлагоджувачі емулюють виконання всіх потенційно небезпечних дій, які програма може використовувати для виходу з-під контролю дослідника. Проте основна проблема створення емулюючих відлагоджувачів полягає в тому, що іноді їм доводиться емулювати реальне периферійне устаткування, а це надзвичайно складне завдання. Можливо тому зараз немає доступних широкій аудиторії емулюючих відлагоджувачів, хоча існує як мінімум два пакети для створення віртуальних комп'ютерів: VMware, розроблений однойменною компанією, і VIRTUALPC, створений в Connectix Corp., що недавно перейшов у власність корпорації Microsoft.

Для того, щоб знайти яку-небудь інформацію в ресурсах виконуваного модуля, можна скористатися таким інструментом, як редактором Microsoft Visual Studio. Але краще використовувати спеціалізовані редактори ресурсів, яких існує досить багато. Ці редактори, як правило, дозволяють проглядати ресурси відомих типів (наприклад текстові рядки, ікони, картинки і описи діалогів) в природному вигляді, а незнайомі ресурси - у вигляді шістнадцяткового дампу. Корисним може бути і модуль розширення Resource Browser, що підключається до файлового менеджера FAR. Цей модуль дозволяє проглядати ресурси у вигляді ієрархічного фрагмента файлової системи з підкаталогами і файлами. При використанні такого представлення ресурсів дуже зручно проводити пошук.

Ще одним інструментом дослідників захисту програм є програми-монітори, що протоколюють спроби доступу до реєстру і дискових файлів. Монітор реєстру (Registry Monitor) і монітор доступу до файлів (File Monitor) - це активні інструменти. Пасивні інструменти просто запам'ятовують стани реєстру або файлів і по розбіжностях дозволяють визначити, що саме змінилося. Простий спосіб виявити змінені файли, не зберігаючи їх повністю - підрахувати і запам'ятати значення хеш-функції від вмісту кожного файлу до і після виконання процесу, що вносить зміни, а потім порівняти два набори хеша між собою. Саме на такому принципі будувалася робота антивірусного монітора Adlnf, що функціонував під DOS. Якщо вдалося встановити, які саме файли піддаються зміні, їх можна заархівувати, а потім, після, внесення змін, порівняти старий і новий вміст. Для цього можна скористатися спеціальними інструментами або утилітою FC (File Compare), що входить до складу Windows. FC дозволяє порівнювати як двійкові, так і текстові файли.

З реєстром працювати не так зручно, як з файлами, через те, що реєстр Windows є досить складною деревовидною структурою. Зате об'єм даних, що зберігаються в реєстрі, порівняно невеликий - декілька десятків мегабайт. Тому можна просто обійти всі гілки реєстру і зберегти значення у власному форматі. Одна з програм, що дозволяють це зробити, - Advanced Registry Tracer (ART), розроблена компанією Elcomsoft Co. Ltd. ART надає користувачеві можливість зберегти декілька "знімків" поточного стану реєстру. Потім окремі знімки реєстру можна попарно порівнювати отримуючи списки доданих, змінених і видалених ключів і значень.

Для доступу до пам'яті процесу можна використовувати функції стандартного Win32 API. У операційних системах сімейства NT деякі процеси можуть бути

запущені з атрибутами безпеки, що не дозволяють простим користувачам діставати доступ до процесу. Але це робиться для того, щоб захистити ядро операційної системи в розрахованому на багато користувачів середовищі. А у випадках дослідження програм, як правило, користувач може поставити собі будь-які права доступу що дозволить одержати доступ до пам'яті досліджуваного процесу.

Існують також спеціальні програми, що дозволяють не просто зберегти фрагмент пам'яті на диск, але записати його у форматі Portable Executable (PE). Така операція називається отриманням дампу виконуваного файлу і застосовується для отримання розшифрованої і розпакованої версії досліджуваної програми.

Пристрою введення-виведення неможливо досліджувати пасивними засобами, зате звернення до них можна протоколювати. Програми для протоколювання натиснень на клавіатуру зазвичай називають клавіатурними шпигунами і застосовують для перехоплення паролів, що вводяться користувачем. Проте цей прийом застосовується або троянськими програмами, або при спробі вивідати секретну інформацію у людини, але ніяк не при дослідженні засобів захисту. Протоколювання введення і виведення в COM- і LPT-портах може здійснюватися, наприклад, за допомогою програми PortMon, розробленої Марком Русиновічем (Mark Russinovich) з компанії Syslnternals.

Дуже багато процесів всередині Windows управляються за допомогою повідомлень. Зрозуміло, існують програми, що дозволяють відстежувати і протоколювати, які саме повідомлення були передані тому або іншому процесу. Однією з таких програм є Microsoft Spy++, що входить до складу Visual Studio. Spy++ дозволяє із списку або інтерактивно на екрані вибрати вікно, повідомлення для якого необхідно відстежувати, і проглянути його властивості. Можна також задати, які саме повідомлення повинні протоколюватися і які їх атрибути показуватимуться. Протокол може відразу записуватися у файл.

Для перехоплення даних, що передаються по мережі, використовуються спеціальні програми - сніфери (sniffer). Як правило, сніфери здатні перехоплювати всі повідомлення, що передаються між пристроями всередині фізичного сегменту мережі, до якого підключений комп'ютер з сніфером. Не дивлячись на те, що вже багато років існують протоколи, що дозволяють приховати від супротивника всю важливу інформацію при передачі по мережі, до цих пір використовуються деякі протоколи, в яких, наприклад, паролі користувачів передаються у відкритому вигляді.

ТЕМА 7. ОСОБЛИВОСТІ ЗЛАМУ АВТОМАТИЗОВАНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

7.1. Поняття та характеристика програмних засобів, призначених для незаконного проникнення в автоматизовану систему

Автоматизовані електронно-обчислювальні системи, як відомо, в інженерно-технічному аспекті складаються з підсистеми комп'ютерних програм, їх пакетів (Software), підсистеми електронно-технічних засобів (Hardware) та автоматизованих баз даних, які можуть складати автоматизований банк даних (знань). Незаконне проникнення до таких систем з метою перекручення або знищення інформації чи носіїв інформації можливе за допомогою спеціально виготовлених комп'ютерних програм та технічних засобів. Через тісну взаємопов'язаність і взаємообумовленість можливе їх конструктивне поєднання, тобто створення "апаратно-програмного комплексу". Проте існують технічні засоби, що не містять комп'ютерних програм, але можуть шкодити роботі автоматизованої системи через електромагнітні випромінювання надвисокої частоти або силу електричного струму тощо.

Серед засобів для незаконного проникнення в автоматизовані системи, що здатні спричинити перекручення або знищення інформації чи носіїв інформації, найбільшого поширення набули так звані "комп'ютерні віруси". Їх ще називають "шкідливими комп'ютерними програмами" або "комп'ютерними програмами, які вчиняють шкоду". Багато з них у автоматичному режимі впроваджуються в інші комп'ютерні програми, розмножуються в певних умовах і пошкоджують інформацію в автоматизованих системах.

Назву "віруси" вони отримали через схожість з біологічними вірусами, зокрема через здатність до автоматичного розмножування. При цьому не тільки в тій системі, де вони утворені, а й у інших системах певного типу.

За підрахунками експертів, у "комп'ютерному світі", "кіберпросторі" існує вже понад 3500 модифікацій комп'ютерних вірусів. Кількість їх постійно збільшується.

Комп'ютерна програма, всередині якої знаходиться комп'ютерний вірус, фахівцями називається "зараженою" або "інфікованою". Коли така комп'ютерна програма отримує виклик на початок роботи, то спочатку система управління комп'ютером дає команду на запуск комп'ютерної програми-вірусу. Після цього комп'ютерна програма - вірус знаходить та інфікує інші комп'ютерні програми або виконує будь-які інші шкідливі дії. Наприклад, відбувається знищення файлів або таблиці розміщення файлів на відповідному магнітному диску; переповнюється непотрібними командами чи даними ("засмічується") постійна чи оперативна пам'ять тощо.

Для маскування дії вірусу щодо зараження інших комп'ютерних програм та заподіяння шкоди можуть виконуватися різні комп'ютерні команди, які не завжди починають діяти відразу, а лише при дотриманні певних умов. Наприклад, після того, як вірус виконає потрібні йому дії, він передає управління тій комп'ютерній програмі, в якій він знаходиться, і вона працює також як звичайна до певного часу або кількості повтору операцій з нею. Через це зовні робота зараженої програми

виглядає так само, як і незараженої. Багато різновидів вірусів побудовані так, що при запуску зараженої програми вірус залишається резидентно (тобто до перезавантаження операційної системи) в електронній пам'яті комп'ютера. При цьому час від часу відбувається зараження комп'ютерних програм чи виконуються шкідливі дії з інформацією в АС.

Потрібно відзначити, що тексти комп'ютерних програм і електронних документів, інформаційні файли баз даних, таблиці табличних процесорів тощо можуть бути використані як носії для зараження вірусом, який може їх тільки зіпсувати в поєднанні з іншим програмно-математичним забезпеченням комп'ютера ("бінарні віруси").

Всі дії комп'ютерного вірусу можуть виконуватися досить швидко і без подання будь-яких помітних повідомлень, тому користувачеві дуже важко помітити, що в комп'ютері відбувається щось незвичайне. Поки в комп'ютері заражено відносно мало програм, наявність вірусу може бути практично непомітною. Та через деякий час у комп'ютері починаються шкідливі процеси. Наприклад, деякі комп'ютерні програми перестають працювати або починають працювати неправильно; на екран відеотермінала (монітора) виводяться сторонні повідомлення, символи тощо; робота комп'ютерних програм істотно сповільнюється; деякі файли виявляються зіпсованими; іноді комп'ютер самовільно перезавантажується, в результаті чого знищується вся попередньо поміщена в нього машинна інформація. На цей момент уже значна кількість (або навіть більшість) комп'ютерних програм, як правило заражені комп'ютерним вірусом, а деякі файли і магнітні диски - зіпсовані. Нерідко буває гірше - заражені комп'ютерні програми з одного комп'ютера переносяться (за допомогою дискет, інших носіїв або через мережу) в інші комп'ютери.

Деякі види вірусів поводяться ще більш підступно. Вони спочатку непомітно заражають значну кількість комп'ютерних програм або магнітних дисків комп'ютера, а потім уже завдають серйозні пошкодження.

Кожний конкретний різновид комп'ютерного вірусу може вражати тільки один або кілька типів електронних файлів. Найчастіше зустрічаються комп'ютерні віруси, що заражають файли типу "*. com", "*. exe" тощо. Деякі комп'ютерні віруси заражають як файли, так і завантажувальні ланки (області) магнітних дисків комп'ютерів.

Набули поширення віруси, що мають файлову систему на магнітному диску. Такі віруси серед фахівців називають DIR-віруси. Вони ховають своє "тіло" в деяку ділянку магнітного диска та позначають її в спеціальній комп'ютерній таблиці розміщення електронних файлів (FAT) як кінець файла. Щоб запобігти виявленню вірусу, порушники застосовують досить хитрі прийоми маскування. Розглянемо деякі з них.

- "Невидимі" віруси. Деякі віруси (файлові та завантажувальні) уникають їх виявлення тим, що перехоплюють звернення операційної системи до заражених файлів і областей магнітного диска та видають їх у початковому (незараженому) вигляді. В цьому випадку ефект спостерігається тільки на першому зараженому комп'ютері, на другому ж зміни у файлах і

завантажувальних ланках магнітного диска можна легко виявити за допомогою відповідних комп'ютерних програм.

- Самомодифіційні віруси. Інший спосіб, який застосовується для того, щоб приховати виявлення - модифікація вірусу. Багато з вірусів зберігають значну частину свого "тіла" в закодованому вигляді, щоб за допомогою спеціальних комп'ютерних програм – дизасемблерів, не можна було розібратися в механізмі їх роботи. Самомодифікуючі віруси використовують цей прийом і часто змінюють параметри цього кодування, а також і свою стартову частину, яка служить для розкодування інших команд вірусу. Таким чином, у "тілі" подібного вірусу немає постійного ланцюжка запису кількості одиниць виміру його величини (байтів), за яким можна було б ідентифікувати його. Це, природно, ускладнює знаходження таких вірусів за допомогою спеціальних пакетів комп'ютерних програм - детекторами комп'ютерних вірусів.

З метою систематизації всі комп'ютерні віруси класифікуються фахівцями за певними ознаками на кілька груп. Виділяються такі групи комп'ютерних вірусів:

- Завантажувальні комп'ютерні віруси - вражають завантажувальні сектори комп'ютерної пам'яті.
- Файлові комп'ютерні віруси - вражають файли, що виконують комп'ютерні команди на зразок "COM", "EXE", "SYS", "BAT"-файли і деякі інші.
- Комбіновані комп'ютерні віруси - зараження відбувається при завантаженні комп'ютера з носія машинної інформації (дискети тощо), що містить вірус, та при завантаженні файлів.

7.2. Атаки на автоматизовані системи в глобальній інформаційній мережі Internet

Одним з широко поширених типів атак на автоматизовані системи через Internet є атаки за допомогою методів соціальної інженерії. Зловмисник, який використовує методи соціальної інженерії, в основному виходить з штучної передумови, що в нього виникли проблеми і йому потрібна допомога іншого оператора. Буває навпаки, коли проблеми виникають у легального користувача системи, і він просить зловмисника допомоги. Допомагаючи користувачу вирішити проблему, такий зловмисник без особливих зусиль може взнати, наприклад, пароль доступу. Така атака називається "атакою за допомогою зворотної соціальної інженерії" (ЗСІ) і складається з трьох частин:

- диверсія - це перший короткий контакт з певним комп'ютером, під час якого навмисно створюється якась неполадка, що вимагає усунення;
- реклама - інформування користувача про те, що тільки зловмисник може подолати ці неполадки в комп'ютері;
- допомога - спілкування з користувачем, у процесі якого вирішуються проблеми останнього, а також питання, пов'язані з отриманням необхідної інформації, наприклад, пароля доступу.

Успішні атаки соціальної інженерії можливі там, де є вразливі місця в політиці безпеки. Політика безпеки - набір законів, правил і практичних рекомендацій, на

основі яких будується управління, захист та розподіл критичної інформації в автоматизованій системі. Вона має охоплювати всі особливості процесу обробки інформації, визначаючи роботу системи в різних ситуаціях. Кожна атака з використанням соціальної інженерії хоч і використовує загальні принципи соціальної інженерії, проте обов'язково враховує і виявлені специфічні особливості досліджуваної інформаційної системи.

Атака за допомогою методів соціальної інженерії завжди починається з ретельної розробки планів вторгнення. Для отримання інформації, яка може виявитися корисною при несанкціонованому втручанні, часто використовуються різноманітні психологічні прийоми. Наприклад, правопорушник дзвонить у офіс і люб'язно розпитує співробітників про переваги нового пакету послуг, ставлячи питання відносно системи безпеки.

Інший прийом, що часто зустрічається – "розгрібання сміття". Дуже часто співробітники компаній досить легковажно викидають внутрішні телефонні довідники, технічну документацію, диски та ще багато чого. Це справжня "золота жила" для зловмисника. Також доступ можна отримати і через телефонну систему технічної підтримки. Іноді застосовуються і більш грубі методи, на зразок здирства, силового тиску, шантажу співробітників. Правопорушник, використовуючи методи соціальної інженерії для атаки на інформаційну систему, збирає інформацію у будь-який доступний спосіб - вивчає, наприклад, відкриті документи, фінансові звіти, технічну документацію. Його цікавлять дані про операційні системи, що використовуються, продукти, які є основою інформаційної системи, номери телефонів, а також фізичні адреси центрів зберігання даних і телефонних вузлів.

Провівши попередні дослідження, правопорушник (група правопорушників) зазвичай складає схему мережі інформаційної системи, яку збирається атакувати.

Віддалені атаки через Internet є ще одним з типів атак через глобальну мережу. Віддалена атака на інформацію може бути здійснена кількома шляхами. По-перше, це атака на систему клієнта з боку серверу. Хакер, у якого є свій WWW сервер, може постаратися за допомогою використання Java-аплетів і Java-скриптів, вбудованих у HTML-документи, вивести з ладу систему користувача або отримати інформацію про неї, яка дозволить йому "зламати" машину користувача. По-друге, хакер через WWW клієнта може спробувати вивести систему користувача або WWW-сервер з ладу чи отримати несанкціонований доступ до інформації. Для цього він може використати "дірки безпеки" в CGI-додатках, погану настройку серверу, спробувати підмінити CGI-додатки. Віддалені атаки через Інтернет можна класифікувати наступним чином:

- Атаки з використанням CGI-додатків і Java-аплетів. Проблема безпеки з CGI (common gateway interface) - програмами полягає в тому, що кожна з них може містити помилку, яка призводить до порушення безпеки серверу. Java-аплети виконуються на клієнтській стороні, а не на сервері, і тому збільшують ризик атаки з боку серверу. У Java вбудовані засоби для обмеження доступу до клієнтської машини. Java-аплетам не дозволяється виконувати системні команди, завантажувати системні бібліотеки або відкривати системні пристрої, такі, як диски. Аплетам дозволяється встановлювати з'єднання по мережі тільки до серверу, звідки аплет був завантажений.

- Атаки типу Nuke - атаки на порти комп’ютера, які призводять до розриву з’єднання і зависання. Може з’явитися синій екран, якщо скинути "блакитні бомби". Словом Nuke (н’юк) називають сьогодні будь-яку атаку на порти віддалених комп’ютерів, що спричинює крах операційної системи або розрив з’єднання з Internet. Nuke не порушує стан файлової системи і не руйнує "залізо" комп’ютера. Перезавантажившись, користувач може продовжити роботу в мережі.
- Атаки типу UDP (User Datagram Protocol). UDP - "ненадійний" протокол, бо не гарантує, що датаграми будуть приходити за тим порядком, як були надіслані, і навіть того, що вони надійдуть взагалі. Якщо надійність - бажана умова, то для її реалізації потрібне програмне забезпечення.
- Атаки типу ICMP (Internet Control Message Protocol, протокол управляючих повідомлень Internet): дозволяє IP-маршрутизаторам посилати повідомлення про помилки і управляючу інформацію іншим IP-маршрутизаторам і головним комп’ютерам мережі. ICMP-повідомлення "подорожують" у вигляді полів даних IP-дейтаграм і обов’язково мають реалізовуватися в усіх варіантах IP. Одним з варіантів такої атаки є Spam, інакше UCE (Unsolicited Commercial Email) – це електронна пошта, що містить рекламні пропозиції, які розсилаються відразу на велику кількість email-адрес.
- Атаки на DNS. Будучи одним з основних елементів інфраструктури IP-мереж, служба доменних імен (DNS) у той же час далеко не ідеальна з точки зору інформаційної безпеки. Застосування транспортного протоколу без встановлення віртуального каналу (UDP), відсутність засобів ідентифікації, аутентифікації і розмежування доступу роблять її вразливою для віддалених атак різних типів. Можливість атаки на DNS шляхом фальсифікації відповіді DNS-серверу відома досить давно. Об’єктом атаки може бути як резолвер (клієнтська частина DNS), так і DNS-сервер. Причини успіху таких атак криються в легкості підробки відповіді серверу. Сучасні протоколи DNS, що використовуються, не передбачають інших засобів перевірки автентичності отриманих даних і їх джерела, повністю покладаючись у цьому на протоколи транспортного рівня. Транспортний протокол (UDP), що використовується з метою підвищення ефективності, не передбачає встановлення віртуального каналу і використовує як ідентифікатор джерела повідомлення IP-адреси, яка може бути елементарно підробленою.

7.3. Використання програмних закладок для зламу автоматизованих систем

Зловмисники-хакери винаходять усе нові й нові атаки на різні елементи підсистем захисту автоматизованих систем. Однією з найбільш небезпечних є атака захищеної системи за допомогою програмних закладок. Програмна закладка - це програма чи фрагмент програми, яка таємно впроваджується в захищену систему, що дозволяє зловмиснику здійснювати НСД до тих чи інших ресурсів захищеної системи. Програмною закладкою ще називають програму, спеціально розроблену для самостійного виконання несанкціонованих дій. При цьому під програмою

розуміється будь-яка послідовність команд, що підлягають виконанню процесором або іншою програмою. Наприклад, макроси, що включаються у файли документів редактора Microsoft Word, також, по суті, є програмами, тому що являють собою послідовності команд, які виконуються редактором для автоматизації дій користувача. Звідси стає зрозуміло, що для розробки програмних закладок може використовуватися вся безліч способів, накопичена в області комп'ютерної вірусології.

Основна небезпека програмних закладок полягає в тому, що така програма, будучи частиною захищеної системи, здатна сама себе активно маскувати в системі. При впровадженні закладки в захищену систему, створюється прихований канал інформаційного обміну, котрий, як правило, залишається непоміченим для адміністраторів системи протягом тривалого часу. Практично всі відомі програмні закладки, що застосовувалися в різний час, були виявлені або через помилки, допущені зловмисниками при їх програмуванні, або випадково.

Першим і одним з найбільш важливих етапів життєвого циклу програмної закладки після її розробки є етап впровадження в комп'ютерну систему, який ще називають інфікуванням. Інфікування комп'ютера можливе тільки при запуску на виконання зараженої (інфікованої) програми. Як правило, копія закладки може вставлятися в інфіковану програму таким чином, що при запуску на виконання зараженої програми вона починає виконувати свої функції. Ознакою інфікування комп'ютерної системи є зараження будь-якої її програми. Як правило, програмна закладка не здатна до самостійного поширення.

Впровадження програмної закладки в комп'ютерну систему може виконуватися за допомогою заражених програм будь-яких типів за вірусною технологією, а також за допомогою апаратних засобів. При впровадженні закладки за вірусною технологією вона обов'язково має бути здатною до самостійного поширення, що властиво звичайному вірусу.

Впровадження програмної закладки за допомогою апаратних засобів припускає зараження програм, що містяться в апаратних пристроях, наприклад, програмах мікросхеми BIOS.

Широкі можливості для впровадження програмних закладок надає використання сучасних автоматизованих систем на основі Internet-технологій. Адже програми навігації, які виконуються на робочій станції, можуть здійснювати переходи до інших ресурсів, активізувати програми на сервері, а також інтерпретувати і запускати на виконання програми, які належать до Web-документа і передаються разом з цим документом із серверу. Такий вид розподіленої обробки дозволив сконцентрувати всю прикладну систему на сервері. Однак можливість виконання на робочих станціях програм із серверу породжує ефективні способи впровадження програмних закладок. Впровадження може бути реалізоване як підміною переданої із сервера програми, так і розміщенням на сервері мобільної програми-закладки.

Щоб програмна закладка почала виконувати свої функції, вона має отримати управління, тобто почати виконувати команди, що належать до цієї закладки.

Закладка активізується при настанні запрограмованих у ній зовнішніх умов. Аналіз зовнішніх умов досягається шляхом обробки закладкою загальних операцій, якими найчастіше можуть бути переривання певної події, а саме: переривання від таймеру; від зовнішніх пристроїв; від клавіатури; при роботі з диском, у тому числі переривання при роботі з файлами. Така програмна закладка має бути розроблена як макрос, що автоматично виконується при певних подіях - відкритті, закритті документів, запуску, завершенні роботи програми оболонки тощо.

Як висновок, необхідно підкреслити, що при розробці політики безпеки і захисту комп'ютерної системи від програмних закладок необхідно враховувати, що зловмисник, швидше за все, буде використовувати недокументовані можливості операційної системи. З цієї причини, доцільно організовувати виявлення уразливих для програмних закладок місць системи захисту за допомогою груп експертів. При використанні такого підходу можуть бути виявлені слабкі місця захисту і розроблені рекомендації щодо адекватної політики безпеки з урахуванням загроз.

СПИСОК ЛІТЕРАТУРИ

1. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах». Відомості Верховної Ради (ВВР), 1994, N 31, ст.286.
2. Загальні положення щодо захисту інформації в комп'ютерних системах від несанкціонованого доступу. – НД ТЗІ 1.1-001-98, ДСТТСЗІ СБ України, Київ, 1998.
3. Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. – НД ТЗІ 2.2-001-98, ДСТТСЗІ СБ України, Київ, 1998.
4. Казарин О.В. Безопасность программного обеспечения компьютерных систем. Монография. – М.: МГУЛ, 2003. – 212 с.
5. Белкин П.Ю. Новое поколение вирусов принципы работы и методы защиты// Защита информации. - 1997.- №2.-С.35-40.
6. Зегжда Д.П., Шмаков Э.М. Проблема анализа безопасности программного обеспечения// Безопасность информационных технологий. - 1995.- №2.- С.28-33.
7. Зима В.М., Молдовян А.А., Молдовян Н.А. Защита компьютерных ресурсов от несанкционированных действий пользователей. - Учеб пособие. - СПб: Издательство ВИКА им. А.Ф. Можайского, 1997.
8. Проблемы безопасности программного обеспечения. Под ред. П.Д. Зегжда. - СПб.: Издательство СПбГТУ, 1995.
9. Правиков Д.И., Чибисов В.Н. Об одном подходе к поиску программных закладок// Безопасность информационных технологий. - 1995.- №1.- С.76-79.
10. Скляров Д.В. Искусство защиты и взлома информации. – СПб.: БХВ-Петербург, 2004. – 288 с.: ил.
11. Терехов А.В., Чернышов В.Н., Селезнев А.В., Рак И.П. Защита компьютерной информации: Учебное пособие. Тамбов: Изд-во Тамб. гос. техн. ун-та, 2003. 80 с.
12. Малюк А.А., Пазизин С.В., Погожин Н.С. Введение в защиту информации в автоматизированных системах. М.: Горячая линия-Телеком, 2001. 148 с.
13. Казарин О.В. Теория и практика защиты программ. – М, 2004. – 450 с.
14. Коркішко Т., Мельник А. Стан та напрямки розвитку надвеликих інтегрованих схем захисту інформації // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. – Київ, 2000. – С. 275 - 281.
15. Класифікація автоматизованих систем і стандартні профілі захищеності оброблюваної інформації від несанкціонованого доступу. – НД ТЗІ 2.2-002-98, ДСТТСЗІ СБ України, Київ, 1998.
16. Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу. – НД ТЗІ 1.1-002-98, ДСТТСЗІ СБ України, Київ, 1998.
17. Методи захисту банківської інформації: Навчальний посібник / В.К.Задірака, О.С. Олексюк, М.О.Недашковський. – К.: Вища шк., 1999, - 261 с.

Підписано до друку 20.01.2007 р.
Формат 84х108\32. Папір офсетний. Друк на різографі.
Умов.-друк. арк. 3,5. Зам. №2432.
Тираж 100 прим.