МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до лабораторних робіт з дисципліни

”Безпека програмного забезпечення комп’ютерних систем”

для здобувачів освітнього ступеня бакалавр спеціальності № 123 «Комп’ютерна інженерія»
(напрям підготовки 6.050102 «Комп’ютерна інженерія»)
усіх форм навчання

Затверджено на засіданні кафедри
інформаційної безпеки та
комп’ютерної інженерії,
протокол № 8 від 27.03.2018 р.

Черкаси 2018

УДК 004.056(07)
М 54

Упорядники: Бабенко Віра Григорівна, к.т.н., доцент
Миронюк Тетяна Василівна, к.т.н., доцент
Стабецька Тетяна Анатоліївна, асистент

Рецензент: Дядюшенко О.О., к.т.н., доц., доцент кафедри пожежно-профілактичної роботи Черкаського інституту пожежної безпеки ім. героїв Чорнобиля НУЦЗ України.

Методичні рекомендації до лабораторних робіт з дисципліни М54 «Безпека програмного забезпечення комп'ютерних систем» для здобувачів освітнього ступеня бакалавра спеціальності № 123 «Комп'ютерна інженерія» (напрям підготовки 6.050102 «Комп'ютерна інженерія») усіх форм навчання [Електронний ресурс] / [Упоряд.: В.Г.Бабенко, Т.В. Миронюк, Т.А. Стабецька]; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси : ЧДТУ, 2018. – 42 с.

Методичні рекомендації містять основні теоретичні відомості та практичні завдання до проведення лабораторних робіт з дисципліни «Безпека програмного забезпечення комп'ютерних систем» з використанням сучасного апаратного і програмного забезпечення. Особлива увага приділяється формуванню навичок застосування методів та засобів забезпечення технологічної та експлуатаційної безпеки програм, зокрема захисту програм від впливу руйнівних програмних засобів.

Для здобувачів освітнього ступеня бакалавр спеціальності № 123 «Комп'ютерна інженерія» (напрям підготовки 6.050102 «Комп'ютерна інженерія») усіх форм навчання.

УДК 004.056(07)

В авторській редакції.

Методичне
електронне видання
мережного використання

ЗМІСТ

ВСТУП

Дані методичні рекомендації призначені для проведення лабораторних робіт з дисципліни "Безпека програмного забезпечення комп'ютерних систем". Лабораторні роботи впроваджені в навчальний процес на підставі навчальних планів з метою навчання проектуванню, написанню та налагодженню програм системного рівня, вироблення вміння класифікувати загрози програмному забезпеченню та оцінювати його вразливість, а також вміння обирати та застосовувати методи та засоби захисту інформації від несанкціонованого доступу, підміни та нав'язування хибних даних у комп'ютерних системах та мережах. Методичні рекомендації містять опис 7 лабораторних робіт. Кожен розділ, що відповідає окремій лабораторній роботі, складається з таких підрозділів: мета роботи, вказівки щодо виконання лабораторної роботи, огляд теми роботи, варіанти завдань на лабораторну роботу, контрольні запитання та завдання.

Порядок виконання роботи

При проведенні лабораторних робіт студент повинен показати творчий підхід до розробки модулів програмного забезпечення, грамотне використання існуючого програмного забезпечення, гарне алгоритмічне мислення, навички висококваліфікованого програмування на алгоритмічних мовах низького рівня. Студенти повинні вміти перетворити свою програму в програмний продукт, виконати якісний аналіз програми, зробити оцінку отриманих результатів при використанні різних варіантів. Завдання передбачають створення унікальної програми, яка написана з використанням процедур та функцій операційної системи. Особливу увагу при розробці необхідно приділити ефективності використання ресурсів обчислювальної системи. Важливе значення має зручний інтерфейс з користувачем і грамотна документація до програми, призначена для користувачів і супроводу програми.

Виконання роботи включає наступні етапи:

1. Перед кожною лабораторною роботою студенти повинні виконати самостійну підготовку:
   - засвоїти теоретичні відомості до роботи;
   - незрозумілі питання з'ясувати з викладачем та за допомогою додаткової літератури;
   - дати відповіді на контрольні питання;
2. Підготовчий етап (до проведення лабораторної роботи):
   а) одержання завдання згідно даної методичної вказівки, номер варіанту і вимог викладача;
   б) вивчення теоретичного матеріалу по темі лабораторної роботи;
   в) розробка алгоритму і примірника програми;
3. Безпосереднє створення програми в комп'ютерному класі:

а) проходження допуску до лабораторної роботи (обговорення з викладачем запропонованого рішення задачі);

б) написання програми з використанням Microsoft Visual C + + або Borland C++;

в) налагодження програми;

г) перевірка правильності виконання завдання (тестування).

4. Виконання звіту та захист лабораторної роботи.

Зміст звіту

Звіт про виконання лабораторної роботи має складатися з таких обов'язкових частин:

- титульний лист з найменуванням лабораторної роботи і даними виконавця;
- тема роботи;
- мета роботи;
- завдання згідно варіанту;
- стислі теоретичні відомості за темою роботи (в потрібній кількості для захисту звіту);
- опис алгоритму вирішення поставленого завдання;
- текст програми з коментарями і поясненнями;
- результати роботи програми та їх аналіз;
- висновки;
- список використаних джерел.

Оцінювання виконання лабораторної роботи

Згідно кредитно-модульної організації навчального процесу кожна лабораторна робота оцінюється відповідною кількістю балів за такими критеріями:

– Кількість балів – 4. Робота виконана відповідно до завдання та вищевикладених вимог, виконана самостійно, має оригінальні технічні рішення. Студент дає чіткі відповіді на поставлені запитання.

– Кількість балів – 3. Робота виконана відповідно до завдання та вищевикладених вимог, виконана самостійно. Студент дає неповні відповіді на поставлені запитання.

– Кількість балів – 2. Робота, що має кілька недоліків в реалізації, деякі прогалини в опрацюванні окремих питань. Студент дає неповні відповіді на запитання.

– Кількість балів – 1. Робота, що має суттєві недоліки в реалізації, слабке опрацювання ключових питань, недостатньо аргументовані відповіді на запитання.

– Кількість балів – 0. Робота, що має досить суттєві недоліки в реалізації. Студент не опрацював ключових питань за даною темою та не може дати аргументовані відповіді на запитання.

Тобто, кожна лабораторна робота оцінюється від 0 до 4 балів. Згідно методичних вказівок з дисципліни передбачено 7 лабораторних робіт. Загалом максимальна кількість балів, яку студент може набрати, виконавши лабораторні роботи – 28 балів.

Захист лабораторної роботи

1. Викладачу здається програма та початковий код в електронному вигляді і роздрукований звіт.
2. Захистити звіт. Звіт оцінюється викладачем згідно кредитно-модульної системи.
3. Студент, який не захистив попередню лабораторну роботу, до виконання наступної лабораторної роботи не допускається.

ЛАБОРАТОРНА РОБОТА №1
СИМЕТРИЧНІ АЛГОРИТМИ ШИФРУВАННЯ ДАНИХ

Мета роботи: отримати навички використання симетричних криптографічних алгоритмів для шифрування даних.

Короткі теоретичні відомості

Існує два основні типи криптографічних алгоритмів:

- симетричні, для яких ключ розшифрування співпадає з ключем шифрування;
- асиметричні (алгоритми з відкритим ключем), які використовують для шифрування і розшифрування два різні ключі.

Симетричні алгоритми діляться на дві категорії:

- потокові шифри, в яких дані обробляються побітно (посимвольно);
- блокові шифри, в яких операції виконуються над групами бітів.

Криптостійкість – характеристика шифру, визначальна його стійкість до дешифрування без знання ключа.

Основні характеристики:

- кількість усіх можливих ключів;
- середній час, необхідний для криптоаналізу.

Загальноприйняті вимоги до криптографічних алгоритмів:

1. Зашифрований текст читається тільки за наявності ключа;
2. Кількість операцій для знаходження ключа по фрагменту шифрованого тексту і відкритого тексту, що відповідає йому, – не менше загального числа можливих ключів;
3. Кількість операцій для дешифрування шляхом перебору всіляких ключів, повинно мати строгу нижню оцінку і виходити за межі можливостей комп'ютерів;
4. Знання алгоритму шифрування не повинне впливати на надійність захисту;
5. Незначна зміна ключа повинна призводити до істотної зміни вигляду зашифрованого повідомлення, навіть, при використанні одного і того ж ключа;
6. Структурні елементи алгоритму шифрування мають бути незмінними;
7. Додаткові біти, що вводяться в повідомлення при шифруванні мають бути повністю і надійно приховані у шифрованому тексті;
8. Довжина шифрованого тексту має дорівнювати довжині початкового тексту;
9. Не повинно бути простих і легковстановлюваних залежностей між ключами, які послідовно використовуються при шифруванні;
10. Будь-який ключ із множини всеможливих повинен забезпечувати надійний захист інформації;
11. Алгоритм повинен допускати як програмну, так і апаратну реалізацію, при цьому зміна довжини ключа не повинна призводити до якісного погіршення алгоритму шифрування.

Серед методів криптографічного закриття можна виділити наступні:

- Заміна(підстановка);
- Перестановка;
- Аналітичне перетворення;
- Гамування;
- Комбіновані методи.

Гамування – накладення на текст псевдовипадкової послідовності згенерованої на основі ключа. Можливі наступні різновиди гам:

- кінцева коротка гама;
- кінцева довга гама;
- нескінченна гама.

Шифрування методом гамування полягає в заміні символів шифрованого тексту і гами цифровими еквівалентами (чи у вигляді двійкового коду).

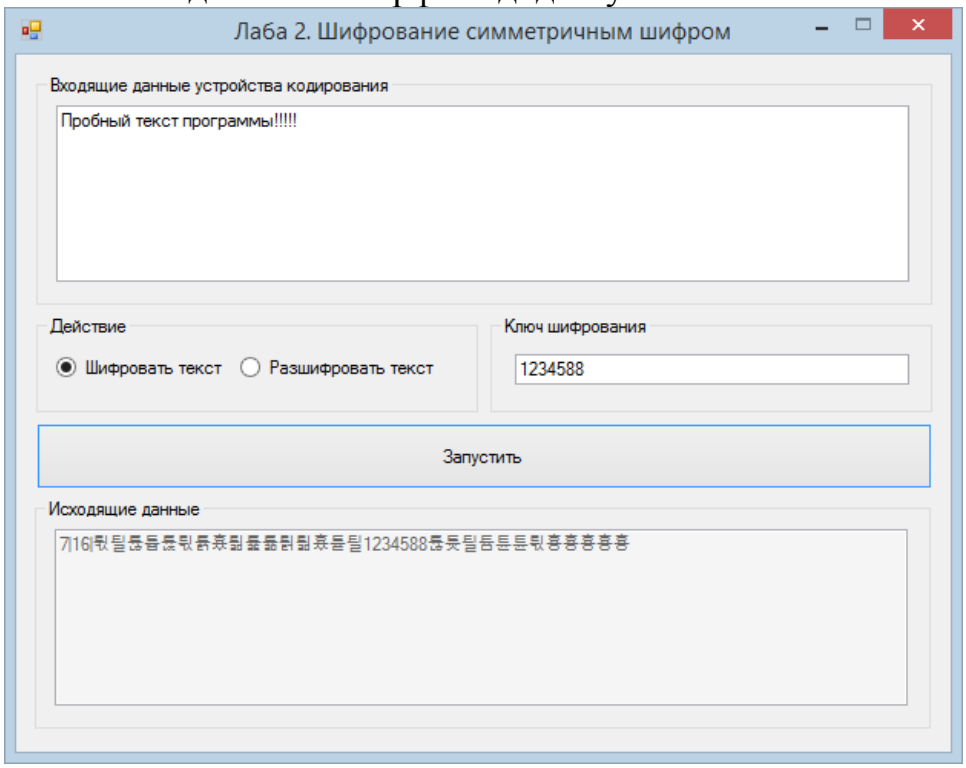
Стійкість шифрування визначається довжиною періоду і рівномірністю статистичних характеристик гами.

Постановка завдання

Завдання лабораторної роботи – розробити програмний додаток, в якому текст, що вводиться, шифрується симетричним алгоритмом за допомогою ключа, що задається.

Рекомендації при розробці додатку

1. При розробці додатку використайте стандартні візуальні компоненти: TMemo, TEdit, TButton, TRadioGroup, TMainMenu, TOpenDialog, TSaveDialog.
2. Рекомендований інтерфейс додатку :



3. Приклад виконання лабораторної роботи:

```
using System;
using System.Drawing;
using System.IO;
using System.Windows.Forms;
```

```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private string xor_en(string message, int key, string result = null)
        {
            for (int i = 0; i < message.Length; i++) result += (char)(message[i] ^ key);
            return result;
        }

        private void textBox1_Click(object sender, EventArgs e)
        {
            if (textBox1.ForeColor != Color.Black)
            {
                textBox1.Text = null;
                textBox1.ForeColor = Color.Black; } }

        private void button1_Click(object sender, EventArgs e)
        {
            int key_length = textBox2.Text.Length, pos = 0;
            string file;

            if (radioButton1.Checked != false)
            {
                Clipboard.SetText(textBox3.Text = xor_en(textBox1.Text,
Convert.ToInt32(textBox2.Text)));
                pos = new Random().Next(1, textBox3.Text.Length);
                file = key_length.ToString() + "|" + pos + "|";
                textBox3.Text = textBox3.Text.Insert(pos, textBox2.Text);
                file += textBox3.Text;
                textBox3.Text = file;
                StreamWriter write = new StreamWriter(new FileStream("text.sts",
FileMode.Create, FileAccess.Write));
                write.Write(file);
                write.Close();
            }
            else Clipboard.SetText(textBox3.Text = xor_en(textBox1.Text,
Convert.ToInt32(textBox2.Text)));

        }

        private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!Char.IsDigit(e.KeyChar) && e.KeyChar != Convert.ToChar(8)) e.Handled =
true;
        }

        private void radioButton3_CheckedChanged(object sender, EventArgs e)
        {
            if (radioButton3.Checked != false)
            {
                OpenFileDialog stas = new OpenFileDialog();
                if (stas.ShowDialog() == DialogResult.OK)
                {
                    StreamReader read = new StreamReader(new FileStream(stas.FileName,
FileMode.Open, FileAccess.Read));
                    string a3 = read.ReadLine();
                    string[] split = a3.Split(new Char[] { '|' });
```

```
                    int a1 = Convert.ToInt32(split[0]), a2 = Convert.ToInt32(split[1]);
                    a3 = a3.Remove(0, a1.ToString().Length + a2.ToString().Length + 2);
                    textBox2.Text = a3.Substring(a2, a1);
                    textBox1.Text = a3.Remove(a2, a1);
                    textBox3.Text = "Здесь будет выведен результат....";
                }
            }
        }
    }
}
```

У наведеному лістингу в програмі виконується переведення символів тексту і ключа в цифровий (тип Byte) аналог таблиці ASCII, організовується цикл перебору елементів масиву тексту з накладенням на них відповідних елементів масиву ключа. Ключ шифрування перед записом у файл перетворюється за допомогою алгоритму шифрування XOR.

4. Розробіть додаток, в якому зашифрований текст зберігається у файлі текстового формату разом з ключем. Перед записом у файл зашифрованого тексту і ключа заздалегідь перемішайте записувані масиви (в отримуваному масиві виділяються фіксовані позиції для зберігання символів ключа; для коректного складання ключа необхідно вказати довжину ключа в символах і крок фіксованих позицій, тобто через скільки символів шифрованого тексту знаходяться символи ключа; вказані ознаки можна записати в якості перших двох байтів файлу або ж в інших позиціях файлу).

Розшифровка отримуваного файлу складається з наступних кроків:

- зчитування характеристик ключа;
- складання ключа;
- розшифровки тексту.

5. Звіт оформляється у формі Zip- архіву, проекту, що містить усі файли (Project1.cfg, Project1.dof, Project1.dpr, Project1.exe, Project1.res, Unit1.dcu, Unit1.dfm Unit1.pas). Zip- архів іменується по прізвищу студента, номера групи і назви лабораторної роботи.

Контрольні запитання

1. Чому криптографічні алгоритми, що вимагають збереження в таємниці послідовності перетворення даних, не знаходять нині широкого застосування?
2. Яким має бути об'єм ключового простору для забезпечення криптографічної стійкості алгоритму?
3. Чи залежить криптографічна стійкість алгоритму від набору можливих символів ключа?
4. Що таке ключ?
5. Чи можна використати послідовності цифр фундаментальних констант при формуванні гами?
6. Назвіть основні показники криптостійкості.
7. Охарактеризуйте заходи по захисту ключів.
8. Виходячи з чого визначається необхідність зміни ключів шифрування?

ЛАБОРАТОРНА РОБОТА №2

ЗАХИСТ ПРОГРАМ ВІД НЕСАНКЦІОНОВАНОЇ ЕКСПЛУАТАЦІЇ ЗА РАХУНОК ПРИВ'ЯЗКИ ДО НОСІЯ ІНФОРМАЦІЇ

Мета роботи: отримати навички по установці захисту на програми, що розробляються, за рахунок прив'язки до носія інформації (облаштування зовнішньої пам'яті).

Короткі теоретичні відомості

Становлення ринкових відносин у суспільстві змушує виробників захищати свій продукт від незаконного його використання. Особливо актуальною ця проблема постає в області інформаційних технологій. Як показує практика, абсолютних способів захисту інформації не існує. Якими б складними і дорогими не були запропоновані на ринку засоби захисту, їх ефективність виявляється умовною. З урахуванням реальної обстановки, що склалася, вкрай необхідними стають нескладні та недорогі засоби захисту, що розробляються і встановлюються самим виробником продукту та спрямовані проти незаконних дій кваліфікованих користувачів.

Ідея захисту ґрунтується на використанні індивідуальних характеристик носіїв інформації. При запуску додатка проводиться перевірка на наявність підключеного зовнішнього пристрою з конкретним серійним номером і на знаходження програми, що стартувала, на цьому пристрої. Даний спосіб захисту дозволяє безперешкодно копіювати додаток, при цьому існуючі стандарти запису інформації не порушуються, специфічних вимог до облаштувань прочитування-запису немає. Можлива організація перевірки з різних точок програми з використанням декількох подібних процедур: мета цих дій – ускладнити роботу кваліфікованого зламувача.

Середовище прискореної розробки додатків Visual Studio дозволяє безпосередньо працювати з функціями API - Windows.

Рекомендації по проектуванню захисту програм:

- не використовуйте стандартні процеси обробки компонентів, а організовуйте перевірки в циклі повідомлень;
- не зберігайте коди в одному місці;
- не перевіряйте код тільки в одному місці;
- не аналізуйте характеристику відразу після її отримання (прочитування);
- не створюйте для перевірки функцію або бібліотеку;
- не задавайте дії, пов'язані з перевіркою, відразу після самої перевірки;
- застосовуйте відволікаючі функції перевірок;
- не зберігайте результати перевірок у змінних;
- не перевіряйте контрольні дані одним алгоритмом;
- не зберігайте результати перевірки в реєстрі;

- застосовуйте шифрування програм і даних;
- не записуйте текстові рядки в програмі в їх реальному вигляді.

Дії, спрямовані проти використання налагоджувачів при зломі програм:

- визначення налагоджувача до запуску програми з подальшим завершенням чи емуляцією помилки;
- зміна роботи програми у разі її виконання у налагоджувачі;
- ускладнення лістингу;
- зашифровані рядки в ресурсах.

Постановка завдання

Завдання лабораторної роботи:

- визначити серійний номер підключеного зовнішнього носія інформації;
- вбудувати в додаток перевірку наявності зовнішнього носія з конкретним серійним номером.

Рекомендації по розробці додатка

1. Можливий варіант інтерфейсу допоміжного застосування наведений на рисунках 1 та 2:

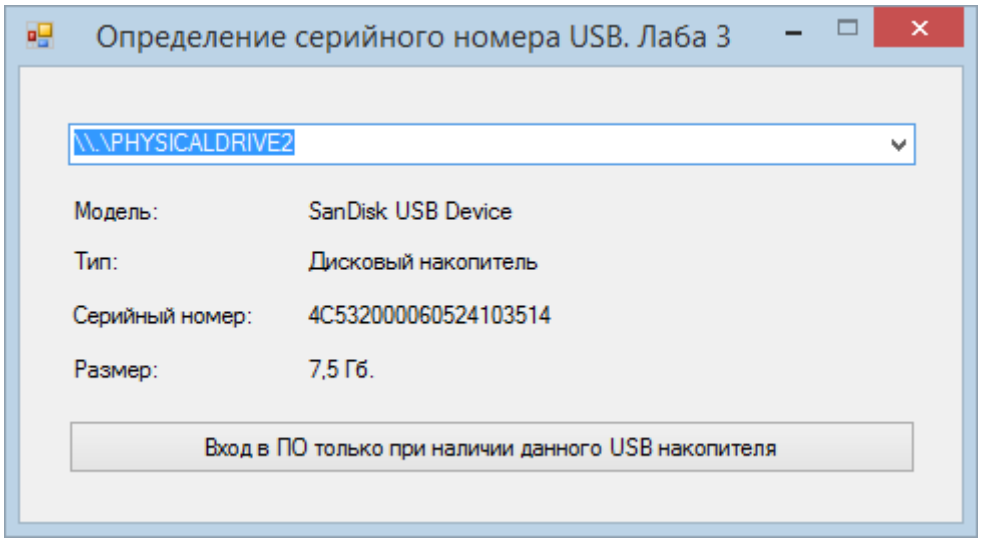


Рисунок 1 – Зовнішній вигляд вікна допоміжного застосування

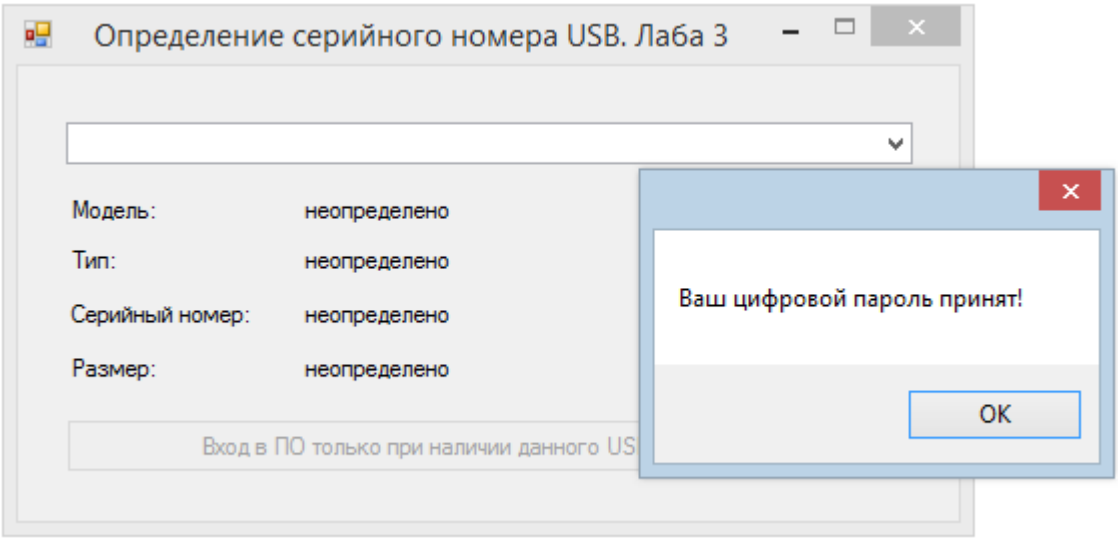


Рисунок 2 – Перевірка, під час завантаження ПЗ

2. Приклад виконання лабораторної роботи:

```
using System;
using System.Diagnostics;
using System.IO;
using System.Management;
using System.Windows.Forms;

namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        private ManagementObjectSearcher theSearcher;
        protected string serial = null;

        public Form1()
        {
            int count = 0;
            InitializeComponent();
            if (System.IO.File.Exists("id.pas"))
            {
                StreamReader read = new StreamReader(new FileStream("id.pas", FileMode.Open,
FileAccess.Read));
                try
                {
                    theSearcher = new ManagementObjectSearcher("SELECT * FROM
Win32_DiskDrive WHERE InterfaceType='USB' AND SerialNumber='"+ read.ReadLine()+"'");
                    foreach (ManagementObject currentObject in theSearcher.Get()) count++;
                    read.Close();
                    if (count != 0) MessageBox.Show("Ваш цифровой пароль принят!");
                    else Process.GetCurrentProcess().Kill();
                }
                catch (InvalidCastException e) { read.Close();
Process.GetCurrentProcess().Kill(); }
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            theSearcher = new ManagementObjectSearcher("SELECT * FROM Win32_DiskDrive WHERE
InterfaceType='USB'");
            foreach (ManagementObject currentObject in theSearcher.Get())
comboBox1.Items.Add(currentObject["DeviceID"].ToString());
```

```
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            button1.Enabled = true;
            foreach (ManagementObject currentObject in theSearcher.Get())
            {
                if (currentObject["DeviceID"].ToString() == comboBox1.Text)
                {
                    ManagementObject theSerialNumberObjectQuery = new
ManagementObject("Win32_PhysicalMedia.Tag='" + comboBox1.Text + "'");
                    label2.Text = currentObject["Model"].ToString();
                    label4.Text = currentObject["Description"].ToString();
                    label5.Text = currentObject["SerialNumber"].ToString();
                    label11.Text = Math.Round(((Convert.ToDouble(currentObject["Size"])) /
1073741824), 1).ToString() + " Гб.";

                    serial = currentObject["SerialNumber"].ToString();
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            StreamWriter write = new StreamWriter(new FileStream("id.pas", FileMode.Create,
FileAccess.Write));
            write.Write(serial);
            write.Close();
            Process.GetCurrentProcess().Kill();
        }
    }
}
```

3. При перевірці серійного номера зовнішнього пристрою реалізуйте вищеперелічені рекомендації по проектуванню захисту програм.

Контрольні запитання

1. Які способи захисту від несанкціонованого копіювання програм Ви знаєте?
2. Охарактеризуйте спосіб захисту, що ґрунтується на використанні міток носія інформації.
3. Охарактеризуйте спосіб захисту, що ґрунтується на фізичних дефектах носія інформації.
4. Охарактеризуйте спосіб захисту, що ґрунтується на тимчасових характеристиках читання носія інформації.
5. Як грамотно з точки зору захисту від злому представляти в початковому тексті програми шаблони для порівняння строкових змінних?
6. Як організувати процедури, що виконують одні і ті ж дії, але що мають різну «операторну начинку»?
7. Чи можна зберігати значення серійного номера пристрою, що перевіряється, не в тілі програми, а в окремому файлі?
8. Чи доцільно виділяти процедури, що здійснюють дії по захисту, в окремі динамічні бібліотеки?
9. Як коректно іменувати процедури, які здійснюють захисні механізми?

ЛАБОРАТОРНА РОБОТА №3

ПРОГРАМУВАННЯ ЗМІН ХАРАКТЕРИСТИК ФАЙЛУ

Мета роботи: отримати навички по маскуванню захисних дій у програмі, що розробляється.

Короткі теоретичні відомості

Розробник програми зацікавлений в якнайповнішому уявленні можливостей своїх розробок для привертання уваги потенційного покупця, в результаті він вимушений ризикувати, пропонуючи на ринок Trial і Demo – версії додатків, що є повнофункціональними.

В якості засобів захисту від несанкціонованої експлуатації програмного забезпечення широко використовуються способи поширення продукту з обмеженими можливостями, такими як:

- обмежений за часом період можливого використання продукту;
- обмежена кількість запусків програми.

При застосуванні типових методик у разі використання вищезгаданих способів захисту, програмісти обмежуються записом контрольних параметрів (лічильник запусків, гранична дата роботи) безпосередньо в реєстр операційної системи. Для кваліфікованого користувача, робота в реєстрі не представляє великих труднощів. Як показала практика згаданий підхід не є ефективним, оскільки місце зберігання контрольних параметрів не є таємницею.

При використанні захисту, що ґрунтується на контролі дати виконання або контролі кількості запусків програми, перспективнішим є зберігання контрольних параметрів в замаскованій формі у файлах вибраних самим розробником і таких, що зберігаються в довільному місці (наприклад серед допоміжних файлів самої програми). При подібному підході актуальним стає маскування звернень до файлів, в яких зберігаються поточні значення контрольних параметрів. Загальновідомо, що при будь-якій зміні вмісту файлу операційна система автоматично коригує зовнішні характеристики файлу, такі, як довжина, час і дата створення, які зберігаються у файловій системі.

Постановка завдання

Організувати захист, який полягає в обмеженні кількості запусків програми, причому значення лічильника запусків зберігати у файлі, зовнішні характеристики якого (дата і час створення) залишаються постійними.

Рекомендації при розробці додатку:

1. Розробіть тестову програму в якій у режимі діалогу можна змінити дату створення обраного файлу.
2. Рекомендований інтерфейс тестової програми приведено на рисунках 1 та 2:

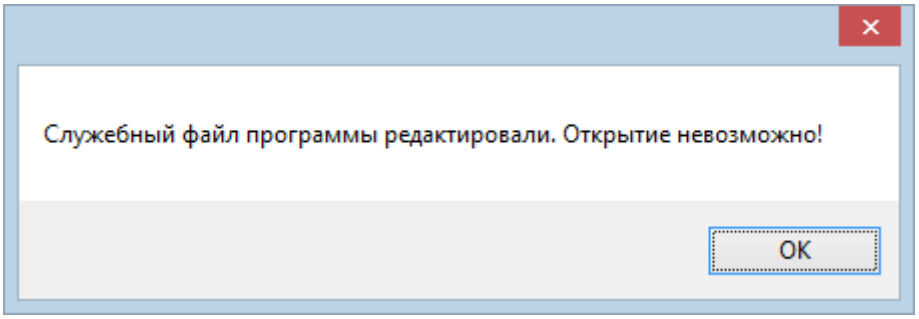


Рисунок 1 – Захист зміни службових параметрів файлу

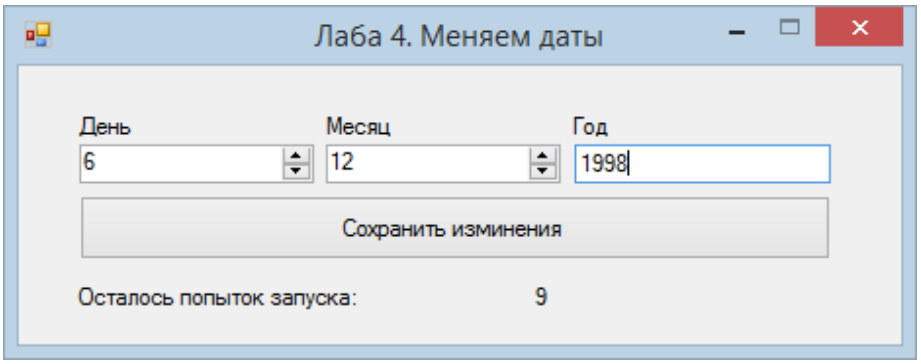


Рисунок 2 – Редагування дати створення та редагування вибраного файлу

3. Приклад виконання лабораторної роботи:

```
using System;
using System.Diagnostics;
using System.IO;
using System.Windows.Forms;

namespace WindowsFormsApplication3
{
    public partial class Form1 : Form
    {
        private string path = null, file = null;
        private int count = 9;
        private Random rd = new Random();
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            OpenFileDialog st = new OpenFileDialog();
            if (st.ShowDialog() == DialogResult.OK)
            {
                path = st.FileName;
            }
            else Process.GetCurrentProcess().Kill();

            if (!System.IO.File.Exists("id.pas"))
            {
                StreamWriter write = new StreamWriter(new FileStream("id.pas",
FileMode.Create, FileAccess.Write));
                for (int i = 0; i <= 99; i++) file += rd.Next(0, 10).ToString();
                file = file.Insert(8, count.ToString());
                write.Write(file);
                write.Close();
                System.IO.File.SetLastWriteTime("id.pas", new DateTime(1992, 10, 22));
```

```
            }
            else
            {
                if (System.IO.File.GetLastWriteTime("id.pas") == new DateTime(1992, 10, 22))
                {
                    StreamReader read = new StreamReader(new FileStream("id.pas",
FileMode.Open, FileAccess.Read));
                    count = Convert.ToInt32(read.ReadLine().Substring(8, 1)) - 1;
                    read.Close();
                    StreamWriter write = new StreamWriter(new FileStream("id.pas",
FileMode.Create, FileAccess.Write));
                    for (int i = 0; i <= 99; i++) file += rd.Next(0, 10).ToString();
                    file = file.Insert(8, count.ToString());
                    write.Write(file);
                    write.Close();
                    System.IO.File.SetLastWriteTime("id.pas", new DateTime(1992, 10, 22));
                }
                else
                {
                    MessageBox.Show("Служебный файл программы редактировали. Открытие
невозможно!");
                    Process.GetCurrentProcess().Kill();
                }
            }
            label5.Text = count.ToString();

            if (count <= 0)
            {
                MessageBox.Show("Пробный период окончен. Запуск ПО невозможен");
                Process.GetCurrentProcess().Kill();
            }
        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs e)
        {
            numericUpDown1.Value = numericUpDown1.Value >= 31 ? 31 : numericUpDown1.Value <=
1 ? 1 : numericUpDown1.Value;
        }

        private void numericUpDown2_ValueChanged(object sender, EventArgs e)
        {
            numericUpDown2.Value = numericUpDown2.Value >= 12 ? 12 : numericUpDown2.Value <=
1 ? 1 : numericUpDown2.Value;
        }

        private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!Char.IsDigit(e.KeyChar) && e.KeyChar != Convert.ToChar(8)) e.Handled =
true;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            System.IO.File.SetLastWriteTime(@path, new
DateTime(Convert.ToInt32(textBox1.Text), (int)numericUpDown2.Value,
(int)numericUpDown1.Value));
            System.IO.File.SetCreationTime(@path, new
DateTime(Convert.ToInt32(textBox1.Text), (int)numericUpDown2.Value,
(int)numericUpDown1.Value));
            MessageBox.Show("Даты сохранены.");
        }
    }
}
```

4. Для маскування дій використовуйте відволікаючі функції (наприклад: організація порожнього циклу з невиконуючими переходами на зайві мітки).

Контрольні запитання

1. Які способи розповсюдження програмних продуктів ви знаєте?
2. Файлам якого формату віддають перевагу для зберігання лічильника запусків програми?
3. У якому вигляді краще зберігати дату та час?
4. Як організувати в тілі програми додаткові перевірки?
5. Що розуміють під незадокументованими точками входу в програму?
6. Сформулюйте рекомендації для забезпечення тестових перевірок роботи додатку у випадку використання захисту, який базується на використанні лічильника запусків програми.
7. Як раціонально організувати ведення протоколу при захисті, що базується на підрахунку кількості запусків програми?
8. Як протистояти використанню налагоджувачів при спробі злому програми?

ЛАБОРАТОРНА РОБОТА №4

ЗАХИСТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВІД НЕСАНКЦІОНОВАНОГО ВИКОРИСТАННЯ ТА КОПІЮВАННЯ

Мета роботи: отримати навички захисту програмного забезпечення від несанкціонованого використання та копіювання.

Короткі теоретичні відомості

На сьогоднішній день, захист програмного забезпечення від несанкціонованого копіювання та використання є актуальною задачею у зв'язку зі збереженням комерційних і авторських прав фірм та розробників. За висновками закордонних фахівців, економічний збиток від "піратського" копіювання програмного забезпечення складає мільярди доларів. Точні втрати установити неможливо через відсутність повних відомостей про число "піратських" копій, але вважається, що з кожної програми їх робиться від 2 до 15.

Захист від несанкціонованого копіювання та використання – це створення засобів, що здійснюють можливість захисту від несанкціонованого виконання. На сучасному етапі найбільшого розповсюдження набули такі технології захисту від копіювання:

1. Створення особливо обумовлених носіїв – найбільш розповсюджена технологія захисту від НСК. Вона передбачає створення на флешці, дискеті або диску спеціально організованої мітки, яка використовується як ознака її дистрибутивності. Спеціальна частина програми, яка захищається виконує функцію контролю мітки. Після того, як відбулося копіювання засобами ОС диска, який захищається, буде скопійована вся інформація, крім мітки. Контролююча частина програми, під час її виконання встановить, що диск не є дистрибутивним, і припинить виконання програми. Цим самим програма прив’язується до певного носія. Для створення мітки застосовуються програмні та апаратні засоби, також є можливість їхнього комбінування;
2. Збереження інформації в закодованому, зашифрованому вигляді є ще одним способом запобігання незаконного використання програм і даних;
3. Використання ключів, які підключаються до COM, LPT чи USB портів.

Виходячи з даних технологій системи захисту від несанкціонованого копіювання можна розділити на такі групи:

- прив’язка до дистрибутивного носія (дискети, CD або DVD-диску);
- прив’язка до комп’ютера;
- прив’язка до ключа;
- опитування довідників;
- обмеження на використання ПЗ.

Особливої уваги потребує захист програмного забезпечення від несанкціонованого копіювання з прив’язкою до комп’ютера, тому що цей тип захисту найпоширеніший.

Розрізняють такі методи прив’язки до комп’ютера:

- прив’язка до вінчестера;
- прив’язка до BIOS;
- прив’язка до архітектури, набору ПЗ і ін.

Коротко розглянемо особливості кожної з прив’язки.

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

Існують такі види прив’язки до вінчестера:

- прив’язка до розташування на вінчестері;
- записування контрольної інформації на ділянку, що не використовується;
- використання фізичних дефектів вінчестера;
- використання серійного номера вінчестера.

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

Прив’язка до програмного забезпечення означає:

- прив’язку до версії операційної системи. Оскільки версія операційної системи, з якою працює користувач на даному комп'ютері, не міняється протягом досить тривалого часу, то її також можна використовувати як параметр для організації перевірки. Дану перевірку бажано проводити в комплексі з іншими методами захисту, наприклад, додатково до перевірки дати створення BIOS, що ще більш звужує кількість машин, на яких може працювати програма, що захищається;
- перевірку ключа, який знаходиться в системному реєстрі;
- перевірку наявності драйверів, програм на вінчестері або в оперативному запам’ятовуючому пристрої;
- перевірку вмісту файлів “autoexec.bat”, “config.sys” на наявність певної послідовності запусків програм і команд DOS;
- модифікацію деяких системних файлів, а потім їх перевірку.

З наведених вище методів прив’язки до комп’ютера та програмного забезпечення, зокрема операційної системи, найкращим та найефективнішим є метод прив’язки до системного реєстру. Оскільки на сучасному етапі розвитку комп’ютерної техніки досить часто відбувається несанкціоновані дії, які несуть шкоду програмному забезпеченню та комп’ютеру взагалі, а даний метод найкраще захищає від даних проблем.

Постановка завдання

1. Для програми, розробленої при виконанні лабораторних робіт №1, №2 і №3, написати програму-інсталятор, яка:

• запитує у користувача папку для установки програми для захисту,

• записує туди файл з виконуваним кодом програми,

• збирає інформацію про комп'ютер, на якому встановлюється програма,

• хешує цю інформацію,

• підписує її особистим ключем користувача програми і записує підпис до реєстру Windows в розділ HKEY_CURRENT_USER \ Software \ Прізвище_студента як значення параметра Signature.

2. У саму програму для захисту включити фрагмент, в якому:

• збирається інформація про комп'ютер, на якому запускається програма,

• обчислюється хеш-значення цієї інформації,

• зчитується підпис із зазначеного вище розділу реєстру, який перевіряється за допомогою відкритого ключа користувача.

3. При невдалій перевірці робота програми для захисту повинна завершуватися з видачею відповідного повідомлення.

4. Зібрана про комп'ютер інформація, включає в себе:

• Ім'я користувача,

• ім'я комп'ютера,

• шлях до папки з ОС Windows,

• шлях до папки з системними файлами ОС Windows,

• а також дані, обрані відповідно до виданого завдання (див. табл. 1).

Таблиця 1 – Інформація, яку збирають про комп'ютер

№	Тип і підтип клавіатури	Кількість кнопок миші	Ширина екрану	Висота екрану	Набір дискових пристроїв	Об'єм пам'яті	Дані про диск на якому встановлена програма
1	Ні	Так	Ні	Так	Ні	Так	Об'єм
2	Ні	Так	Так	Ні	Ні	Так	Об'єм
3	Так	Ні	Ні	Так	Ні	Так	Об'єм
4	Так	Ні	Так	Ні	Ні	Так	Об'єм
5	Ні	Так	Ні	Так	Так	Ні	Об'єм
6	Ні	Так	Так	Ні	Так	Ні	Об'єм
7	Так	Ні	Ні	Так	Так	Ні	Об'єм
8	Так	Ні	Так	Ні	Так	Ні	Об'єм
9	Ні	Так	Ні	Так	Ні	Так	Мітка тому
10	Ні	Так	Так	Ні	Ні	Так	Мітка тому
11	Так	Ні	Ні	Так	Ні	Так	Мітка тому
12	Так	Ні	Так	Ні	Ні	Так	Мітка тому
13	Ні	Так	Ні	Так	Так	Ні	Мітка тому
14	Ні	Так	Так	Ні	Так	Ні	Мітка тому
15	Так	Ні	Ні	Так	Так	Ні	Мітка тому
16	Так	Ні	Так	Ні	Так	Ні	Мітка тому
17	Ні	Так	Ні	Так	Ні	Так	Серійний №
18	Ні	Так	Так	Ні	Ні	Так	Серійний №
19	Так	Ні	Ні	Так	Ні	Так	Серійний №

20	Так	Ні	Так	Ні	Ні	Так	Серійний №
21	Ні	Так	Ні	Так	Так	Ні	Серійний №
22	Ні	Так	Так	Ні	Так	Ні	Серійний №
23	Так	Ні	Ні	Так	Так	Ні	Серійний №
24	Так	Ні	Так	Ні	Так	Ні	Серійний №
25	Ні	Так	Ні	Так	Ні	Так	Файлова система
26	Ні	Так	Так	Ні	Ні	Так	Файлова система
27	Так	Ні	Ні	Так	Ні	Так	Файлова система
28	Так	Ні	Так	Ні	Ні	Так	Файлова система
29	Ні	Так	Ні	Так	Так	Ні	Файлова система
30	Ні	Так	Так	Ні	Так	Ні	Файлова система

Рекомендовані для розробки програмного додатку засоби мови C++ і системи C ++ Builder

1. Збір інформації про комп'ютер:

// Отримання в буфері lpBuffer довжини nSize імені користувача поточного сеансу

BOOL GetUserName (LPTSTR lpBuffer, LPDWORD nSize);

/ * Отримання імені комп'ютера в буфері lpBuffer довжини nSize> = MAX_COMPUTERNAME_LENGTH + 1 * /

BOOL GetComputerName (LPTSTR lpBuffer, LPDWORD nSize);

/ * Отримання в буфері lpBuffer довжини uSize> = MAX_PATH шляху до каталогу з ОС Windows * /

UINT GetWindowsDirectory (LPTSTR lpBuffer, UINT uSize);

/ * Отримання в буфері lpBuffer довжини uSize> = MAX_PATH шляху до системного каталогу Windows * /

UINT GetSystemDirectory (LPTSTR lpBuffer, UINT uSize);

// Отримання типу (nTypeFlag = 0) або підтипу (nTypeFlag = 1) клавіатури

int GetKeyboardType (int nTypeFlag);

/ * Отримання кількості кнопок миші (nIndex = SM_CMOUSEBUTTONS), ширини (nIndex = SM_CXSCREEN) або висоти (nIndex = SM_CYSCREEN) екрану * /

int GetSystemMetrics (int nIndex);

/ * Отримання в буфері lpBuffer довжини nBufferLength рядка з кореневими каталогами всіх дисків, розділених 0-символами; результат – довжина отриманого рядка без заключного 0-символа * /

DWORD GetLogicalDriveStrings (DWORD nBufferLength, LPTSTR lpBuffer);

/ * Отримання в буфері * lpBuffer структури типу MEMORYSTATUS з характеристиками пам'яті комп'ютера (поле dwTotalPhys містить ціле число, що дорівнює загальному обсягу фізичної пам'яті в байтах) * /

VOID GlobalMemoryStatus (LPMEMORYSTATUS lpBuffer);

/ * Отримання інформації про обсяг поточного диска (lpRootPathName = NULL): кількості секторів в кластері (lpSectorsPerCluster), розмірі сектора (lpBytesPerSector), загальній кількості кластерів (lpTotalNumberOfClusters), lpNumberOfFreeClusters = NULL * /

BOOL GetDiskFreeSpace (LPCTSTR lpRootPathName,
LPDWORD lpSectorsPerCluster, LPDWORD lpBytesPerSector,
LPDWORD lpNumberOfFreeClusters,
LPDWORD lpTotalNumberOfClusters);

/ * Отримання інформації про поточний диск (lpRootPathName = NULL): мітці тома (в буфері lpVolumeNameBuffer довжини nVolumeNameSize), серійний номер (у змінній * lpVolumeSerialNumber), файлової системи (в буфері lpFileSystemNameBuffer довжини nFileSystemNameSize), lpMaximumComponentLength = NULL, lpFileSystemFlags = NULL * /

BOOL GetVolumeInformation (LPCTSTR lpRootPathName,
LPTSTR lpVolumeNameBuffer, DWORD nVolumeNameSize,
LPDWORD lpVolumeSerialNumber,
LPDWORD lpMaximumComponentLength, LPDWORD lpFileSystemFlags, LPTSTR lpFileSystemNameBuffer, DWORD nFileSystemNameSize);

2. Отримання та перевірка електронного цифрового підпису (ЕЦП) (константи, типи даних і прототипи функцій визначені у файлі wincrypt.h):

HCRYPTPROV, HCRYPTKEY, HCRYPTHASH – типи даних для дескрипторів криптопровайдера (CSP), криптографічного ключа, хеш-об'єкта

ALG_ID – тип даних для кодів криптографічних алгоритмів

/ * Ініціалізація криптопровайдера:
в * phProv записується його дескриптор,
psisContainer = NULL,
psz Provider = NULL,
dwProvType = PROV_RSA_FULL,
dwFlags = 0 або (якщо при першому запуску програми CryptAcquireContext повертає FALSE) реєстрація нового користувача в Криптопровайдери dwFlags = CRYPT_NEW_KEYSET * /

BOOL CryptAcquireContext (HCRYPTPROV * phProv, LPCSTR pszContainer,
LPCSTR pszProvider, DWORD dwProvType, DWORD dwFlags);

/ * Створення в Криптопровайдері з дескриптором hProv пари ключів ЕЦП (Algid = AT_SIGNATURE, dwFlags = 0) і запис дескриптора відкритого ключа в * phKey * /

BOOL CryptGenKey (HCRYPTPROV hProv, ALG_ID Algid,
DWORD dwFlags, HCRYPTKEY * phKey);

/ * Отримання у криптопровайдера з дескриптором hProv дескриптора відкритого ключа ЕЦП (dwKeySpec = AT_SIGNATURE) у змінній * phUserKey (якщо функція повертає FALSE, то пару ключів ЕЦП потрібно створити за допомогою функції CryptGenKey) * /

BOOL CryptGetUserKey (HCRYPTPROV hProv, DWORD dwKeySpec,
HCRYPTKEY * phUserKey);

/ * Створення порожнього хеш-об'єкта (hProv – дескриптор ініціалізованого криптопровайдера, Algid – код алгоритму хешування, hKey = 0, dwFlags = 0, в * phHash записується дескриптор хеш-об'єкта) * /

BOOL CryptCreateHash (HCRYPTPROV hProv, ALG_ID Algid,
HCRYPTKEY hKey, DWORD dwFlags, HCRYPTHASH * phHash);

/ * Додавання в хеш-об'єкт даних з буфера * pbData довжини dwDataLen (hHash – дескриптор хеш-об'єкта, dwFlags = 0) * /

BOOL CryptHashData (HCRYPTHASH hHash, CONST BYTE * pbData,
DWORD dwDataLen, DWORD dwFlags);

/ * Отримання для хеш-об'єкта з дескриптором hHash ЕЦП в буфері pbSignature довжини * pdwSigLen (після виконання функції в цю змінну записується фактична довжина ЕЦП); dwKeySpec = AT_SIGNATURE, sDescription = NULL, dwFlags = 0 * /

BOOL CryptSignHash (HCRYPTHASH hHash, DWORD dwKeySpec,
LPCTSTR sDescription, DWORD dwFlags, BYTE * pbSignature,
DWORD * pdwSigLen);

/ * Перевірка ЕЦП з буфера * pbSignature довжини dwSigLen для хеш-об'єкта з дескриптором hHash за допомогою відкритого ключа hPubKey (sDescription = NULL, dwFlags = 0) * /

BOOL CryptVerifySignature (HCRYPTHASH hHash, BYTE * pbSignature,
DWORD dwSigLen, HCRYPTKEY hPubKey, LPCTSTR sDescription,
DWORD dwFlags);

// Руйнування хеш-об'єкта з дескриптором hHash

BOOL CryptDestroyHash (HCRYPTHASH hHash);

// Руйнування ключа шифрування з дескриптором hKey

BOOL CryptDestroyKey (HCRYPTKEY hKey);

// Звільнення криптопровайдера з дескриптором hProv (dwFlags = 0)

BOOL CryptReleaseContext (HCRYPTPROV hProv, DWORD dwFlags);

3. Робота з реєстром Windows:

Клас TRegistry (визначений у файлі vcl \ registry.hpp):

• конструктор без параметрів;

• властивості:

HKEY RootKey (кореневий розділ реєстру, за замовчуванням HKEY_CURRENT_USER);

HKEY CurrentKey (поточний розділ реєстру, тільки для читання);

AnsiString CurrentPath (шлях до поточного розділу реєстру, тільки для читання).

• методи:

/ * Відкриття або (якщо CanCreate = true) при необхідності створення поточного розділу реєстру Key * /

bool OpenKey (const AnsiString Key, bool CanCreate);

/ * Запис (перезапис) в поточний розділ реєстру значення параметра Name з буфера Buffer довжини BufSize * /

void WriteBinaryData (const AnsiString Name, void * Buffer, int BufSize);

// Запис і закриття поточного розділу реєстру

void CloseKey ();

// Перевірка існування в реєстрі розділу Key

bool KeyExists (const AnsiString Key);

/ * Читання з поточного розділу реєстру значення параметра Name в буфер Buffer довжини BufSize * /

int ReadBinaryData (const AnsiString Name, void * Buffer, int BufSize);

Контрольні запитання

1. Що собою являє захист від несанкціонованого використання та копіювання?
2. Які технології захисту від НСК та НСВ відомі на сьогоднішній день?
3. Наведіть короткий опис технології захисту від НСК – створення особливо обумовлених носіїв;
4. Назвіть, будь ласка, групи систем захисту від несанкціонованого використання та копіювання.
5. Які існують методи захисту програмного забезпечення, шляхом прив’язки до параметрів комп’ютера?

ЛАБОРАТОРНА РОБОТА № 5

АЛГОРИТМИ ПОВЕДІНКИ ВІРУСНИХ ТА ІНШИХ ШКІДЛИВИХ (ЗЛОВМИСНИХ) ПРОГРАМ

Мета роботи: знайомство з деякими алгоритмами поведінки вірусних та інших шкідливих (зловмисних) програм.

Короткі теоретичні відомості

Історично перше визначення комп'ютерного вірусу було дано у 1984 р. Фредом Коеном: «Комп'ютерний вірус - це програма, яка може заражати інші програми, модифікуючи їх шляхом включення в них своєї, можливо зміненої копії, причому остання зберігає здатність до подальшого розмноження». Ключовими поняттями в цьому визначенні є здатність вірусу до саморозмноження і здатність до модифікації обчислювального процесу.

В даний час під комп'ютерним вірусом прийнято розуміти програмний код, що володіє наступними властивостями:

• здатністю до створення власних копій, які не обов'язково збігаються з оригіналом, але володіють властивостями оригіналу (самовідтворення);

• наявністю механізму, що забезпечує впровадження створюваних копій у виконувані об'єкти обчислювальної системи.

Зазначені властивості слід доповнити властивостями деструктивності і прихованості дій даної шкідливої програми в обчислювальному середовищі.

Основною і найбільш поширеною класифікацією комп'ютерних вірусів є класифікація за середовищем існування або за типами об'єктів комп'ютерної системи, в які впроваджуються віруси. Відповідно до цієї класифікації віруси діляться на файлові, завантажувальні, мережеві (черв'яки) і макровіруси.

Існує також багато комбінованих типів комп'ютерних вірусів.

Крім вірусів прийнято виділяти ще кілька видів шкідливих програм. Це троянські програми, логічні бомби, хакерські утиліти прихованого адміністрування віддалених комп'ютерів, програми, що крадуть паролі доступу до ресурсів Інтернет та іншу конфіденційну інформацію. Чіткого поділу між ними не існує: троянські програми можуть містити віруси, у віруси можуть бути вбудовані логічні бомби і т. д. Також до шкідливого програмного забезпечення відносять конструктори вірусів та поліморфні генератори.

Конструктор вірусів - це утиліта, призначена для виготовлення нових комп'ютерних вірусів. Відомі конструктори вірусів для DOS, Windows і макровірусів. Вони дозволяють генерувати вихідні тексти вірусів (ASM-файли), об'єктні модулі, і/чи безпосередньо заражені файли.

Деякі конструктори (VLC, NRLG) оздоблені стандартним віконним інтерфейсом, де за допомогою системи меню можна вибрати тип вірусу, об'єкти для зараження (COM і/чи EXE), наявність або відсутність самошифрування, протидії налагоджувачу, внутрішні текстові рядки, вибрати ефекти, що супроводжують роботу вірусу і т.п.

Інші конструктори (PS-MPC, G2) не мають інтерфейсу і зчитують інформацію про тип вірусу з конфігураційного файлу.

Поліморфні генератори (або поліморфік-генератори), як і конструкттори вірусів, не є вірусами в буквальному значенні цього слова, оскільки в їх алгоритми не закладаються функції розмноження, тобто відкриття, закриття і записування у файли, читання і записування секторів і т.д. Головною функцією подібного роду програм є шифрування тіла вірусу і генерація відповідного розшифровувача.

Постановка завдання

Варіант – номер за списком в журналі (див. варіант в табл. 2). Розробити програму, яка імітує деякі дії вірусу або іншої шкідливої програми та підготувати звіт про виконану роботу.

Таблиця 2 – Варіанти завдання до лабораторної роботи №5

Варіант	Дія вірусної або іншої шкідливої програми	Завдання	Вхідні дані процедури	Вихідні дані процедури	Примітки
1	**"Пошук жертви"**	Розробити і налагодити процедуру пошуку файлів по заданій масці у поточній папці	Маска файлу	Масив імен знайдених файлів і їх кількість	Маска файлів: *.exe
2	**"Підміна файлу"**	Розробити і налагодити процедуру заміни зазначеного файлу на інший вказаний файл	Імена двох файлів	Ознака успішності заміни	Підміна шляхом видалення файлу і використання його імені для зміни імені "вірусного" файлу
3	**"Модифікація файлу"**	Розробити і налагодити процедуру запису інформації з одного зазначеного файлу в кінець іншого зазначеного файлу	Імена двох файлів	Перетворений файл	
4	**"Принцип дії програми-шпигуна"**	Розробити процедуру копіювання вмісту всіх файлів, до яких звертається користувач в заданий файл	Ім’я файлу	Перетворений файл	У програмі створити форму звернення до декількох файлів (своєрідний каталог)
5	**"Створення звукових ефектів"**	Розробити процедуру запуску звукового файлу при настанні якої-небудь події (спроба доступу до файлу, каталогу, настання певного часу і т. п.)	Ім’я файлу, каталогу, заданий час або інше	Запуск звукового файлу	Вміст звукового файлу: творча робота
6	**"Перевірка файлу на зараження" (вірус не повинен заражати уже заражені файли)**	Розробити і налагодити процедуру пошуку заданого рядка у файлі	Ім’я файлу, рядок тесту	Логічна змінна	

7	**"Маскування"**	Розробити і налагодити процедуру запуску зазначеної програми на виконання	Ім’я файлу з програмою	Логічна змінна	Принцип маскування: після виконання даного алгоритму вірус запускає на виконання сам файл, щоб приховати від користувача факт зараження
8	**"Маскування"**	Розробити і налагодити процедуру зміни дати і часу створення зазначеного файлу на задані значення	Ім’я файлу, значення дати. Часу	Логічна змінна	
9	**"Маскування"**	Розробити і налагодити процедуру зміни атрибутів і розміру зазначеного файлу на задані значення	Ім’я файлу, значення атрибутів і розміру	Логічна змінна	
10	**"Принцип роботи логічної бомби"**	Розробити і налагодити процедуру видалення вмісту всіх файлів у заданому каталозі при настанні якого-небудь часу	Задане значення часу, ім’я каталогу	Логічна змінна	
11	**"Імітація збоїв ОС та апаратури"**	Розробити і налагодити процедуру появи повідомлень про збій ОС і / або апаратури при спробі доступу до якого-небудь файлу (файлам)	Ім’я файлу (файлів)	Поява повідомлень про збій	
12	**"Маскування"**	Розробити і налагодити процедури шифрування і дешифрування зазначеного файлу	Ім’я файлу	Логічна змінна	
13	**"Принцип роботи логічної бомби"**	Розробити і налагодити процедуру запуску деякої програми при настанні якої-небудь години або хвилини	Задане значення часу, імені файлу	Запуск деякої програми	
14	**"Принцип роботи банера"**	Розробити і налагодити процедуру запуску банера при спробі переходу по якому-небудь посиланні	Посилання, банер	Запуск банера	Зміст банера: творча робота

15	**"Принцип роботи логічної бомби"**	Розробити і налагодити процедуру видалення всіх файлів у заданому каталозі при настанні якого-небудь часу	Задане значення часу, імені каталогу	Логічна змінна	
16	**"Модифікація файлу"**	Розробити і налагодити процедуру запису інформації з одного зазначеного файлу в середину іншого файлу	Імена двох файлів	Перетворений файл	Місце в середині файлу повинно вибиратися випадковим чином
17	**"Принцип роботи логічної бомби"**	Розробити і налагодити процедуру запуску деякої програми при настанні якої-небудь дати	Дата, ім'я файлу, що запускається	Запуск деякої програми	
18	**"Жадібна програма"**	Розробити і налагодити процедуру створення копій заданого файлу (файлів) і розміщення їх в заданому каталозі (каталогах)	Ім'я файлу, кількість копій, заданий каталог	Створені копії	
19	**"Принцип роботи логічної бомби"**	Розробити і налагодити процедуру запуску деякої програми при настанні якого-небудь дня тижня	День тижня. ім'я файлу, що запускається	Запуск деякої програми	
20	**"Пошук жертви"**	Розробити і налагодити процедуру пошуку файлів по заданій масці у поточній папці	Маска файлу	Масив імен знайдених файлів і їх кількість	Маска файлів: *.bat
21	**"Модифікація файлу"**	Розробити і налагодити процедуру перемішування символів (рядків) у файлі	Ім'я файлу	Перетворений файл	
22	**"Маскування"**	Розробити процедуру підміни зараженого файлу незараженим при зверненні до нього користувача	Ім'я файлу	Відкритий файл	Для перевірки працездатності програми створити 2 файли: "заражений" і "незаражений"
23	**"Підміна файлу"**	Розробити і налагодити процедуру заміни зазначеного файлу на інший вказаний файл	Імена двох файлів	Ознака успішності заміни	Підміна шляхом заміни вмісту файлу на вміст "вірусного" файлу

24	**"Жадібна програма"**	Розробити і налагодити процедуру створення копій заданого каталогу (каталогів)	Заданий каталог, кількість копій	Створені копії	
25	**"Перевірка файлу на зараження" (вірус не повинен заражати уже заражені файли)**	Розробити і налагодити процедуру перевірки наступного факту: чи є вміст двох файлів однаковим	Імена двох наявних файлів	Логічна змінна	
26	**"Перевірка файлу на зараження" (вірус не повинен заражати сам себе)**	Розробити і налагодити процедуру перевірки наступного факту: чи не є деякий файл, заданий своїм ім'ям, тією програмою, яка цю перевірку здійснює (тобто самою запущеною програмою)	Ім’я файлу, який перевіряється	Логічна змінна	

Контрольні запитання

1. Що відносять до шкідливого програмного забезпечення і як його класифікують?
2. Класифікуйте комп’ютерні віруси за середовищем їх існування та за способом зараження комп’ютерів.
3. Наведіть класифікацію вірусів за алгоритмами, які вони використовують при функціонуванні, та за своїми деструктивними можливостями.
4. Наведіть алгоритм роботи файлових вірусів.
5. В чому полягає принцип дії завантажувального вірусу?
6. Наведіть алгоритм роботи завантажувального вірусу: резидентного і нерезидентного.
7. Дайте загальну характеристику макро-вірусам, їх особливостям та розташуванню.
8. В чому особливості мережевих вірусів?
9. Охарактеризуйте стелс-віруси. Які різновиди цих вірусів ви знаєте?
10. Для чого існують і як функціонують конструктори вірусів та поліморфік-генератори?

ЛАБОРАТОРНА РОБОТА № 6
АЛГОРИТМИ ПОПЕРЕДЖЕННЯ І ВИЯВЛЕННЯ ВІРУСНИХ ЗАГРОЗ

Мета роботи: знайомство з деякими алгоритмами попередження і виявлення вірусних загроз.

Короткі теоретичні відомості

Для захисту від комп'ютерних вірусів та інших шкідливих програм можуть використовуватися:

- загальні методи та засоби захисту інформації;
- спеціалізовані програми для захисту від вірусів;
- профілактичні заходи, що дозволяють зменшити ймовірність зараження вірусами.

Існують два основних різновиди загальних методів і засобів захисту інформації, також ефективних при боротьбі з вірусними загрозами:

- засоби копіювання інформації;
- засоби розмежування доступу.

При зараженні комп'ютера вірусом важливо його знайти. До зовнішніх ознак прояву діяльності вірусів можна віднести наступні:

- виведення на екран непередбачених повідомлень або зображень;
- подача непередбачених звукових сигналів;
- зміна дати і часу модифікації файлів;
- зникнення файлів і каталогів чи спотворення їх вмісту;
- часті зависання і збої в роботі комп'ютера;
- повільна робота комп'ютера;
- неможливість завантаження ОС;
- істотне зменшення розміру вільної оперативної пам'яті;
- припинення роботи або неправильна робота раніше успішно функціонуючих програм;
- зміна розмірів файлів;
- несподіване значне збільшення кількості файлів на диску.

Але мало помітити, що комп'ютерна система піддалася впливу шкідливого ПЗ, необхідно виявити джерело загрози. До основних методів виявлення комп'ютерних вірусів можна віднести наступні:

- метод порівняння з еталоном;
- евристичний аналіз;
- антивірусний моніторинг;
- метод виявлення змін;
- вбудовування антивірусів та ін.

Розрізняють такі види антивірусних програм:

- програми-фаги (сканери);
- програми-ревізори (CRC-сканери);
- програми-блокувальники;

– програми-імунізатори.

Однак, абсолютно надійних програм, що гарантують виявлення і знищення будь-якого вірусу, не існує. Важливим методом боротьби з комп'ютерними вірусами є своєчасна профілактика. Щоб істотно зменшити ймовірність зараження вірусом і забезпечити надійне зберігання інформації на дисках, необхідно виконувати наступні заходи профілактики:

- застосовувати тільки ліцензійне ПЗ;
- оснастити комп'ютер сучасними антивірусними програмами і постійно оновлювати їх версії;
- завжди перевіряти знімні носії інформації на наявність вірусів (запускаючи антивірусні програми свого комп'ютера) перед зчитуванням з них інформації, записаної на інших комп'ютерах;
- при перенесенні на свій комп'ютер файлів у заархівованому вигляді перевіряти їх відразу ж після розархівації на жорсткому диску, обмежуючи область перевірки тільки знову записаними файлами;
- періодично перевіряти на наявність вірусів жорсткі диски комп'ютера, запускаючи антивірусні програми для тестування файлів, пам'яті і системних областей дисків;
- завжди захищати знімні носії інформації від запису при роботі на інших комп'ютерах, якщо на них не проводитиметься запис інформації;
- обов'язково робити на знімних дисках архівні копії цінної для користувача інформації;
- використовувати антивірусні програми для вхідного контролю всіх виконуваних файлів, одержуваних з комп'ютерних мереж.

Постановка завдання

Варіант – номер за списком в журналі (див. варіант в табл. 3). Розробити програму, яка імітує деякі (див. варіант) дії щодо попередження вірусних загроз, виявлення і видалення вірусних та інших шкідливих програм та підготувати звіт про виконану роботу.

Таблиця 3 – Варіанти завдання до лабораторної роботи №6

Варіант		Завдання	Вхідні дані процедури	Вихідні дані процедури	Додаткові умови
1	Алгоритм роботи антивірусної програми-ревізора	Ревізори запам'ятовують початковий стан файлів / каталогів, тоді, коли комп'ютер ще не заражений вірусом, а потім періодично порівнюють поточний стан файлу / каталогу з початковим. Якщо виявлені зміни, то на екран дисплея виводяться повідомлення. Розробити процедуру пошуку заданих (див. додакові умови) змін у файлі / каталозі.	Ім'я файлу (файлів) / Ім'я каталогу (каталогів)	Повідомлення про наявність / відсутність змін	Пошук змін у даті та часу створення файлу
2					Пошук змін в атрибутах і розмірі файлу
3					Пошук змін у змісті файлу
4					Пошук змін у змісті каталогу
5	Виявлення файлів-компаньйонів	Програма повинна здійснювати пошук файлів-компаньйонів (виконувані файли з тією ж назвою, що й початковий файл, але іншим розширенням) і за рішенням користувача здійснювати наступні дії: (див. дод. ум.)	Ім’я файлу	Список виявлених файлів-компаньйонів	Видалення файлів- компаньйонів
6					Переміщення файлів- компаньйонів у інший каталог (на карантин)
7	Виявлення ознак зараження вірусом	Розробити процедуру виявлення копій файлів у заданому каталозі. Здійснювати пошук по імені файлу і по вмісту. Інформувати користувача. Пропонувати на вибір наступні дії: (див. дод. ум.)	Ім’я файлу, каталогу	Список виявлених копій	Видалення виявлених копій
8					Переміщення виявлених копій у інший каталог (на карантин)
9	Профілактика зараження вірусом (Резервне копіювання)	Розробити процедуру створення резервних копій. Передбачити можливість вибору користувачем періодичності створення резервних копій (див. дод. ум.). При цьому повинні робитися копії лише тих файлів, які були створені або змінені в період після попередньої процедури копіювання.	Ім’я каталогу	Логічна змінна	Періодичність копіювання: раз на тиждень (надати можливість вибору дня тижня)
10					Періодичність копіювання: через день (надати можливість вибору парних або непарних чисел)

11					Періодичність копіювання: раз на кілька годин (надати можливість вибору інтервалу часу, що проходить між процедурами копіювання)
12	Виявлення вірусного коду в тілі файлу	Розробити і налагодити процедуру пошуку заданого рядка цілком або частково в заданих файлах (див. дод. ум.). У разі виявлення вірусного коду в тілі файлу реалізувати наступний алгоритм «лікування»: (див. дод. ум.)	Рядок, ім’я файлу (файлів, каталогу)	Логічна змінна	Пошук заданого рядка і його фрагментів (слів) у вказаному файлі. Алгоритм "лікування": видалення рядка або його фрагментів
13					Пошук заданого рядка і його фрагментів (слів) у вказаному файлі. Алгоритм "лікування": переміщення зараженого файлу в інший каталог (на карантин)
14					Пошук заданого рядка і його фрагментів (слів) у вказаному файлі. Алгоритм "лікування": видалення зараженого файлу
15					Пошук заданого рядка у всіх текстових файлах заданого каталогу. Алгоритм "лікування": видалення рядка з усіх файлів
16					Пошук заданого рядка у всіх текстових файлах заданого каталогу. Алгоритм "лікування": переміщення заражених файлів в інший каталог (на карантин)
18	Захист від клавіатурних шпигунів	Розробити генератор одноразового пароля на основі псевдовипадкового вибору символів із даних, введених користувачем. Застосувати наступний алгоритм ГПВЧ: (див. дод. ум.)	Масив із набором даних користувача (для спрощення завдання: кожен елемент масиву -	Одноразовий пароль довжиною N символів	X_i=round(10* sin(i* sin(i/У$_i$)) +10), де X_i – номер елемента, який обирається, в масиві і - лічильник [1;N]; У - елемент в масиві даних користувача, N=10. Мінімальна кількість елементів масива: 20
19					Лінійний конгруетний метод (функція Random), N=12

20			цифра з даних користувача: номери паспорта, дати народження і т.п.)		Метод Фібоначчі з запізнюваннями *: $X_i = \begin{cases} Y_{i-a} - Y_{i-b}, якщо Y_{i-a} \geq Y_{i-b} \\ -(Y_{i-a} - Y_{i-b}), якщо\ Y_{i-a} < Y_{i-b} \end{cases}$ де X_i - номер елемента, який обирається, в масиві і - лічильник [max(a,b)+1; N+ max(a,b)+1]; У- елемент в масиві даних користувача, a,b – цілі додатні числа, які називаються лагами, рекомендовані значенння (a,b)=(17,5), N=7. Мінімальна кількість елементів масива: max(a,b)
21	Захист від масової розсилки спаму методом **CAPTCHA**	Розробити програму реєстрації користувача з перевіркою методом CAPTCHA. Для завершення реєстрації користувачеві повинна бути запропонована така задача, яку з легкістю може вирішити людина, але яку незрівнянно складніше вирішити комп'ютеру (див. дод. ум.).	Дані користувача	Логічна змінна	Як задачу запропонувати користувачеві ввести число (слово) з картинки (однієї або декількох)
22					Як задачу запропонувати користувачеві здійснити показану на зображенні просту арифметичну операцію
23					Як задачу запропонувати користувачеві вибрати з декількох картинок одну, відповідну певній умові
24	Захист від програм відкриття пароля	Розробити "розумну" програму запиту паролів (smart password asker). Цей метод передбачає використання спеціальної програми запиту паролів, яка працює не за стандартним алгоритмом, а за алгоритмом з псевдовипадковим результатом (див. дод. ум.). Реалізувати 7-10 запитів пароля.	Пароль (див. дод. ум.)	Логічна змінна	Розробити програму, яка запитує лише частину пароля: перші три символи
25					Розробити програму, яка запитує не сам пароль, а суму цифр, що входять в пароль
26					Розробити програму, яка запитує не сам пароль, на суму частини пароля і числа місяця поточної дати
27					Розробити програму, яка запитує лише частину пароля: перший, третій і останній символ

* У даній лабораторній роботі представлений один з широко поширених датчиків Фібоначчі з деякими змінами, так як датчик розрахований на генерацію випадкових дійсних чисел з діапазону $[0,1)$, а для виконання завдання потрібно згенерувати ціле число.

Контрольні запитання

1. Які існують методи і засоби для захисту від комп’ютерних вірусів?
2. За якими ознаками можна виявити факт зараження комп'ютерним вірусом?
3. Які заходи рекомендується вживати, щоб запобігти зараженню комп'ютерним вірусом?
4. Що таке антивірусна програма? Які типи антивірусів ви знаєте?
5. Охарактеризуйте різновиди антивірусних програм.
6. Які правила треба знати та виконувати, щоб не наражати свій комп'ютер на небезпеку зараження комп'ютерними вірусами?
7. Що треба робити, якщо ви виявили зараження комп'ютера вірусом?

ЛАБОРАТОРНА РОБОТА № 7

АНАЛІЗ ТА ДОСЛІДЖЕННЯ СУЧАСНИХ ЗАСОБІВ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Мета роботи: Ознайомлення з основними функціями, перевагами і недоліками сучасного антивірусного ПЗ.

Короткі теоретичні відомості

На сьогоднішній день перелік доступних антивірусних програм досить великий. Вони розрізняються як за ціною, так і за своїми функціональними можливостями. Найбільш потужними (і як правило, найбільш дорогими) антивірусними програмами є насправді пакети спеціалізованих утиліт, здатних при спільному їх використанні забезпечити різнобічний захист комп'ютерної системи.

Більшість сучасних антивірусних пакетів виконують такі функції:

- сканування пам'яті і вмісту дисків;
- сканування в реальному режимі часу за допомогою резидентного модуля;
- розпізнавання поведінки, характерної для комп'ютерних вірусів;
- блокування та / або видалення виявлених вірусів;
- відновлення заражених інформаційних об'єктів;
- примусова перевірка підключених до корпоративної мережі комп'ютерів;
- віддалене оновлення антивірусного програмного забезпечення і баз даних через Інтернет;
- фільтрація трафіку Інтернету на предмет виявлення вірусів у переданих програмах і документах;
- виявлення потенційно небезпечних Java-аплетів і модулів ActiveX;
- ведення протоколів, що містять інформацію про події, що стосуються антивірусного захисту та ін.

Постановка завдання

1. Підготувати коротку доповідь по заданому питанню (див. табл. 4), використовуючи будь-які доступні джерела інформації. Варіант – номер за списком в журналі.

Рекомендація: зібраний матеріал буде найбільш актуальним, якщо включити в нього дані, отримані практичним шляхом. Для цього при можливості, встановіть демонстраційну версію заданого пакета ПЗ і протестуйте її протягом декількох днів.

2. Заповнити таблицю "Пакети антивірусних програм" на основі підготовленого матеріалу, а також доповідей інших студентів.

3. Провести аналіз зібраної інформації і зробити висновки.

Таблиця 4 – Варіанти завдання до лабораторної роботи №7

Пакет антивірусного ПЗ	Основні функції	Переваги	Недоліки
Антивірус Касперського	1	2	3
Антивірус Dr.Web для Windows	4	5	6
Panda Antivirus	7	8	9
ESET NOD32 Антивірус	10	11	12
Avast! Free Antivirus	13	14	15
Avira AntiVir Personal	16	17	18
Norton AntiVirus	19	20	21
Trend Micro Internet Security	22	23	24
Microsoft Security Essentials	25	26	27
McAfee VirusScan	28	29	30
Будь-який інший за вибором студента	31	32	33

Контрольні запитання

1. Що таке антивірус?
2. Наведіть, будь ласка, класифікацію антивірусних програм.
3. Наведіть приклади антивірусів. Коротко охарактеризуйте їх.
4. Які основні функції антивірусів ви знаєте?
5. Чи можна заразити вірусом простий текстовий файл, що має розширення txt?
6. Вкажіть основні функції антивірусу Касперського.
7. Проведіть порівняльний аналіз антивірусного ПЗ Panda Antivirus та Norton AntiVirus вказуючи на їх переваги та недоліки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Гроувер Д., Сатер Р., Фипс Дж. И др. Защита программного обеспечения. Под ред. В.Г. Потемкина. – М.:Мир, 1992г., 286 с.

2. Дудатьев А.В., Каплун В.А., Семеренко В.П. Захист програмного забезпечення. Ч.1. Навчальний посібник. – Вінниця: ВНТУ, 2005. – 140 с

3. Партыка Т.Л., Попов И.И. Информационная безопасность. – М.: ФОРУМ: ИНФРА-М, 2004. – 368 с.

4. Панов А.С. Реверсинг и защита программ от взлома. СПб: БХВ-Петербург, 2006 – 256с.

5. Скляров Д.В. Искусство защиты и взлома информации. – СПб.: БХВПетербург, 2004. – 288 с.

6. Щербаков А. Защита от копирования: построение программных средств. – М.: Эдель, 1992. – 334с.

7. http://inf.rosvita.rv.ua/index.php?pl=lesinf08s018.