УДК 681.321;621.395

Гладиш С.В.
Gladish S.V.

РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ЗАХИСТУ ДЛЯ СИСТЕМИ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ ЛИНІЙНО-КАБЕЛЬНИХ СПОРУД ТА МЕРЕЖ ЗВ'ЯЗКУ «АПР ЛКС»

DEVELOPMENT OF A PROTECTION SOFTWARE FACILITY FOR THE AUTOMATED SYSTEM OF DESIGNING LINEAR-CABLE STRUCTURES AND COMMUNICATION NETWORKS – "APR LKS"

Анотація. Проаналізовано можливості розробки програмних засобів захисту для комплексу програмного забезпечення «АПР ЛКС». Порівняно ефективність варіантів їх реалізації за допомогою механізмів, які надає ОС Windows XP/2000/NT на різних рівнях мережної архітектури. Запропоновано структуру та варіант реалізації програмного засобу захисту.

Summary. The opportunities of protection software development for the software complex "APR LKS" are determined. The comparison of the effectiveness of their realization variants with the help of different mechanisms which gives Windows XP/2000/NT at different network architecture levels is done. The structure and a variant of security software realization is offered.

Існує загальна проблема забезпечення інформаційної безпеки інформаційно-телекомунікаційних систем. Ця проблема включає в себе наукові і технічні задачі захисту інформаційних ресурсів та розробки політики інформаційної безпеки системи автоматизації проектних робіт з будівництва, розширення та реконструкції телекомунікаційних мереж (ТМ). Для реалізації положень Закону України "Про телекомунікації" щодо захисту інформації необхідно забезпечувати інформаційну безпеку ресурсів комплексно на всіх етапах і стадіях життєвого циклу: технічного завдання, проектування, створення системи інформаційної безпеки (СІБ), її експлуатації та утилізації.

Спочатку подібні задачі вирішувались двома методами: розробкою захищених систем проектування в захищених операційних середовищах [1]; створенням систем захисту, надбудованих над стандартною операційною системою [2]. Тепер стало можливим застосування проміжного методу доробки модулів стандартної операційної системи (ОС) та розробки програмного засобу захисту, який міг би бути інтегрованим з комплексом ПЗ «АПР ЛКС», розширюючи можливості щодо захисту проектної інформації, та ефективно використовуючи штатні механізми та сервіси безпеки, що надають ОС Windows XP/2000/NT на різних рівнях мережної архітектури. Розв'язання цієї задачі започатковано в роботі [3], де проаналізовано особливості автоматизації проектних робіт з будівництва, розширення та реконструкції ТМ з точки зору інформаційної безпеки. При вирішенні задачі був використаний програмний продукт компанії *NuMega* – «***SoftIce***» – засіб відладки, який дозволяє відстежувати такі компоненти ядра, як драйвери.

Але на сьогодні ці задачі не знайшли остаточного вирішення. Розробка захищених операційних систем вимагає надто великих ресурсів і часу. Надбудовані системи захисту орієнтовані на вузьке коло застосувань.

Мета даної роботи: дослідити та визначити можливості інтеграції програмних засобів захисту для системи автоматизованого проектування лінійно-кабельних споруд та мереж зв'язку «АПР ЛКС» на різних рівнях мережної архітектури ОС Windows XP/2000/NT, починаючи від прикладного рівня і закінчуючи рівнем драйверів пристроїв.

1. Комплекс програмного забезпечення "АПР ЛКС" як об'єкт захисту інформації. Комплекс програмного забезпечення "АПР ЛКС", розроблений Центром з проектування лінійно-кабельних споруд та мереж зв'язку «Проект-Телеком» ОНАЗ ім. О.С.Попова, призначений для автоматизації процесу проектування, складання та випуску проектної документації по лінійно-кабельних спорудах зв'язку при будівництві, розширенні та реконструкції ТМ. Комплекс ПЗ «АПР ЛКС» працює на базі ОС Windows XP/2000, які завдяки своїй універсальності, мережній архітектурі, розширюваності є найбільш розповсюдженими. Основні функції комплексу ПЗ «АПР ЛКС» згідно з [5]:

– розробка та вибір оптимальних кабельних трас первинної мережі, розробка та вибір оптимальних кабельних трас кабельної каналізації від АТС до шаф та абонентських виносів, розробка та вибір оптимальних кабельних трас кабельної розподільної мережі шафного району та зон доступу із використанням абонентських виносів, вибір оптимальних кабельних трас у зоні прямого живлення;

– розбивка території на зони прямого живлення, шафні райони та зони доступу з використанням абонентських виносів;

– оптимізація місця розташування АТС, підстанцій, розподільних шаф у шафних районах по мінімальних витратах на магістральні кабелі;

– розрахунок та прогнозування міжстанційних та внутрішньостанційних навантажень.

Комплекс ПЗ «АПР ЛКС» складається із 5 основних модулів (рис. 1). Кожен модуль є окремою програмою, яка під'єднується до спільної бази даних, зокрема, модуля друку. Модуль друку невід'ємний від векторного редактора, тому що отримує дані для друку безпосередньо від векторного редактора, та працює тільки в його середовищі.

Можна виділити дві категорії користувачів: програміст (він також є і адміністратор бази даних); проектувальник.

У функції першого входить інсталяція "АПР ЛКС", супроводження його в процесі експлуатації, формування і ведення бази, а також робота з редактором умовних зображень для створення еталонних об'єктів, які в подальшому використовує проектувальник. Проектувальник виконує еталонні проектні процедури, якими користується при вирішенні задач по створенню проектів.

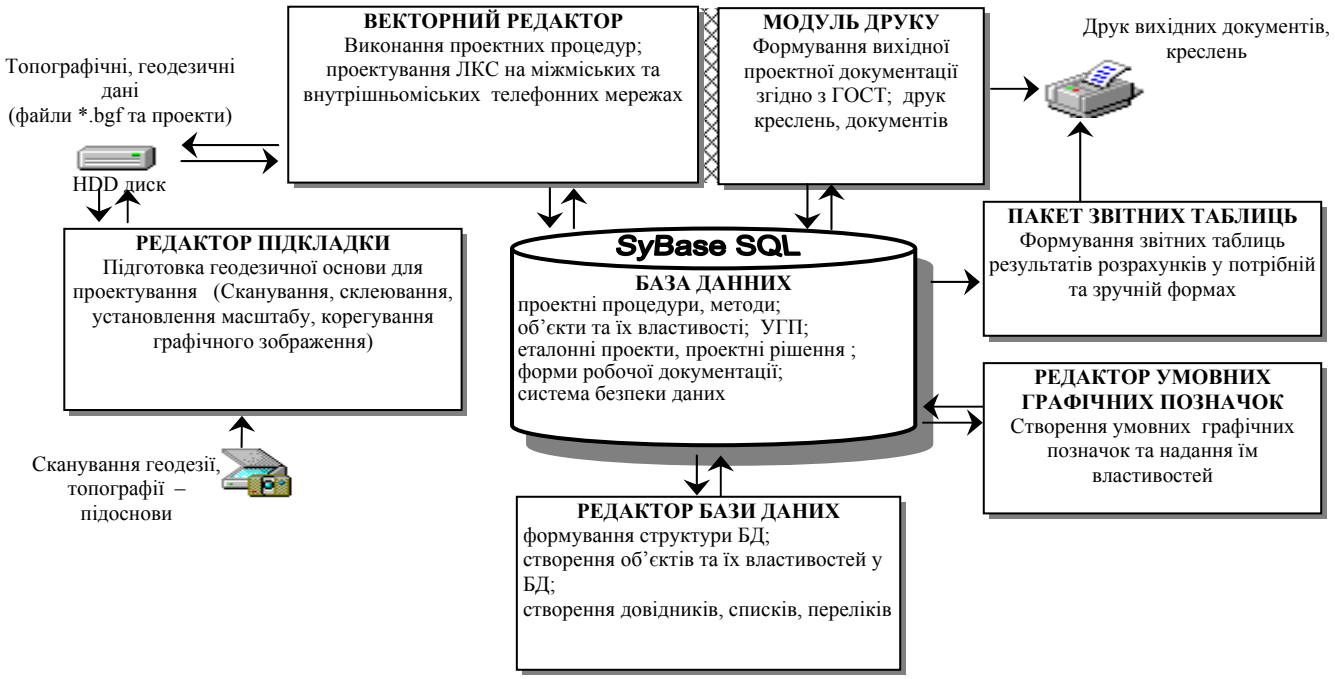

Рисунок 1 – Структурна схема "АПР ЛКС"

Програмний код представлений у файлах формату *.exe та не підлягає редагуванню. Вихідний код програми користувачеві не постачається. Права на можливість зміни у вихідному коді програми має тільки розробник системи. Файли програм (*.exe) основані на стандартних бібліотеках компонентів Windows, які вміщені у кожний файл. Перейменування файлів заборонено. Переміщувати файли програми дозволено у межах одного комп'ютера, на якому була установлена система. При пошкодженні файлів внаслідок дій комп'ютерних вірусів чи пошкодження дискових носіїв, на яких розміщена система, вони втрачають працездатність і відновлюються шляхом заміни із резервної копії (чи переустановленням системи). Поновлення програми до наступної версії виконується перезаписом старої.

Усі дані при роботі системи зберігаються у централізованій базі даних. Файли бази даних розміщені на сервері (при роботі у мережі) чи локально на комп'ютері (якщо комп'ютер не є сервер – то користування даними виконується у монопольному режимі). На сервері установлюється мережна версія системи керування базою даних. При роботі системи файли бази даних дзеркально резервуються на інший, незалежний носій даних. Таким чином підвищується цілісність даних і зменшується ймовірність втрати. Контроль за роботою бази даних здійснюється автоматично за допомогою СКБД SyBase AnyWhere версії 5.5. Поновлення бази даних виконується спеціальними програмами, які постачаються розробником, при виході нової версії. Файли бази даних це:

– Apr_base.db, є головний і має найбільший розмір (в залежності від наповнювання робочими даними);

– APR_BASE.log, веде реєстрацію всіх дій з даними, і дає можливість відновлювати дані;

– APR_BASE.mlg – дзеркальна копія файла APR_BASE.log, який розташовується на іншому незалежному носію даних;

– count_SL.dll – динамічна бібліотека “Розрахунок кількості міжстанційних ліній”.

ОС Windows XP/2000/NT на всіх рівнях мережної архітектури надає можливості інтеграції додаткових програмних модулів, які можуть контролювати запити мережних програмних компонентів відповідних рівнів і реалізовувати функції захисту інформації. Але в багатьох випадках виникає необхідність використовувати недостатньо документовані компанією Microsoft методи.

2. Порівняльний аналіз варіантів реалізації програмних засобів захисту. Розглянемо особливості реалізації програмних засобів захисту на різних рівнях мережної архітектури ОС Windows XP/2000/NT. Таке дослідження має установити, як і куди можна інтегрувати засіб захисту, яким чином цьому засобу надати максимальні можливості з боку ОС, оскільки від цього може залежати вирішення конкретних практичних завдань захисту, які він зможе реалізувати. Під реалізацією захисту на будь-якому рівні будемо розуміти: чи можливо за допомогою розробки компонентів цього рівня реалізувати функції захисту та чи можливо реалізувати програмний засіб захисту, який міг би впливати будь-яким чином на вже існуючі компоненти даного рівня і виконувати функції захисту.

Реалізація захисту на рівні застосувань і власних DLL. Захист на рівні застосування не є прозорим. При запуску застосування створюється новий процес, адресний простір якого захищено від інших процесів. Тому застосування може захищати лише свої дані. Для того, щоб дані інших застосувань могли оброблятись застосуванням, яке реалізує захист цих даних, необхідно використовувати спеціальні механізми міжпроцесорної комунікації: копіювання даних з одного адресного простору в інший, або створення області, яка була б видимою з обох адресних просторів.

Реалізація захисту на рівні власної DLL також не забезпечує прозорого захисту. Для того щоб захист даних застосувань виконувала бібліотека DLL необхідно, щоб застосування викликало функції цієї DLL. Після того, як виконуваний код DLL буде спроектовано на адресний простір викликаючого процесу, код і дані DLL стануть частиною процесу.

Рівень застосувань і власних DLL не має всіх можливостей щодо реалізації функцій захисту мережних ресурсів. Тільки у взаємодії з власним драйвером застосування може стати ефективним засобом захисту.

Реалізація захисту на рівні системних DLL. До цих DLL відносяться бібліотеки, які надають застосуванням мережні інтерфейси:
Netapi32.dll, Ws2_32.dll, Wsock32.dll, Mpr.dll, Kernel32.dll, Wininet.dll, Rpcrt4.dll, Rasapi32.dll, Nddeapi.dll, Tapi32.dll, Dlcapi.dll, Wsnmp32.dll, mgmtapi.dll, snmpapi.dll.

Схема загрузки та використання функцій цих DLL дозволяє реалізувати захист на рівні системних DLL наступними способами:

– змінення в *Import Address Table* відповідних адрес на адреси власних оброблювачів;

– створення власної DLL з тим же ім’ям, в якій замінюється частина функцій;

– змінення коду системних DLL з метою передачі керування (командами *jump*, *call*) власному коду.

Дані методи є недокументованими і дуже складними в реалізації. Але не дивлячись на це, програмні продукти, побудовані на цих методах все ж існують, наприклад, ***«PrudensSpy»*** реалізує технологію перехоплення будь-яких DLL, в тому числі й системних.

Реалізація захисту на рівні системного API. Знаючи індексний номер функції, яка реалізує системний сервіс, можна за таблицею розподілу системних сервісів (*KeServiceDescriptionTable*), яка експортується з ntoskrnl.exe і обробляється перериванням *int 2E*, визначити адресу початку необхідної функції та замінити її на адресу початку власного оброблювача режиму ядра. Після виконання власного оброблювача, необхідно повернути керування за старою адресою, щоб дати можливість стандартному оброблювачу виконати запитувані дії. Таблиця *KeServiceDescriptionTable* знаходиться у системній області пам’яті, тому така заміна може бути зроблена тільки кодом режиму ядра – драйвером.

Реалізація захисту на рівні драйвера MUP. Драйвер *mup.sys* відіграє важливу роль при відкритті файлів та пристроїв за ім’ям UNC. Тому доцільно розглядати можливості реєстрування та аудиту відкриття файлів та пристроїв за допомогою драйвера-фільтра, під’єднаного до драйвера *mup.sys*.

Реалізація захисту на рівні драйверів файлових систем. ОС Windows XP/2000/NT надають стандартний механізм для перехоплення запитів вводу/виводу до драйвера файлової системи – використання драйверів-фільтрів. Для його реалізації можна використовувати книгу [7], яка присвячена розробці драйверів файлових систем, зокрема реалізації драйверів-фільтрів. Завадою реалізації захисту на рівні драйверів файлових систем є закритість і недокументованість цього інтерфейсу. Вперше Microsoft пролила світло на це питання в 1997 році, випустивши інструмент для розробки драйверів файлових систем, що містить заголовні файли для успішної компіляції драйверів за допомогою MS Visual C++. Вдалим прикладом реалізації фільтра до драйверів файлових систем є програма «***FileMon***», розроблена *Mark Russinovich, Bruce Cogwell* [7].

Реалізація захисту на рівні транспортного драйвера. Розробка таких драйверів документована в [4] розділ Network Drivers, підрозділ Intermediate NDIS drivers and TDI drivers. Є приклад транспортного драйвера в DDK\src\network\tdi. Крім того в принципі є можливість реалізувати драйвер-фільтр, який буде приєднано до об'єкта пристрою, який створюється драйвером транспорту. Наприклад, до об'єктів-пристроїв *\Device\Tcp* та *\Device\Udp*, що створюються драйвером TCP/IP.

Реалізація захисту за допомогою перехоплення функцій бібліотеки NDIS. Ідея цього методу полягає у створенні оболонки над бібліотекою NDIS. Метод дуже важко реалізувати на практиці, він недокументований, але надзвичайно ефективний. Його реалізація полягає у зміні адрес необхідних функцій бібліотеки NDIS, в результаті можна отримати повний контроль над усіма мережними операціями в системі.

Реалізація захисту на рівні мережного драйвера проміжного рівня, підтримуючого інтерфейс NDIS. Розробка проміжних драйверів документована в [4], розділ Network Drivers, підрозділ Intermediate NDIS drivers and TDI drivers. Є приклад проміжного драйвера в DDK\src\network\packet\driver.

Розробка захисту на рівні драйверів мережних пристроїв. Розробка власних драйверів мережних пристроїв, що підтримують інтерфейс NDIS, документована в [4], розділ Network Drivers, підрозділ Miniport NIC drivers. Є багато прикладів в DDK\src\network. Якщо пристрій не є мережною картою, то можна скористатися загальними рекомендаціями з розробки драйверів в розділі Kernel-mode drivers [4].

Можемо провести порівняльний аналіз варіантів реалізації програмних засобів захисту відносно наступних факторів (табл. 1).

Об'єм контрольованої інформації. Цей фактор визначає, які дані проходять через засіб захисту і відповідно можуть ним контролюватися та оброблятися.

Складність реалізації. Визначає складність реалізації засобу захисту, як з позиції недокументованості, так і складності налагодження. Низька складність означає, що при розробці використовуються добре документовані інтерфейси і доступні засоби налагодження. Середня складність означає, що використовуються документовані інтерфейси, але налагодження засобів захисту не проста в зв'язку з тим, що даний засіб виконується в режимі ядра, більшість помилок в ньому призводять до краху ("зависання") ОС, при налагодженні доводиться працювати з асемблером. Висока складність означає, що до проблем з налагодженням додається часткова або повна відсутність документації для розробки засобу захисту.

Можливості інтеграції захисту. Визначає, чи дозволяє ОС можливість інтеграції (установлення) засобу захисту й розширення своєї функціональності на даному рівні.

Прозорість захисту означає, чи повинен користувач виконати будь-які дії для того, аби захистити свою інформацію за допомогою засобу захисту.

Можливості, які ОС надає засобу захисту. В залежності від того, чи виконується засіб захисту в режимі користувача, або в привілегірованому режимі – режимі ядра, йому надаються різні можливості.

В режимі ядра дозволено виконання всіх команд процесора та доступна системна область пам'яті й обладнання, на відміну від режиму користувача, в якому деякі команди заборонено, а системні області пам'яті є недосяжними. Засоби захисту рівня ядра можуть використовувати внутрішні інтерфейси для взаємодії з компонентами ОС. Модуль захисту, який виконується в режимі ядра, може підвищувати або знижувати *IRQL* процесора, приховуючи таким чином переривання з рівними або меншими *IRQL*.

Таблиця 1 – Порівняльний аналіз варіантів реалізації програмних засобів захисту

Рівень	Об'єм інформації	Складність реалізації	Інтеграція з ОС	Прозорість захисту	Можливості
Застосування	Тільки дані самого застосування	Низька	Є	-	Мінімальні
Власна DLL	Дані застосувань, які використовують цю DLL	Низька	Є	-	Мінімальні
Системна DLL	Дані застосувань, які використовують системну DLL	Висока	Немає	+	Залежно від способу реалізації
Мережний сервіс	Залежить від мережного сервісу	Висока	Немає	+	Залежно від способу реалізації
Системний API	Дані всіх застосувань	Висока	Немає	+	Максимальні
Драйвер файлової системи	Дані застосувань, які використовують відповідний мережний API	Висока	Є	+	Максимальні
Транспортний драйвер	Дані застосувань, які використовують цей транспорт + застосування, які прямо взаємодіють з іншим транспортом	Висока	Є	+	Максимальні
Драйвер *ndis.sys*	Дані всіх застосувань	Висока	Немає	+	Максимальні
Проміжний драйвер	Дані застосувань з транспортами, прив'язані знизу до цього драйвера + застосування, які прямо взаємодіють з проміжним драйвером	Середня	Є	+	Максимальні
Драйвер мережного пристрою	Дані всіх застосувань	Середня	Є	+	Максимальні

Модуль захисту, який виконується в режимі користувача, не може отримати доступ до системних даних інакше, як, визвавши функції, які надає ОС. Останні в свою чергу звертаються до системних сервісів, здійснивши попередню перевірку параметрів виклику.

3. Залежність вибору варіанта реалізації програмного засобу захисту від вимог до захищуваної системи. Вибір того чи іншого варіанта реалізації програмного засобу захисту інформації залежить від попередньо сформульованих початкових вимог до конкретної захищуваної інформаційної системи.

З наведеного вище аналізу мережної архітектури ОС Windows XP/2000/NT та порівняльної характеристики варіантів реалізації програмних засобів захисту інформації можна дійти висновку, що варіанти реалізації засобу захисту на рівні проміжних драйверів та драйверів мережних пристроїв є найбільш прийнятними. Спираючись на попередній аналіз, можемо співставити найчастіші вимоги, що висуваються до програмних засобів захисту з найбільш ефективними рівнями їх реалізації (табл. 2).

Таблиця 2 – Залежність вибору варіанта реалізації програмного засобу захисту від вимог до захищуваної системи

Вимоги	Рівень реалізації
1	2
Необхідно реалізувати програму безпечного обміну даними мережею	Реалізація на рівні застосування та DLL
Необхідно забезпечити безпечний обмін даними по мережі між застосуваннями, які використовують конкретний мережний API або забезпечити контроль доступу до мережного API	Реалізація на рівні системної DLL, яка надає даний API, або на рівні системного API, або на рівні відповідного драйвера файлової системи

Закінчення таблиці 2

1	2
Необхідно забезпечити контроль доступу до мережних сервісів	Реалізація на рівні мережного сервісу (яка зводиться до реалізації захисту на рівні системної DLL *rpcrt4* або системного API) або реалізація на рівні мережних драйверів для контролю за адресами
Необхідно забезпечити безпечний обмін даними по мережі між застосуваннями, які використовують конкретний транспортний протокол або забезпечити аналіз трафіка за цим протоколом	Реалізація на рівні драйвера *ndis.sys* або драйвера транспорту, або проміжного драйвера, або драйвера пристрою
Необхідно забезпечити безпечний обмін даними по мережі між будь-якими застосуваннями, або забезпечити аналіз всього трафіка в мережі	Реалізація на рівні драйвера *ndis.sys* або проміжного драйвера, або драйвера пристрою

4. Пропозиції щодо реалізації програмного засобу захисту. При створенні програмного засобу захисту для комплексу програмного забезпечення "АПР ЛКС" необхідно реалізувати наступні функціональні підсистеми:

- підсистему ідентифікації та автентифікації користувачів;
- підсистему контролю доступу;
- підсистему аудиту.

При цьому підсистему ідентифікації та автентифікації доцільно асоціювати зі штатною підсистемою ідентифікації та автентифікації ОС Windows XP/2000/NT, яка реалізується *MSGINA.DLL*, а підсистему контролю доступу та аудиту – з окремо працюючим драйвером контролю доступу, який повинен бути пов'язаний з підсистемою ідентифікації та автентифікації. Такий зв'язок необхідний тому, що контроль доступу до об'єктів до входу конкретного користувача в систему може виявитись досить нетривіальним завданням, наприклад, якщо для конкретного користувача задано ізольоване програмне середовище (фіксований перелік задач, які він може запускати), то незрозуміло як бути з задачами, які запускаються до виконання успішної процедури ідентифікації та автентифікації даного користувача. Крім того, зв'язок зі штатним модулем ідентифікації та автентифікації може служити для передачі ключової інформації, необхідної для ініціалізації персональних криптографічних модулів, призначених для шифрування трафіка або для періодичного тестування коректності роботи криптографічних функцій чи інших механізмів безпеки.

Основними компонентами пропонованого програмного засобу захисту є:

1. Драйвер контролю доступу для перехоплення файлових операцій на рівні ядра.
2. Модифікована бібліотека *GINA* (NEWGINA), побудована за принципом повного перехоплення експортованих функцій оригінальної *MSGINA* (Microsoft Graphical Identification aNd Authentication DLL for Winlogon).

При вході користувача бібліотека *GINA* повертає директиви процесу Winlogon, згідно з якими він або не повинен змінювати поточний статус процесу ідентифікації та автентифікації, або виконувати деякі дії (наприклад, примусово вигружати користувача). В залежності від значення параметра *dwOptions* у функції *WlxLoggedOutSas*( ) бібліотеки *GINA*, застосування *Winlogon* або не повинно загружати профіль вхідного користувача, або повинно виконувати загрузку того користувацького профілю, інформацію про який повернула бібліотека *GINA*. Розглянемо детальніше дві операції.

Перехоплення операцій відкриття, створення та видалення файлів. Реалізація драйвером перехоплення файлових операцій ґрунтується на офіційно не документованому компанією Microsoft механізмі перехоплення системних сервісів. Принцип функціонування драйвера базується на механізмі заміни адрес функцій, які надаються системним API. Оброблювач переривання *int2E* для виклику відповідного мережного сервісу використовує таблицю розподілу системних сервісів *KeServiceDescriptionTable*.

Під час ініціалізації драйвера у функції *DriverEntry*, після успішного створення об'єкта-пристрою, установлюються власні оброблювачі файлових операцій. Для цього спочатку необхідно отримати адресу таблиці розподілу системних сервісів:

ServiceTable = KeServiceDescriptionTable;

Знаючи індексний номер функції, яка реалізує системний сервіс, можливо за таблицею визначити адресу її початку та замінити її на адресу власного оброблювача режиму ядра:

#define SYSCALL (_function);

ServiceTable –> ServiceTable [(PULONG) ((PUCHAR) _function + 1)];*
RealCreateFile = SYSCALL (ZwCreateFile);
SYSCALL (ZwCreateFile) = (PVOID) HookCreateFile;
RealOpenFile = SYSCALL (ZwOpenFile);
SYSCALL (ZwOpenFile) = (PVOID) HookOpenFile;

При перехопленні функцій *ZwCreateFile* та *ZwOpenFile* необхідно мати на увазі, що операції запуску виконуваних файлів, перейменовування та видалення об'єктів проводиться саме цими функціями при установлених наступних флагах: *DesiredAccess & DELETE, DesiredAccess & FILE_EXECUTE, DesiredAccess & GENERIC_EXECUTE.*

Окремим завданням є розрізнення операцій з об'єктами типу PIPE або COM-порт. Це можливо зробити, використовуючи ім'я, наприклад:
memcmp (ANSIString.Buffer, "\\dosdevices\\com", 15) після виконання фрагмента коду:
UnString.Length = (USHORT) ObjectAttributes –> ObjectName –> Length;
UnString.MaximumLength = (USHORT) ObjectAttributes –> ObjectName –> MaximumLength;
UnString.Buffer = ObjectAttributes –> ObjectName –> Buffer;
RtlUnicodeStringToANSIString (&ANSIString, &UnString, TRUE);

Власні оброблювачі операції створення файла та операції відкриття файла. Власний оброблювач створення (відкриття) файла може, наприклад, перевірити права доступу поточного процесу до створюваного (відкриваного) файла, і якщо доступ заборонено, то повернути статус створення (відкриття) файлу *STATUS_ACCESS_DENIED* (доступ заборонено), або в іншому випадку повернути керування оригінальному оброблювачу. При цьому можливо вести журнал вдалих і невдалих спроб доступу, реєструючи ім'я процесу, в контексті якого здійснювалась спроба отримати доступ до ресурсу, та ім'я створюваного (відкриваного) файла.

```
NTSTATUS HookCreateFile (OUT PHANDLE FileHandle;
IN ACCESS_MASK DesiredAccess;
IN POBJECT_ATTRIBUTES ObjectAttributes;
OUT PIO_STATUS_BLOCK IoStatusBlock;
IN PLARGE_INTEGER AllocationSize, IN ULONG FileAttributes;
IN ULONG ShareAccess, IN ULONG CreateDisposition;
IN ULONG CreateOptions, IN PVOID EaBuffer, IN ULONG EaLength)
{   If (Access (ObjectAttributes));
     Return RealCreateFile (FileHandle, DesiredAccess, ObjectAttributes, IoStatusBlock;
 AllocationSize, FileAttributes, ShareAccess, CreateDisposition, CreateOptions, EaBuffer, EaLength);
     Return STATUS_ACCESS_DENIED;}.
```

Отже, досліджено метод побудови та визначені можливості інтегрування операцій захисту і програмних засобів захисту для системи автоматизованого проектування лінійно-кабельних споруд та мереж зв'язку «АПР ЛКС» на різних рівнях мережної архітектури ОС Windows XP/2000/NT системи інформаційної безпеки "АПР ЛКС" шляхом інформацій в мережну архітектуру ОС Windows XP/2000/NT. Метод може бути застосований для побудови системи захисту інформаційних ресурсів систем проектування об'єктів інформаційної діяльності та в інших сферах.

Література

1. *Потій А.* Технология проектирования систем обеспечения ИТ- безопасности // Служба безопасности. – 2002. – 2 (68). – С. 24-25.
2. *Тардаскін М.Ф., Кононович В.Г.* Аналіз захищеності інформації в автоматизованих системах управління при модернізації телекомунікаційних систем // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. – 2002. – № 5. – С. 35-41.
3. *Гладиш С.В.* Розробка політики інформаційної безпеки щодо автоматизації проектних робіт з будівництва, розширення та реконструкції інформаційно-телекомунікаційних систем // Зб. доп. III міжнародного молодіжного форуму «Інформаційні технології в XXI столітті». – Дніпропетровськ, 2005.
4. http://www.microsoft.com . Microsoft Windows NT Device Driver Kit.
5. *Центр* з проектування линійно-кабельних споруд та мереж зв'язку «Проект-Телеком» / Інформаційні матеріали семінару-наради. – Одеса: ОНАЗ ім. О.С.Попова, 2004. – 30 с.: іл.
6. *Rajeev Negar*. Windows NT File System Internals: Developer's guide. USA, O'Reilly & Associate, Inc., 1997.
7. http://www.sysinternals.com/ntdll.htm. Mark Russinovich. Inside the native API.