**Актуальність.** З часів появи перших комп'ютерів і по цей день віруси є однією з головних причин виникнення неполадок у роботі ЕОМ, просочувань інформації, в тому числі і конфіденційних даних. При цьому за весь цей час комп'ютерні віруси встигли еволюціонувати і набути нових чисельних форм. Зараз рядовому програмістові нічого не варто створити вірусну програму розробленому шаблону, тому нове шкідливе ПЗ з'являється майже кожну секунду. Проте і еволюція антивірусних програм не стоїть на місці. Методи виявлення та відвертання шкідливих дій продовжують розвиватися. Також існують сучасні реалізації захисту, що надають функціонал контролю трафіку і маршрутизації тільки дозволеним хостам окрім роботи з вірусами: наприклад фаєрвол (міжмережевий екран), VPN і так далі. Самою по собі проблеми комп'ютерних вірусів могло і не виникнути, оскільки самі собою ЕОМ спочатку були тільки у великих корпорацій, урядових організацій та подібних структур. В 1979 р. компанія Apple не випустила перший персональний комп'ютер Apple II. Після того як багато користувачів отримали своє розпорядження особисту ЕОМ, у вірусу як у типу програми з'явилося те, що саме по собі робить можливим його існування: потенційне довкілля для поширення. З того моменту було написано безліч програм з поширення та боротьби з вірусами. Неможливо не помітити, яка величезна кількість вірусів існують і створюються щохвилини у світі, а також які збитки вони наносять користувачам комп'ютерів та мережі Інтернет. На щастя, існують антивірусні програми, які хоч і не повністю, але можуть запобігти збиткам або ліквідувати наслідки. Тому користувачам однаково необхідно навчитися правильно використовувати як антивіруси, так і незнайомі файли та посилання - у будь-який момент вони можуть виявитися шкідливими. Методи виявлення та знищення вірусів на комп'ютері далеко не ідеальні і Саме в таких перегонах ефективності вірусів та антивірусів користувачі повинні навчитися за допомогою різного ПЗ забезпечувати свою безпека та безпека ПК. Останнім часом віруси настільки поширились, що на даний момент кожен може завантажити вірус добровільно для дослідження, не кажучи вже про навмисні та скеровані атаки на окремі пристрої, локальні мережі чи навіть АС. Спираючись на ці факти жоден веб-ресурс, на даний момент, не може бути безпечним для користувача і потребує додаткової перевірки, не виняток і розвиваючі повним ходом месенджери і соціальні мережі.

Із появою месенджерів, використання повномасштабних соціальних мереж витісняється на другий план. Простий інтерфейс, швидкість обробки повідомлень та їх шифрування, а також відсутність зайвої додаткової інформації про користувача – всі ці характеристики властиві месенджерам і бракують соціальним мережам. Отож аргументований та природній перехід не змусив себе чекати. Велика частина користувачів підсвідомо відмовляються від соцільних мереж і переходять на більш зручні і сучасні месенджери. Виникнення та швидкий розвиток месенджера Telegram, прив'язаного до телефонному номеру, розпочалося у серпні 2013 року. Це кросплатформовий майданчик, що дозволяє користувачам обмінюватися не лише повідомленнями текстового формату, а й медіафайлами (відео, фото, аудіо), які мають велику кількість форматів. Його особливість полягає в тому, що воно закодоване потужними корпораціями. Неймовірна швидкість роботи програми обумовлена ​​тим, що резервації файлів зберігаються в різних хмарних кластерах (віртуальних серверах), які знаходяться на території численних країн: найбільшими є США та Німеччина. Будь-які передані дані або файли шифруються складними системними алгоритмами. Згодом мережами проходять вже недоступні для хакерів пакети даних. Технологіями Telegram забезпечується шифрування і конфіденційність персональних даних. Проте актуальною залишється проблема передачі шкідливих та зловмисних даних, будь то файл, посилання чи IP адреса веб-серверу, на якому містяться шкідливі домени. Щодня через майданчик пересилається петабайти неврегульованого та потенційно зараженого трафіку, який проходить перевірку на зараження вже безпосередньо в ОС пристрою.

Під час передачі інформації дані часто втрачаються. Причиною цього можуть бути помилки в мережах, в вузлах зв’язку, обладнанні, програмні помилки (віруси, інтернет-черви, піратство, спам і т.д.). Перевірка даних та інформації в процесі обміну є важливим аспектом для зберігання безпеки програмного забезпечення, цілісності ОС та захисту особистої приватності. Спираючись на необхідність і доцільність перевірки даних під час використання месенджера Telegram метою роботи є розробити інтегрований програмний модуль для захисту інформації і розпізнавання сучасних програмних загроз за допомогою мови програмування Java, шляхом створення бота із зазначеним функціоналом і легким способом зворотнього зв’язку з користувачем.

Для досягнення даної мети необхідно розв’язання таких задач:

* Аналіз сучасних програмних загроз (вірусів, інтернет-червів) та методів захисту від них
* Аналіз сучасних можливостей і технологій Telegram, як сучасного месенджера зберігаючого конфіденційність і приватність, для захисту інформації
* Розробка і дослідження створеного програмного модуля, як способу захисту і запобігання ураження ОС

**Об'єкт дослідження:** процес виявлення зараженних даних і зловмисних ресурсів за допомогою стороннього додатку.

**Предмет дослідження:** програмний модуль захисту інформації в месенджері Telegram.

**Оцінка сучасного стану проблеми на основі вітчизняної та зарубіжної літератури.** Проблематика захисту інформації, криптографічне шифрування даних та зберігання її на серверах сучасних месенджерів дуже активно обговорються провідними спеціалістами і кваліфікованими фахівцями компаній, відповідаючих за методи забезпечення захисту інформації в месенджерах.

Прихід світового гіганту, месенджеру Signal, 2013 року спричинив собою багаторічну затяту конкуренцію із Telegram. Цей всесвітньо відомий майданчик набув свою переконливу репутацію протоколом наскрізнього шифрування всієї інформації Signal protocol. Декілька програм із закритим вихідним кодом, наприклад, такі як WhatsApp, стверджують, що реалізували цей протокол, який, за їхніми словами, шифрує розмови більше мільярда людей у ​​всьому світі. Facebook Messenger також стверджують, що вони пропонують цей протокол для додаткових «секретних сеансів зв'язку», як і Google Allo для «режиму інкогніто». На відміну від WhatsApp, який використовує як протокол end-to-end шифрування реалізацію алгоритму Double Ratchet від Open Whisper Systems (Signal), автори Viber написали його реалізацію з нуля. При цьому шифрування піддаються текстові повідомлення, дзвінки, а також файли, що пересилаються між користувачами. На відміну від своїх аналогів Telegram має індивідуальний протокол шифрування інформації MTProto. В основі протоколу лежить оригінальна комбінація симетричного алгоритму шифрування AES (в режимі IGE), протоколу Діффі-Хеллмана для обміну 2048-бітними RSA-ключами між двома пристроями та ряду хеш-функцій. Протокол допускає використання шифрування end-to-end з опціональним звірянням ключів.

В 2021 році засновник месенджера Signal Моксі Марлінспайк розкритикував Telegram. Він наголосив, що всі повідомлення, контакти та дані користувачів за замовчуванням зберігаються на серверах Telegram у вихідній формі та взагалі ніяк не шифруються. Цей факт, на його думку, ставить під сумнів безпеку месенджера, оскільки співробітники компанії та хакери можуть легко отримати всю необхідну інформацію про користувача. Попри це засновник Telegram Павло Дуров відповів, що месенджер стримує свою обіцянку і не розкриває дані користувача третім особам. Засновник месенджера послався на внутрішнє дослідження ФБР, в якому докладно описано, які дані можуть отримати правоохоронні органи США з популярних месенджерів. Згідно з документом, правоохоронці можуть мати доступ до повідомлень, надісланих через iMessage, WhatsApp і Line, але не до повідомлень, надісланих за допомогою Telegram, Signal, Viber та WeChat. Крім того, Агентство національної безпеки США протягом багатьох років брало активну участь у розробці міжнародних стандартів шифрування інформації. Виходячи з цього, представники АНБ можуть отримати доступ до всіх даних, зашифрованих загальноприйнятими методами захисту. Також альтернативні підходи до шифрування інформації зазвичай відзначаються як «ненадійні», що запобігає спробам їх використання.

**Галузь застосування.** Даний програмний модуль захисту інформації в месенджері Telegram може використовуватися в усіх сферах ІТ, адже, доки віруси існують у всіх куточках простору Інтернету, потреба швидкої перевірки на вірусне програмне забезпечення вимушено залишається актуальною.

**Новизна.** Новизна науково-технічного дослідження полягає в тому, що вперше розглядається створення інтегрованого засобу перевірки на шкідливе програмне забезпечення із подальшим зворотнім зв’язком у вигляді результатів аналізу на кросплатформенному майданчику Telegram і підключенням стороннього API VirusTotal (із великою вбудованою базою антивірусів).

**Практична цінність.** Запропонований програмний модуль може використовуватися будь-яким звичайним користувачем месенджеру прямо перед відправкою потенційно заражених даних. Крім того, сформульоване програмне рішення буде корисним експертам з інформаційної безпеки для сканування великого потоку інформації. Прикладом може слугувати корпоративна, або локальна мережа, яка веде поштове листування між хостами всередині мережі. Великий трафік у вигляді файлів, посилань, IP-адрес протягом робочого дня може скануватися за допомогою розробленого програмного модулю для випередження зараження всієї мережі, лише від одного зловмисного листа.

**Розділ 1**

**Проблематика захисту інформації в сучасних умовах комунікації.**

* 1. **ЕВМ і комп’ютерні дані в мережі.**
     1. **ЕВМ та інформація на комп’ютері.**

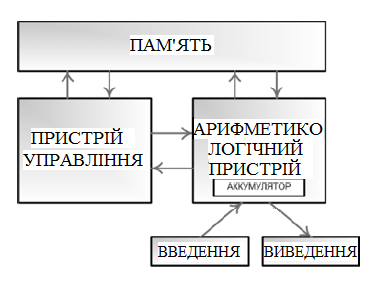
В силу вимог ЕОМ повинна була мати:

* пристрій для одержання (введення) інформації;
* пристрій для зберігання інформації (пам'ять);
* пристрій для обробки інформації (включає два компоненти: 1) арифметично-логічний пристрій, призначений для оцінки ситуації і 2) пристрій управління, призначений для вибору способу поведінки);
* пристрій для виведення результатів обробки інформації.

Робота комп'ютера відбувається так:

* пристрій введення інформації приймає сигнал і перетворює його на пристрій обробки інформації;
* пристрій обробки інформації (арифметично-логічний блок) перетворює сигнал на машинно-читаний код, який передається в блок управління;
* блок управління розділений на осередки, у кожному з яких «перебуває» певна команда, здатна робити дії з інформацією;
* отриманий сигнал активізує команду з першого осередку, яка може виконувати будь-яку з наступних операцій:
  + виконання логічних чи арифметичних операцій за допомогою арифметично-логічного блоку;
  + читання з пам'яті даних виконання арифметичних чи логічних операцій;
  + запис результатів на згадку;
  + введення даних із зовнішнього пристрою на згадку;
  + виведення даних із пам'яті на зовнішній пристрій.
* після виконання команди з першого осередку активізується наступний осередок з іншою командою. Однак, цей лінійний порядок може бути змінений за допомогою команд передачі управління (переходу). Вони змушують пристрій управління активізувати комірки не одну за одною, а переходити туди, де знаходиться необхідна для вирішення цього завдання команда.
* після виконання останньої команди результати обробки інформації виводяться на зовнішній пристрій.

Вочевидь, що можливості машини загалом визначаються кількістю осередків у блоці управління та обсягом команд. Чим більше комірок – тим більше команд, тим більше завдань може вирішувати машина. Наприклад, у калькуляторі кількість осередків невелика, команд із обробки інформації там небагато. Тому калькулятор може працювати з числовими даними, але не здатний виконувати операції з текстовими або візуальними даними.



Необхідно відзначити, що перед творцями перших обчислювальних машин стояла проблема – чи пристрій для зберігання інформації (пам'ять) і пристрій для її обробки перебувати в різних апаратах, або їх об'єднати. У кожного рішення були свої позитивні та негативні якості.

Поділ пам'яті та управління дозволяв спростити програмування, знижував можливість збоїв, полегшував роботу користувачів. На першій машині, в якій пам'ять та управління розділялися («Марк I», розробленої в Гарвардському університеті), дані зберігалися в електромеханічних пристроях, а програми запроваджувалися за допомогою перфораційних стрічок. Зрозуміло, операції з даними та операції з програмами доводилося здійснювати по-різному. Однак це спричинило збільшення розмірів машини і поява додаткових складнощів у її обслуговуванні. Оскільки ідею було висунуто викладачами Гарвардського університету (головний розробник – Говард Ейкен), то принцип поділу «пам'яті» і «програми» у межах обчислювального пристрою отримав найменування «гарвардської архітектури». Навпаки, група вчених Прінстонського університету, членом якої був відомий американський математик угорського походження Джон (Янош) фон Нейман, вважала, що пам'ять і програми повинні знаходитися в одному апаратному пристрої. Тобто команди та числа, які за допомогою цих команд обробляються, знаходяться в однакових осередках, які послідовно обробляються. Це, у свою чергу, вимагало, щоб як дані, так і команди з їх обробки, писалися тією ж мовою, причому і над програмами, і над даними можна було виконувати одні й ті самі операції. Цей принцип, що називається «принципом єдності пам'яті», став характерною ознакою «архітектури фон Неймана». Вразливим місцем неймановской архітектури було запровадження даних. Пристрій Ейкена могло приймати дані у звичній для людини десятковій формі обчислення. За такої форми всі числа описуються за допомогою десяти різних знаків – від 0 до 9. Архітектура Неймана цього не дозволяла. Але для вирішення проблеми групою Неймана було знайдено оригінальний шлях, який, зрештою, і визначив майже всюди перемогу «нейманівської архітектури», а також привів до появи комп'ютера в традиційному розумінні.

Цей шлях був у використанні двійкового коду. При двійковому коді всі числа записуються за допомогою двох знаків – 0 та 1. У двійковому коді нулю відповідає 0, одиниці – 1, а ось число «2» у двійковому коді вже пишеться як «10». Істотною перевагою двійкового кодування є те, що воно дозволило «матеріалізувати» інформацію, перевести її з області абстрактного сигналу в область «матеріалу», який може бути оброблений за допомогою технічних засобів. Сигнал або є (1) або його немає (0). "Пам'ять" комп'ютера складається з величезної кількості елементарних електромагнітних пристроїв, які або випромінюють електричні імпульси (що для пристрою обробки є 1), або не випромінюють (це дає 0). Якщо сигнал змінюється з 1 на 0, це означає зменшення числа, тобто віднімання, а якщо з 0 на 1 – це додавання. Звичайно, запис навіть невеликого числа на кшталт 32 та арифметичних операцій з ним у двійковому коді виглядає дуже громіздко, але швидкість зчитування сигналів урівноважувала всі недоліки. Спочатку елементарні пристрої були помітними розмірами, і зчитування сигналів з них було відносно довгим. Але широке використання напівпровідників, мікросхем тощо дозволили кардинально зменшити розміри обчислювальних пристроїв і одночасно підвищити їхню швидкодію. Тим не менш, аж до теперішнього часу всі комп'ютери, як і їхні далекі попередники, можуть тільки складати і віднімати числа, записані в двійковому коді, але роблять це швидше. Слід зазначити, що використання двійкового коду далеко не відразу було визнано єдиним вірним рішенням. Як мовилося раніше, були види систем, що у десятковій системі. Можна також згадати оригінальний експеримент, здійснений у СРСР при створенні обчислювального пристрою, що працює в троїчному коді (машина "Сетунь"(рис 1.1)).

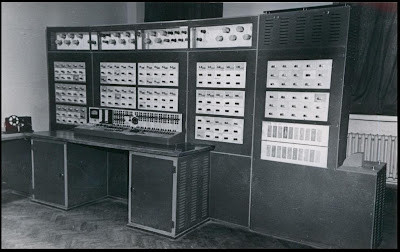


Рис 1.1. Машина "Сетунь"

Завдяки двійковому коду з'явилася як можливість апаратної обробки інформації. Її стало можливим виміряти. Елементарною одиницею інформації є біт. Він означає або відсутність сигналу (0 біт), або наявність сигналу (1 біт). Відповідно, 1 у двійковому коді – 1 біт інформації.

Але якщо комп'ютер як апаратний пристрій оперує саме бітами, то людина як користувач має користуватися звичними для неї символами. Досвідченим шляхом було встановлено, що всі символи можуть бути описані для комп'ютера числами двійкового коду, що не перевищують 8 розрядів (всього 256 чисел). Інакше кажучи, символ (буква, пробіл, цифра) можна описати за допомогою 8 біт. Ця кількість інформації одержала найменування "байт".

Слід зазначити, що утворення кратних одиниць виміру кількості інформації дещо відрізняється від прийнятих у більшості наук. Традиційні метричні системи одиниць як множники кратних одиниць використовують коефіцієнт 10n, де n = 3, 6, 9 і так далі, що відповідає десятковим приставкам Кіло (), Мега (),), Гіга (),) і так далі. Оскільки комп'ютер оперує числами над десятковою, а двійковій системі числення, в кратних одиницях виміру кількості інформації використовується коефіцієнт 2n. Тому кратні байту одиниці виміру кількості інформації вводяться так:

1 Кбайт = байт = 1024 байт

1 Мбайт = Кбайт = 1024 Кбайт

1 Гбайт = Мбайт = 1024 Мбайт

1 Тбайт = Гбайт = 1024 Гбайт

* + 1. **Дані в мережі Internet.**

Ми опишемо два важливі архітектурні типи — еталонні моделі OSI та TCP/IP. Незважаючи на те, що протоколи, пов'язані з еталонною моделлю OSI, зараз не використовуються, сама модель досі дуже актуальна, а властивості її рівнів, які будуть обговорюватися в цьому розділі дуже важливі. У еталонній моделі TCP/IP все навпаки: сама модель зараз майже не використовується, а її протоколи є чи не найпоширенішими. Модель OSI являється еталонною (за винятком фізичного середовища). Ця модель заснована на розробці Міжнародної організації зі стандартизації (International Organization for Standardization, ISO) і є першим кроком до міжнародної стандартизації протоколів, що використовуються на різних рівнях.

Модель OSI має сім рівнів. Поява саме такої структури була обумовлена ​​такими міркуваннями.

1. Рівень повинен створюватися за необхідності окремого рівня абстракції.

2. Кожен рівень має виконувати строго певну функцію.

3. Вибір функцій кожного рівня має здійснюватися з урахуванням створення

стандартизованих міжнародних протоколів.

4. Межі між рівнями повинні вибиратися так, щоб потік даних між

інтерфейси були мінімальним.

5. Кількість рівнів має бути достатньо великою, щоб різні функції не об'єднувалися в одному рівні без необхідності, але не надто високим, щоб архітектура не ставала громіздкою

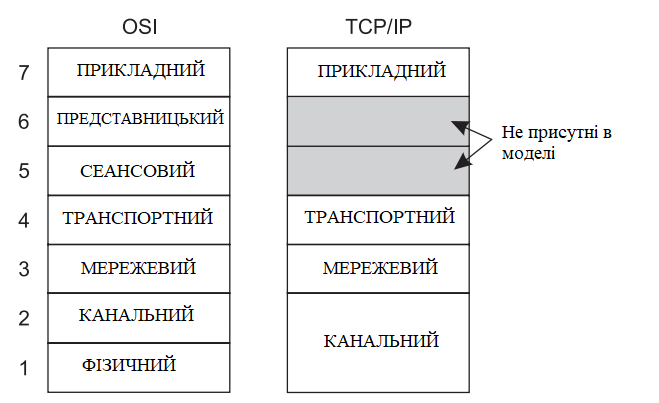
Модель OSI, включає сім рівнів:

* фізичний рівень;
* канальний рівень;
* мережний рівень;
* транспортний рівень;
* сеансовий рівень;
* представницький рівень ;
* прикладний рівень.

Розглянемо еталонну модель TCP/IP, що використовувалася в комп'ютерній мережі Інтернет. Коли згодом з'явилися супутникові мережі та радіомережі, виникли великі проблеми при поєднанні з ними інших мереж за допомогою наявних протоколів. Знадобилася нова еталонна архітектура. Таким чином, можливість об'єднувати різні мережі в єдине ціле була однією з головних цілей із самого початку. Пізніше ця архітектура отримала назву еталонної моделі TCP/IP, відповідно до своїх двох основних протоколів.

Модель TCP/IP, включає чотири рівні:

* канальний рівень;
* мережевий рівень;
* транспортний рівень;
* прикладний рівень.



Різні рівні протоколу мають різні імена для пакетів даних, які називаються сегментами на транспортному рівні, IP-пакетами на мережному рівні та кадрами на канальному рівні. Дані в TCP або UDP знаходяться на рівні програми. Дані інкапсулюються в кадр і відправляються в середовище передачі. Після досягнення хоста призначення відповідний заголовок кожного рівня протоколу видаляється, і, нарешті, дані рівня програми передаються прикладної програми для обробки.

Дані подорожують по мережі у формі пакетів, кожен з яких складається із заголовка та корисного навантаження. Заголовок містить відомості про те, звідки прибув пакет і куди він прямує. Заголовок, крім того, може включати контрольну суму, інформацію, характерну для конкретного протоколу та інші інструкції з обробки. Корисне навантаження - це дані, що підлягають пересиланню. У тих низькорівневих апаратних засобів пакети часто називають кадрами. Коли пакет подорожує вниз стеком протоколів, готуючись до відправки, кожен протокол вводить у нього власну інформацію заголовка. Закінчений пакет одного протоколу стає корисним навантаженням пакета, що генерується наступним протоколом. Ця операція відома як інкапсуляція або оформлення. На приймаючій машині інкапсульовані кадри розвертаються у зворотному порядку. Наприклад, UDP-пакет, що передається по мережі Ethernet, містить три різні "обгортки". У середовищі Ethernet він "загортається" в простий заголовок, що містить відомості про апаратні адреси джерела та одержувача, довжину кадру та його контрольну суму. Корисним навантаженням Ethernet-кадра є IP-пакет. Корисне навантаження IP-пакета UDP пакет, і, нарешті, корисне навантаження UDP-пакета складається власне з даних, що передаються. ІР-пакет вкладається в кадр Ethernet і саме в нього вкладаються пакети UDP, TCP. IP-протокол пропонує ненадійне транспортне середовище. Ненадійну у тому сенсі, що немає гарантії благополучної доставки IP-дейтограми. Алгоритм доставки в рамках даного протоколу простий: при помилці дейтограма викидається, а відправнику надсилається відповідне ICMP-повідомлення (або не надсилається нічого). Забезпечення ж надійності покладається на більш високий рівень (UDP чи TCP).

Нижче на рис 1.2 описані дії, які виконуються на кожному рівні:

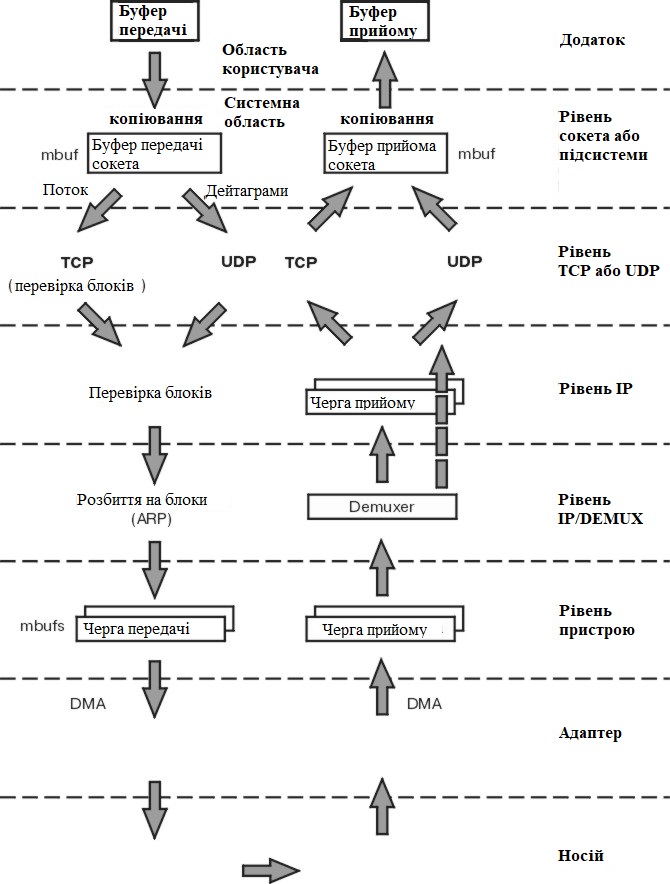


Рис 1.2. Процес передачі пакета

* Після того як програма відправляє запит на запис даних, дані копіюються з робочого сегмента програми буфер передачі сокета.
* На рівні сокету чи підсистеми викликається служба UDP чи TCP.
* Операційна система підтримує кластери різного розміру. Оптимальний розмір вибирається, коли:
* UDP копіює дані в буфер сокету та обчислює контрольну суму.
* TCP копіює дані в буфер сокету.
* Якщо розмір даних перевищує розмір максимального блоку передачі (MTU) локальної мережі, то:
* TCP розбиває дані на сегменти розмір яких не перевищує розміру MTU.
* У разі UDP фрагментація даних виконується на наступному рівні – рівні IP.
* При необхідності IP розбиває дані на блоки таким чином, щоб розмір вихідних пакетів не перевищував обмеження MTU.
* Пакети розміщуються в чергу виведення пристрою і передаються адаптером локальної мережі системі-одержувачу. Якщо черга виводу пристрою переповнена, пакет відкидається.
* Пакети, що надходять, поміщаються в чергу прийому драйвера пристрою і передаються рівню IP через інтерфейсний рівень.
* Якщо рівень IP в системі, що приймає, визначає, що рівень IP в передавальної системі розбив дані на фрагменти, він збирає фрагменти у вихідний блок даних і передає його рівню TCP або UDP.
* TCP за необхідності об'єднує сегменти даних і поміщає дані в буфер прийому сокету.
* UDP передає отримані дані буфер прийому сокету. Якщо буфер прийому сокету (udp\_recvspace) переповнений, пакет відкидається.
* Коли програма надсилає запит на читання, необхідні дані копіюються з буфера прийому сокета, розташованого в сегменті ядра, до буфера програми.
  1. **Аналіз одночасного існування та сучасного протистояння вірусів та антивірусів.**
     1. **Віруси. Їх розвиток і класифікація.**

Ідея та концепція механічної структури, здатної до самовідтворення, активації, захоплення і мутації, було виведено ще 1959 р. американським ученим Л.С.Пенроузом, після чого Ф.Г. Сталь як дослідження реалізував цю ідею при допомозі машинного коду на IBM 650. Самою по собі проблеми комп'ютерних вірусів могло і не виникнути, оскільки самі собою ЕОМ спочатку були тільки у великих корпорацій, урядових організацій та подібних структур, так як складність самої по собі ЕОМ робила її вкрай дорогим задоволенням для звичайної людини, поки в 1979 р. компанія Apple не випустила першого персональний комп'ютер Apple II. Після того як багато користувачів отримали своє розпорядження особисту ЕОМ, у вірусу як у типу програми з'явилося те, що саме по собі робить можливим його існування: потенційна довкілля та поширення. З того моменту було написано безліч програм з поширення та боротьби з вірусами. Багато хто з них став свого роду легендами. Як говорилося раніше, еволюція у світі інформаційних технологій не стоїть на місці та різних видів шкідливих програм стало значно більше, внаслідок чого виникла потреба у систематизованій класифікації цих програм.

В таблиці 1.1 вказується класифікація вірусів.

*Таблиця 1.1*

|  |  |
| --- | --- |
| Шкідливість | • Нешкідливі — такі програми просто  здатні поширюватися в мережі, переходячи з одного  ПК на інший, але при цьому не виконують будь-які  деструктивні функції щодо системи.  • Небезпечні — шкідливі програми, здатні перевантажувати  пам'ять ПК або генерувати звукові сигнали, зображення.  • Небезпечні — програми, здатні завдавати шкоди системі.  • Вкрай небезпечні — віруси, здатні знищувати дані,  що знаходяться в різних сегментах та секторах пам'яті,  призводити до поломки механічних частин ПК |
| Середовище існування | • Файлові — ураження файлів, що виконуються, середовищем. Існування є відповідно в COM і EXE файлах.  • Завантажувальні — ураження секторів завантаження (Boot секторів)  жорстких дисків чи секторів системного завантажувача.  • Мережеві — ураження комп'ютерних мереж та систем.  • Макро — ураження файлів Microsoft Office. |
| Спосіб зараження | • Резидентні віруси – віруси, що залишаються в оперативній пам'яті  після виконання роботи інфікованої програми. Після перезавантаження системи будуть видалені з пристрою, якщо код програми не закладено функцію автозапуску, за такої умови  Лише відбудеться повторне інфікування системи.  • Нерезидентні віруси – віруси, які не займають оперативну пам'ять пристрою та виконувані лише одноразово під час виконання вірусної програми |
| Особливості  алгоритму роботи | • Companion-віруси – ураження ЕХЕ-файлів, при якому створюється двійник СОМ-файлу, після чого спочатку виконується файл із вірусом, потім файл самої програми. Вміст ЕХЕ-файла за такого алгоритму не змінюється.  • Worms-віруси — розповсюдження в мережі шляхом обчислення адрес інших підключених до цієї мережі пристроїв та надсилання власних копій на ці пристрої.  • Віруси-паразити — віруси, здатні змінювати вміст файлів та сегментів пам'яті заражених пристроїв.  • Stealth-віруси – віруси, здатні перехоплювати звернення дискової операційної системи (DOS) до уражених дільницям диска та підставляти незаражені сегменти пам'яті. Крім іншого, такі віруси здатні обманювати внутрішні резидентні монітори системи.  • Віруси-поліморфи - віруси, що не мають повторюваних частин. У зв'язку з цим такі віруси дуже важко виявити.  • Макровіруси - віруси, що вражають макроси у файлових редакторів, таких як Microsoft Word та Microsoft Excel.  • Віруси, здатні до самошифрування, - віруси, здатні змінювати свій програмний код.  • Віруси з нестандартним алгоритмом – віруси, що мають свої власні сигнатури та структурні алгоритми,  сильно утруднюючі виявлення вірусу |

* + 1. **Класифікація антивірусів. Методи виявлення вірусів.**

Виділяють два основних методи виявлення комп'ютерних вірусів антивірусами :

1. *Виявлення вірусів за «словником».* Це досить простий спосіб. Антивірус просто сканує всі файли та програми та порівнює їх зі словником, куди внесено існуючі віруси. Якщо є збіг, то антивірус видалить шкідника або внесе до карантину. Зрозуміло, щоб цей метод справлявся зі своїм завданням, необхідно оновлювати словник та вносити до нього нові шкідливі програми.
2. *Виявлення вірусів щодо поведінки програм.*

Антивіруси, що працюють за цим принципом, відстежують те, як поводяться програми та які дії вони виконують. В основному вся підозріла активність програм зводилася до запису нових даних у файл, що виконується, але тепер і звичайні програми нерідко роблять те саме. Як наслідок, користувач отримує багато хибних попереджень, коли антивірус знову приймає невинний файл за шкідливий. Не дивно, що цей спосіб застосовується все рідше і рідше. При цьому антивірусне програмне забезпечення можна розділити на кілька видів, у яких все ж таки кілька відрізняється функціонал:

• Детектори. Вони знаходять проблему та «лікують» її, використовуючи метод словника. До цього виду відносять банальні та всім відомі Doctor Web, антивірус Касперського.

• Фільтри. Такі антивіруси стежать за диском. При спробі будь-якої програми записатися на нього фільтр повідомить користувачеві про це та запросить у нього дозвіл на здійснення операції. Таким чином можна боротися і з новими невідомими вірусами, якщо, звичайно, вони взаємодіють із диском, а нез BIOS.

• Вакцинатори. Даний вид антивірусу використовується тільки для боротьби з конкретними відомими шкідливими програмами, оскільки вакцинатору необхідно взяти ознаки вірусу. Потім він їх записує в безпечну програму користувача, та вірус вважає, що вона вже заражена.

• Ревізори. Ревізори зберігають у собі інформацію про станпрограм та файлів, а при повторному скануванні використовують їх для порівняння та аналізу змін. Перевіряється безліч факторів: від розміру файлів та часу їх створення до стану BOOT-сектору. Проте сам антивірус не визначає, шкідливий перед ним файл чи ні. Він передає всі дані про зміни користувачеві, який вже сам повинен вирішити, що спричинило їм причину. Якщо це, на думку людини, вірус, тоді вже ревізор видаляє небезпечні дані або поміщає їх у карантин.

* + 1. **Дослідження проблеми мережевих перехоплювачів на антивірусах.**

Мережеве перехоплення - це процес, що виробляється за допомогою "атаки посередника". Особливе ПЗ перенаправляє на себе шифроване з'єднання користувача з будь-яким сайтом, на який надійшов запит, і прикидається їм. Потім перехоплювач відкриває нове з'єднання з початковим веб-ресурсом і пропускає дані через себе між двома з'єднаннями. Оскільки той, хто перехоплює трафік, отримує доступ до більшості даних в рамках з'єднання, він може вважати, поміняти і заблокувати будь-який контент, що передається або отримуваний клієнтом. Це може використовуватися як у хороших (блокування шкідливих сайтів), так і в поганих цілях (шахрайство, злом пристроїв). Взимку 2017 р. представниками Google, Mozilla, Cloudflare і кілька університетів були різко розкритиковано процеси перехоплення HTTPS-трафіку антивірусами та мережевими фільтрами. Завдяки зробленому дослідженню з'ясували, що мережевий перехоплення HTTPS-трафіку антивірусними програмами може нести загрозу для безпеки користувачів та їх підключення до всесвітньої мережі. Як правило, це ПЗ не може отримати доступ до HTTPS-пакетів, проте антивірусні компанії знайшли спосіб для аналізу даних, що йдуть зашифрованим підключенням: вони стали встановлювати власні кореневі сертифікати на пристрій користувача, що значно знижує безпеку підключення. Більш того, аналіз показує, що сканери трафіку, представлені в деяких антивірусах, за рахунок своїх недоробок мають ще більше вразливостей. Перехоплені підключення використовують слабкі криптографічні алгоритми і користуються до цього зламаними шифрами, які можуть дозволити проводити атаки на пристрої та розшифровувати підключення. Таким чином, як мінімум, близько 10% трафіку перехоплюється не лише антивірусами, а й стороннім. ПЗ, яке користується ним, з легкістю розшифровує та аналізує у своїх цілях. Саме тому антивірусним компаніям варто задуматися про новий спосіб збирання інформації. Але лише півбіди полягає в тому, що перехоплення HTTPS-пакетів знижує захищеність користувача та його даних у мережі. Інша проблема - це те, наскільки часто зустрічається мережевий перехоплення.

Були оцінені робочі процеси інтернет-магазину, сайту Cloudflare та серверів оновлення Firefox. Розглядали, скільки браузерного трафіку вони перехоплюють. І результати, у свою чергу, показали, що перехоплюється від 4 до 10% трафіку, при цьому 4% - це сервери Firefox, а 10% - Cloudfare. Це чимало, але необхідно пам'ятати, що частина перехоплень виконується не зловмисниками. Якщо розбити перехоплені HTTPS-пакети по операційним системам, то з'ясовується, що з Windows перехоплюється набагато частіше, ніж з MacOS чи Linux. А трафік із мобільних пристроїв (Android та IOS) перехоплюється рідше, ніж з ОС для ПК, але не у випадку із серверами Firefox. Як не дивно, там найчастіше перехопленням займаються мобільні провайдери. Можливо, це походить від того, що настільна версія Firefox використовує окреме сховище для кореневих SSL-сертифікатів, що зменшує можливість даних бути перехопленими. На цей момент це один із маневрів, який може бути тимчасовим вирішенням проблеми перехоплення HTTPS-трафіку. Але мінус маневру у тому, що його надає власник сервера та запитаного інтернет-ресурсу, а не антивірус, що виступає ініціатор перехоплення. Далеко не кожен сайт може зробити це. Таким чином, ми маємо можливість лише напевно визначити масштаб шкоди, що завдається користувачеві мережевими перехоплювачами, і лише трохи ліквідувати його, адже все залежить від великої кількості факторів: використовуваних ПЗ та з'єднань, запитуваного сайту, пристрою користувача та ОС на ньому. Але при цьому абсолютно точно неможливо цього уникнути, поки виробники антивірусів не еволюціонують до менш уразливого способу контролю HTTPS трафіку.

* 1. **Аналіз методів шифрування та хешування інформації.**
     1. **Шифрування.**

Шифрування є універсальним засобом захисту у цьому контексті, воно застосовується повсюдно: протоколи безпечного обміну даними, конфіденційна інформація у базах даних (БД), цифрових підписах та ін. Для забезпечення захисту коду, що виконується, шифрування застосовується для протидії дизассемблювання та модифікації. Очевидно, що реінжиніринг зашифрованого коду неможливо і, як наслідок, модифікувати певні значення у виконуваному файлі так само важко. Шифрування перестає бути ефективним методом, коли застосовується трасування програми під налагоджувачем. Тоді щоб програма працювала коректно, їй потрібно буде розшифрувати код. Можливо, у порушника піде чимало часу на трасування, але відстеживши момент дешифрування, захист буде вважатися подоланим. Зважаючи на це, слід враховувати деякі особливості в реалізації шифрування виконуваного коду:

* алгоритм шифрування, що впроваджується, може бути як симетричним, так і асиметричним. Існують різні думки, яка із систем краще підходить для захисту виконуваного коду. При цьому використання двоключових криптосистем у разі успішної спроби аналізу логіки роботи захисного механізму не дозволить модифікувати код, оскільки зміни необхідно впровадити у зашифрованому у код, а закритий ключ недоступний порушнику.

У такому разі зловмисник спробує знайти відкритий ключ, який використовується при дешифруванні коду; згенерує нову пару ключів та замінить код та відкритий ключ у додатку. Виконавши ці дії, система захисту заснована на криптографії, буде подолана. Варто взяти до уваги продуктивність та складність реалізації одно- та двоключових алгоритмів, так як вони не рівнозначні;

* алгоритм шифрування, що впроваджується, можна комбінувати з методом обфускації (заплутування) для підвищення витраченого часу під час використання порушником трасування.

Посилити захист здатні:

* багатопрохідне розшифрування коду; можлива реалізація з двома та більше ключами для дешифрування чи використання майстер-ключа;
* динамічне шифрування, що є частиною динамічного захисту, здатне виконувати шифрування програми в ОЗУ безперервно, а також модифікувати виконуваний файл.

Недоліки методу:

* існує проблема зберігання ключів;
* спосіб малоефективний при трасуванні програми.

У захисті коду, що виконується, можна застосовувати методи стеганографії. Наприклад, для приховування ключової (або іншої) інформації, необхідної для дешифрування коду, замаскований під код, що виконується. Очевидно, адже якщо прописувати ключі в ресурсах або сегменті даних програми, де розташовані константи та інша інформація, то знайти ключ досить легко. Але знайти прихований ключ в виконуваному файлі вже непросто. Можна також приховувати можна частини коду, що захищається, але слід пам'ятати, що контейнер повинен бути в багато разів більше прихованих даних. Недолік методу полягає в тому, що він дозволяє приховувати лише невеликі фрагменти інформації.

Способи шифрування даних:

* Симетричне шифрування. Симетричне шифрування, також відоме як шифрування з секретним ключем, виконується з використанням одного і того ж ключа для шифрування та розшифровки інформації. Секретний ключ також відправляється одержувачу разом із секретним текстом, та за допомогою цього ключа виконується перетворення зашифрованого тексту на звичайний текст.
* Асиметричне шифрування. В асиметричному шифруванні, також відомому як шифрування з відкритим ключем, є 2 різних ключа, відкритий і закритий ключі, на відміну від симетричного алгоритму. Ключ розшифровки та шифрування відрізняється від симетричного ключа шифрування. Ключ до шифру – це відкритий ключ, а дешифратор – закритий ключ. Текст, зашифрований за допомогою відкритого ключа може бути розшифрований тільки за допомогою ключа, що належить цьому користувачеві. Це робить його набагато більше ефективнішим, ніж симетричне шифрування.

Основні методи симетричного шифрування:

* DES (стандарт шифрування даних). Це свого роду стандарт шифрування даних, прийнятий міжнародними стандартами після того, як він був розроблений в IBM в 1970 року. Зашифрований текст ділиться на частини, і кожна частина шифрується незалежно один від одного, так що шифрування виконується в певних блоки. При цьому ті самі операції виконуються над блоками для відкриття зашифрований текст. Цей метод шифрування створює новий ключ, специфічний для цього використання, щоразу, коли він використовується. Він використовує 54-бітний метод шифрування, незважаючи на 64-бітні довгі ключі. Отже, безперешкодна передача інформації та таємність були нашою найважливішою передумовою.
* Шифр RC4. Алгоритм шифрування, розроблений у 1987 році, також відомий як "Rivest Cipher 4". Його безпека залежить від використання випадкового ключа, хоча зазвичай використовується ключ довжиною 5-16 байт. Він сприймає послідовність, що підлягає шифруванню, як поточна послідовність. Основна логіка алгоритму заснована на випадковій перестановці. У той час як висока швидкість вважається однією з його сильних сторін, одноразове використання ключа алгоритму є слабкою характеристикою.
* Потрійний DES (TDES). Це алгоритм шифрування, розроблений IBM у 1978 році. Він працює з методом шифрування методом шифрування Des 3 рази та з використанням різних ключів. Саме тому він вважається ефективної формою шифрування. Два або три ключі по 112 або 168 біт виконуються один за одним. Це двонаправлений алгоритм. Дані зберігаються у зашифрованому вигляді та можуть бути викликані у час. Але недоліком цього алгоритму є повільна робота. Він використовується в інтернет-протоколі та банківських транзакції.
* AES (розширений стандарт шифрування). Він був розроблений у 2000 році двома бельгійськими криптографами з використанням блокового шифру Rijndael. Для цього алгоритму шифрування використовуються ключі довжиною 128,192 і 256 біт, які надійніші, ніж DES. Дані використовуються в алгоритмі шляхом їх поділу на матриці (4x4), звані станами. Коли шифрування AES завершено, окремі цикли виділяються для ключів довжиною 128, 192 та 256 біт. Згідно алгоритму DES, він простий у застосуванні і вимагає менше пам'яті, що є однією з його сильних сторін.

Основні методи асиметричного шифрування:

* RSA. Він був розроблений у 1978 році Роном Рівестом, Аді Шаміром та Леонардом Адлеманом і отримав свою назву від ініціалів людей, які його розробили. RSA обробляє як шифрування, і дешифрування. Цей алгоритм шифрування можна використовуватиме забезпечення конфіденційності та цифрового підпису. Принцип роботи заснований на модульній арифметиці великих чисел. Утворюється ціле число, утворене множенням двох дуже великих простих чисел. Хоча розмір номера забезпечує перевагу з точки зору надійності, він є одним з недоліків повільності транзакцій. Алгоритм RSA складається з 3 кроків, відомих як генерація ключа, шифрування та дешифрування.
* DSA (алгоритм цифрового підпису). Він був введений як стандарт цифрового підпису Національним інститутом стандартів та технологій (NIST) у 1991 році. DSA, який являє собою алгоритм шифрування з відкритим ключем, такий як RSA не шифрує з використанням закритих ключів і не розшифровує з використанням відкритих ключів, на відміну RSA. Для створення цифрової підписи, що складається з 160-бітного числа, використовується унікальна математична операція. Оскільки отримано хеш повідомлення, яке має бути передане повідомлення, зашифроване за допомогою закритого ключа, вказаного відправником, створює цифровий підпис. З відкритим ключ, що посилається на закритий ключ, це доводить, що повідомлення надійшло від правильного одержувача.
* Діффі-Хеллман. Цей алгоритм, перший алгоритм з відкритим ключем, був включений в статтю Діффі та Хеллмана 1976 року «Нові напрямки в криптографії». Він ґрунтується на компонентах, які ніколи не передаються безпосередньо. Метою цього алгоритму є забезпечення безпечної передачі ключа між двома користувачами та розгадування зашифрованих повідомлень ключа. За допомогою цього алгоритму зміна та розподіл ключів у алгоритмах симетричного шифрування вирішуються значною мірою.
  + 1. **Хешування.**

Стиснення ділянок виконуваного коду або весь код – досить поширений спосіб захисту. Функція стиснення перетворює вихідні дані до іншого виду, дизассемблировать які немає сенсу. Хороший ступінь захисту приносить комбінування методів стиснення різними пакувальниками за умови, що пакувальник не є одним з популярних (наприклад, UPX або ASPack) і добре відомих зловмисникам, а написаний розробником самостійно. Такий захист ускладнить аналіз та зменшить розмір програми. Недолік методу - популярні функції стиснення добре відомі багатьом зловмисникам;

Основне завдання використання хеш-функцій у системі захисту коду – захист від модифікації. Хешування працює у динамічному режимі. Принцип роботи полягає у наступному. Шляхом обчислення контрольних сум (КС) та виконуючи порівняння їх з еталонами, прошитими у програмі, забезпечується контроль цілісності (КЦ) ділянок коду, що захищаються, або його цілком. КЦ виробляється не лише при перезапуску програми, але й при безперервному контролі захищених ділянок пам'яті у ОЗУ. Варто зазначити, що обійти захист із використанням простих функцій (CRC або FNV та ін) досить просто, тому варто використовувати MD, SHA, SIMD або написати власну хеш-функцію.

Недоліки методу:

* проблема зберігання КС;
* популярні хеш-функції добре відомі багатьом зловмисникам.

Наприкінці слід зазначити, що це криптографічні методи не захищають додаток від динамічного аналізу, незважаючи на свою ефективність у боротьбі з дизасемблюванням. Для цього систему захисту програми необхідно доповнити методами захисту від налагодження.

Основні способи хешування інформації:

* алгоритми сімейства MD. Включають в себе алгоритми MD2, MD4, MD5, HAVAL. Ці алгоритми хешування були розроблені всесвітньо відомим криптологом Рональдом Рівестом при участі RSA Data Security. Ці алгоритми були розроблені послідовно. Усі три створюють 128-розрядні хеш-значення. Виконується доповнення вхідних даних до розміру, кратного 16 байтам. Потім обчислюється 16-байтна контрольна сума повідомлення, яка також доповнює повідомлення перед його подальшою обробкою. Всі три, як відомо, мають слабкі місця і повинні використовуватися лише у випадках, коли це необхідно для сумісності.
* алгоритми сімейства SHA. Включають в себе підсімейства алгоритмів SHA-1, SHA-2. Обидва були розроблені Національним інститутом стандартів та технологій (NIST) та Агентством національної безпеки (NSA). SHA-1 алгоритм був розроблений для використання з DSA (алгоритмом цифрового підпису) або DSS (стандарт цифрового підпису). Цей алгоритм створює 160-розрядне хеш-значення. SHA-2 алгоритм має чотири варіанти: SHA-224, SHA-256, SHA-384 та SHA-512, які називаються відповідно до кількості бітів у вихідних даних.
  1. **Ураження захисту локальної мережі.**

Розглянемо навмисні загрози безпеці інформаційних процесів у локальних обчислювальних мережах автоматизованих комплексів захисту (ЛВС АКЗ), не розглядаючи загрози, зумовлені природними явищами та надійністю програмних та апаратних засобів ЛОМ. Виходячи з класифікаційної основи, загрози інформаційних процесів в АКЗ, включають:

* загрози доступу (проникнення) в операційне середовище та виконання деструктивних дій із використанням штатного програмного забезпечення (засобів операційної системи або прикладних програм загального застосування);
* загрози створення персоналом позаштатних режимів роботи програмних (програмно-апаратних) коштів за рахунок навмисних змін службових даних, ігнорування передбачених у штатних умовах обмежень на склад та характеристики оброблюваної інформації, спотворення (модифікації) самих даних тощо;
* загрози впровадження шкідливих програм (програмно-математичного впливу);
* комбіновані загрози, що являють собою поєднання зазначених загроз.

Можливе наступне трактування загроз інформаційних процесів у АКЗ. Загрози доступу (проникнення) в операційне середовище автоматизованого робочого місця (АРМ) поділяються на загрози безпосереднього та віддаленого доступу. Загрози безпосереднього доступу здійснюються за допомогою програмних та програмно-апаратних засобів введення/виводу АРМ.

Внаслідок реалізації загроз «Відмову в обслуговуванні» відбувається переповнення буферів інформації від кінцевого обладнання АКЗ та блокування процедур обробки, «зациклювання» процедур обробки даних від пристроїв та «зависання» АРМ, відкидання пакетів повідомлень та ін. Загрози впровадження шкідливих програм (програмно-математичного впливу) характеризуються різноманіттям шкідливих програм, і, як наслідок, різноманіттям методів та наслідків їх застосування (Інфікування).

Апаратні закладки, як правило, містять мікропрограми, що забезпечують взаємодію закладки з обчислювальною системою, тому їх часто називають програмно-апаратними. Апаратні закладки можуть забезпечувати збирання та накопичення інформації, що обробляється в АКФЗ, формувати технічні канали її витоку з можливою її обробкою, та передачею за межі території, що охороняється (акваторії). При активізації апаратна закладка може ініціювати передбачені під час її розробки деструктивні дії АКФЗ. Виходячи зі специфіки АКЗ, реалізація загроз несанкціонованого доступу (НСД) до інформації в таких системах може призводити до двох видів порушення безпеки інформації: порушення цілісності (знищення,

модифікація, дезінформація); порушення доступності. Порушення цілісності інформації в АКЗ може бути викликано впровадженням у неї шкідливої ​​програми програмно-апаратної закладки або дією на систему захисту інформації чи її елементи. Крім цього, у ЛВС АКЗ можливий вплив на технологічну мережеву інформацію, яка може забезпечувати функціонування штатних засобів управління: конфігурацією мережі; адресами та маршрутизацією передачі даних у мережі; функціональним контролем мережі; безпекою інформації в мережі. Порушення доступності інформації забезпечується шляхом формування (модифікації) вихідних даних, які під час обробки викликають неправомірне функціонування, відмови апаратури або захоплення (завантаження) обчислювальних ресурсів ЛОМ АКЗ, які необхідні для виконання програм та роботи апаратури. Зазначені дії можуть призвести до порушення або відмови функціонування практично будь-яких засобів АКЗ: засобів обробки інформації; засобів введення/виводу та зберігання інформації; апаратури та каналів її передачі; засобів захисту. Реалізація будь-яким із розглянутих способів загроз безпеці

інформаційних процесів у АКЗ призводить до потенційної можливості безперешкодного проникнення зловмисників на КВО, що може призвести до значної шкоди. Це призводить до необхідності розробки досконалих механізмів захисту інформації у цих системах.

Основою для побудови базового захисту інформаційної системи підприємства може бути поєднання:

* програмно-апаратні засоби захисту інформації;
* фізична безпека, створення локальних актів ІБ.

Для захисту корпоративних локальних мереж та інформації, що передається по них, необхідно використовувати такі програмно-апаратні засоби (див. таблицю 1.2).

*Таблиця 1.2*

|  |  |
| --- | --- |
| Міжмережевий екран (фаєрвол, брандмауер) | Являє собою програмний або програмно-апаратний елемент, що здійснює контроль та фільтрацію мережного трафіку відповідно до заданих правил. |
| Система виявлення вторгнень  (Intrusion Detection System) | Програмний або апаратний засіб,  призначений для виявлення фактів  неавторизованого доступу (вторгнення або мережевий атаки) у комп'ютерну систему чи мережу.  Використовується зазвичай у комплексі із системою запобігання вторгненням |
| Система запобігання вторгненням (Intrusion Prevention System) | Програмна чи апаратна система забезпечення безпеки, що активно блокує вторгнення по мірою їхнього виявлення. Підозрювальний мережевий трафік може бути автоматично перекритий, у разі виявлення вторгнення в мережу автоматично, а повідомлення про це негайно надіслано адміністратору. |
| Антивірусне програмне забезпечення | Є основним рубежем захисту для  більшості сучасних підприємств.  Ефективним рішенням є паралельне  використання кількох антивірусів, у яких реалізовані різні методи виявлення шкідливого ПЗ |
| Віртуальні приватні мережі (VPN) | Організація приватної мережі поверх публічної, коли користувач має доступ до локальної мережі не маючи фізичного приєднання до неї |

* 1. **Автентифікація користувача.**

Одним із головних елементів будь-якої системи захисту від несанкціонованого доступу (НСД) є елемент, що забезпечує контроль доступу до ЕВМ та контроль роботи в них. Цей елемент захисту виконує свої функції за допомогою процедур ідентифікації та автентифікації користувачів. Ці процедури важливі, тому що будь-якій системі захисту від НСД для виконання свого завдання необхідно, щоб усі легальні користувачі були ідентифіковані та гарантовані б відповідність між користувачами та їх ідентифікаторами, оскільки всі інші елементи системи захисту працюють із ідентифікованими суб'єктами. Система захисту виконує автентифікацію на основі певної унікальної інформації, яка характеризує конкретного користувача системи. Така інформація називається автентифікаційною. Залежно від її типу, методи автентифікації класифікують на наступні групи: парольна, майнова та біометрична. Простими методами автентифікації є методи, засновані на використанні унікальної інформації, яка відома користувачеві, яку він пред'являє в процесі автентифікації, так звана парольна система автентифікації. Дані методи набули найбільшого поширення завдяки простоті реалізації, впровадження та використання. Але всі вони мають величезний недолік, який робить їх непридатними для застосування у серйозних системах захисту. Цей недолік полягає в тому, що при порушенні конфіденційності пароля користувача відразу порушується захист усієї інформації, до якої він має доступ. Для підвищення стійкості автентифікації (тобто ускладнення будь-кому можливості автентифікуватися під чужим ім'ям) часто використовують кілька методів автентифікації одночасно. Автентифікація на основі одночасного пред'явлення автентифікаційної інформації двох видів називається двофакторною.

Види автентифікації:

*Автентифікація за допомогою електронного підпису*

Протоколи або схеми електронного підпису є основним криптографічним засобом забезпечення автентичності інформації:

* за допомогою електронного підпису одержувач документа може довести, що він належить відправнику, при цьому автор підпису не зможе заперечити факт надсилання підписаного документа;
* електронний підпис неможливо підробити; лише абонент, чий підпис стоїть на документі, міг підписати цей документ;
* електронний підпис є невід'ємною частиною документа, перенести його на інший документ не можна;
* ні відправник, ні отримувач що неспроможні змінити документ, залишивши цей факт непоміченим;
* будь-який користувач, який має зразок підпису, може впевнитись у справжності документа.

У класичній схемі електронного підпису передбачається, що підписуючий (абонент А) знає зміст документа, який він підписує; перевіряючий (абонент В), знаючи відкритий ключ перевірки підпису, може перевірити правильність підпису будь-коли без будь-якого дозволу чи участі претендента А.

*Автентифікація за багаторазовими паролями*

Цей алгортим полягає у введенні вашого ідентифікатора користувача, так званого «логіна» і пароля - деяких конфіденційних відомостей. Достовірна (еталонна) пара логін-пароль зберігається у спеціальній базі даних.

Проста автентифікація має такий загальний алгоритм:

* суб'єкт запитує доступ до системи та вводить особистий ідентифікатор та пароль;
* введені унікальні дані надходять на сервер автентифікації, де порівнюються з стандартними;
* при збігу даних з еталонними автентифікація визнається успішною, за відмінності - суб'єкт переміщається до 1-го кроку.

*Автентифікація за одноразовими паролями*

Автентифікація за одноразовими паролями зазвичай застосовується додатково до автентифікації за паролями для реалізації two-factor authentication (2FA). У цій концепції користувачеві необхідно надати дані двох типів для входу в систему: щось, що він знає (наприклад, пароль) і щось, чим він володіє (наприклад, пристрій для генерації одноразових паролів). Наявність двох факторів дозволяє значною мірою збільшити рівень безпеки, що м. б. потрібна для певних видів веб-додатків.

*Автентифікація за допомогою SMS*

Процедура такої автентифікації включає наступні кроки:

* введення імені користувача та пароля;
* відразу після цього PhoneFactor (служба безпеки) надсилає одноразовий автентифікаційний ключ як текстового SMS-повідомлення;
* отриманий ключ використовується для автентифікації.

Привабливість даного методу полягає в тому, що ключ виходить не по тому каналу, яким проводиться автентифікація, що практично виключає атаку типу «людина посередині». Додатковий рівень безпеки може призвести до введення PIN-коду мобільного засобу. Цей метод набув широкого поширення у банківських операціях через інтернет.

*Автентифікація за сертифікатами*

Сертифікат є набір атрибутів, що ідентифікують власника, підписаний certificate authority (CA). CA виступає у ролі посередника, який гарантує справжність сертифікатів (за аналогією з ФМС, що випускає паспорти). Також сертифікат криптографічно пов'язаний із закритим ключем, який зберігається у власника сертифіката та дозволяє однозначно підтвердити факт володіння сертифікатом. На стороні клієнта сертифікат разом із закритим ключем можуть зберігатися в операційній системі, браузері, файлі, на окремому фізичному пристрої (smart card, USB token). Зазвичай закритий ключ додатково захищений паролем або PIN-кодом.

*Автентифікація за ключами доступу*

Цей спосіб найчастіше використовується для автентифікації пристроїв, сервісів або інших програм при зверненні до веб-сервісів. Тут як секрет застосовуються ключі доступу (access key, API key) — довгі унікальні рядки, що містять довільний набір символів, які по суті замінюють комбінацію username/password. У більшості випадків сервер генерує ключі доступу на запит користувачів, які далі зберігають ці ключі в клієнтських програмах. При створенні ключа також можна обмежити термін дії та рівень доступу, який отримає клієнтська програма під час автентифікації за допомогою цього ключа.

*Автентифікація за токенами*

Такий спосіб автентифікації найчастіше застосовується при побудові розподілених систем Single Sign-On (SSO), де одна програма (service provider або relying party) делегує функцію автентифікації користувачів іншому додатку (identity provider або authentication service). Типовий приклад цього способу - вхід до програми через обліковий запис у соціальних мережах. Тут соціальні мережі є сервісами автентифікації, а програма довіряє функцію автентифікації користувачів соціальним мережам. Реалізація цього способу полягає в тому, що identity provider (IP) надає достовірні відомості про користувача у вигляді токена, а service provider (SP) додаток використовує цей токен для ідентифікації, автентифікації та авторизації користувача.

*Біометрична автентифікація*

Біометрія дозволяє ідентифікувати користувачів, спираючись на їх поведінкові та фізіологічні характеристики. До фізіологічним характеристикам можна віднести відбитки пальців, риси обличчя, геометрія долонь, вушних раковин, сітківка ока і т. д. Поведінкові характеристики включають рукописний текст почерк людини, хода, тембр голосу, швидкість набору тексту на клавіатурі і т.п. Якщо визначення користувача за допомогою сканування сітківки ока дуже дорогий спосіб, у зв'язку з вартістю обладнання, то ідентифікація користувача за клавіатурним почерком – дешевий і досить простий для реалізації варіант, оскільки для такої системи не потрібне додаткового обладнання. Потрібен стандартний набір периферійних пристроїв, які має у своєму розпорядженні будь-який персональний комп'ютер – клавіатура та монітор. А як система безпеки виступатиме програмний продукт, розробка якого і становить основну складність. Одним із доступних і актуальних методів біометричної автентифікації являється клавіатурний почерк. У процесі того як людина вводить інформацію використовуючи клавіші в нього виробляється свій особистий стиль набору тих чи інших слів. Важливою особливістю завдання автентифікації користувача на клавіатурному почерку є необхідність «навчання» програми, яка вироблятиме автентифікацію.

**Висновок.** Незважаючи на те, що існує безліч видів антивірусів з різними функціоналами і принципами роботи, а також великий реєстр розробників даного ПЗ, недоліків антивірусів, на жаль, теж чимало. Проте ще жодна антивірусна програма не зможе гарантувати вам стовідсоткового захисту від будь-якого вірусу. Це може бути новий невідомий вірус, який поки що не занесений до словників, або вірус, який був сильно зашифрований. Тоді знадобиться потужний розпакувальник, чого, зрозуміло, немає у багатьох антивірусах. Більш того, антивіруси люблять знаходити загрозу в абсолютно безпечних файлах. Тому рядові користувачі самі пропускають повз очі попередження про віруси і шкідливі файли, що робить захист менш надійним. Якщо подивитись на захист інформації під іншим кутом, то уряди багатьох країн брали активну участь у розробці міжнародних стандартів шифрування інформації. Виходячи з цього, урядовці можуть отримати доступ до всіх даних, зашифрованих загальноприйнятими методами захисту. Велика кількість відомих методів шифрування не дає стовідсоткової захищеності і індивідуальності інформації. Адже коротка вихідна довжина та простота операцій зробили більшість алгоритмів дуже легкими для злому та сприйнятливими до атаки. Основною причиною слабкої захищеності є відносно легке знаходження колізій (можливість отримання однакового результату хеш-функції при відмінних вхідних значеннях) при шифруванні. Недостатньо захищені дані призвели до поширеної появи автентфікації користувачів перед використанням будь-якого ресурсу. Недоліком всіх методів автентифікації можна вважати їх невдале проходження із подальшою втратою доступу до всіх даних закріплених за користувачем і відповідно проблемою додаткового підтвердження достеменності вашої особистості сервісу, який намагається захистити свою безпеку і цілісність.