МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.

«___» _____2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТР"

Тема: Програмний модуль протидії шкідливим програмам для операційних систем родини *MS Windows*

Виконавець: Скорін К.О.

Керівник: Кучеров Д.П.

Нормоконтролер: Тупота Є.В.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Освітнього ступеня магістр
Спеціальність 123 "Комп'ютерна інженерія"
(шифр, найменування)
Спеціалізація 123.02 "Системне програмування"
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Литвиненко О. Є.

«____»_____2020 р.

ЗАВДАННЯ
на виконання дипломної роботи (проекту)

Скоріна Костянтина Олександровича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема роботи: "Програмний модуль протидії шкідливим програмам для операційних систем родини *Microsoft Windows* "
затверджена наказом ректора від " 27 " серпня 2020 року № 1203 /ст.
2. Термін виконання роботи: з 05.10.2020 до 31.12.2020
3. Вихідні дані до роботи: 1) вимоги до змісту програмного модуля;
2) основні функції програмного модуля
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1) аналіз доцільності створення захисних програм;
2) проектування програмного модуля;
3) оцінювання якості програмного модуля протидії шкідливим програмам для операційних систем родини ms windows
5. Перелік обов'язкового графічного матеріалу:
1) функціональна модель системи виявлення та блокування ботнетів;
2) функціональна структура агента виявлення шкідливих програм;
3) вікно програмного модуля з попередженням про можливу наявність шкідливої програми;
4) таблиця коефіцієнтів зниження продуктивності інформаційної системи;
5) схема алгоритму виявлення мережевих атак.

6. Календарний план

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1	Провести аналіз літератури за темою дипломної роботи та аналіз існуючих систем	05.10.2020 – 06.10.2020	
2	Підготувати перший розділ пояснювальної записки	07.10.2020 – 10.11.2020	
3	Розробити структуру програмних засобів системи	11.11.2020 – 14.11.2020	
4	Розробити програмні засоби. Підготувати другий розділ пояснювальної записки	15.11.2020 – 19.11.2020	
5	Провести налаштування та тестування системи	20.11.2020 – 03.12.2020	
6	Завершити оформлення пояснювальної записки	04.12.2020 – 13.12.2020	
7	Підготувати презентацію та графічні матеріали	14.12.2020 – 22.12.2020	

7. Дата видачі завдання 05.10.2020

Керівник Кучеров Д.П.
(підпис)

Завдання прийняв до виконання Скорін К.О.
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи “Програмний модуль протидії шкідливим програмам для операційних систем родини *Microsoft Windows”*: 85 с., 16 рис., 29 літературних джерел, 2 додатки.

МОНІТОРИНГ, МЕРЕЖА, ТРАФІК, ІНФОРМАЦІЙНА БЕЗПЕКА, ШКІДЛИВІ ПРОГРАМИ, АНОМАЛІЇ ТРАФІКУ, ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА

Під час роботи над дипломною роботою було розроблено програмний модуль для протидії шкідливим програмам для операційних систем родини *Microsoft Windows.*

Об’єкт дипломного дослідження – процес протидії шкідливим програмам.

Предмет дипломного дослідження – програмний модуль для протидії шкідливим програмам.

Мета дипломного дослідження – розгляд основних методів моніторингу та аналізу аномалій мережевого трафіку, які спричинено діями шкідливих програм.

Розроблений в даній роботі програмний модуль для протидії шкідливим програмам для операційних систем родини *Microsoft Windows* дозволяє виявляти програми, що спричиняють аномальний трафік у комп’ютерній мережі.

Результати дипломної роботи було представлено на науково-практичній конференції та опубліковані у збірнику тез доповідей: Скорін К.О. Програмний модуль протидії шкідливим програмам для операційних систем родини *Microsoft Windows*// Тези доповідей наук.-практ. конф. “Сучасні тенденції розвитку системного програмування” (25-26 листопада 2020 р.). – К.: НАУ, 2020. – С. 16.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ВСТУП

На сьогоднішній день комп'ютерні віруси залишаються одним з найбільш обговорюваних видів загроз. Більш того саме з захисту від комп'ютерних вірусів зазвичай починають створювати систему інформаційної безпеки компанії. Однак, не дивлячись на те, що галузь антивірусної безпеки існує вже більше десяти років, даний вид погроз залишається одним з найбільш актуальних і небезпечних. Так, наприклад, за даними багатьох науково-дослідних центрів в Європі і США, щорічно збільшується не тільки кількість успішних вірусних атак, але і рівень шкоди, яка завдається компаніям в результаті використання зловмисниками шкідливого коду.

Багато в чому дана ситуація пояснюється тим, що багато хто вважає необхідною і достатньою умовою захисту від шкідливого коду використання одного лише антивірусного програмного забезпечення. Однак, на жаль, такий підхід вже давно застарів і не дозволяє забезпечити необхідний рівень інформаційної безпеки для більшості компаній. Практика показує, що стратегія антивірусної безпеки підприємства повинна визначатися моделлю порушника, від якого повинна бути захищена компанія.

Особливістю протидії шкідливим програмам є те, що це дуже широке коле програм, навіть за визначенням має різні поняття в різних джерелах:

1) шкідлива програма – програмне забезпечення, призначене для отримання несанкціонованого доступу до обчислювальних ресурсів самої ЕОМ або до інформації, що зберігається на ЕОМ, з метою несанкціонованого використання ресурсів ЕОМ або заподіяння шкоди (нанесення збитку) власникові інформації, і / або власнику ЕОМ, і / або власнику мережі ЕОМ, шляхом копіювання, спотворення, видалення або підміни інформації (Вікіпедія);

2) шкідлива комп'ютерна програма – комп'ютерна програма або інша комп'ютерна інформація, свідомо призначена для несанкціонованого знищення, блокування, модифікації, копіювання комп'ютерної інформації або нейтралізації засобів захисту комп'ютерної інформації (ст. 361-1 КК України);

3) шкідливий код – комп’ютерна програма, призначена для впровадження в автоматизовані системи, ПЗ, засоби обчислювальної техніки, телекомунікаційне обладнання кредитної організації і її клієнтів – користувачів систем дистанційного банківського обслуговування, що приводить до знищення, створення, копіювання, блокування, модифікації і (або) передачі інформації (в тому числі інформації, що захищається відповідно до пункту 2.1 Положення *N* 382-П), а також до створення умов для такого знищення, створення, копіювання, блокування, модифікації і (або) передачі (Лист Банку України від 24.03.2014 *N* 49- Т «Про рекомендації по організації застосування засобів захисту від шкідливого коду при здійсненні банківської діяльності).

4) програмне забезпечення, спеціально створене для того, щоб завдавати шкоди окремому комп’ютеру, серверу, або комп’ютерної мережі, незалежно від того, чи є воно вірусом, шпигунською програмою і т.д. (Визначення компанії *Microsoft).*

5) програма, призначена для здійснення доступу несанкціонованого до інформації і (або) впливу на інформацію або ресурс системи інформаційної (ГОСТ Р 50922-2006).

Це далеко не всі визначення, але, мабуть, вибірка досить характерна. По суті вона показує, що єдиної думки, від чого захищає антивірус – серед теоретиків немає. Більш того, немає навіть єдності щодо слова "шкідливий" – в ряді перекладних документів використовується поняття "Захист від зловмисних кодів" (наприклад, Р *ISO* / *IEC* ТО 13335-4-2007 Інформаційні технології. Методи і засоби забезпечення безпеки , *ISO* / *IEC* 27002 Інформаційні технології. Звід правил з управління захистом інформації).

Всі вищенаведені визначення відрізняються або чітким переліком конкретних функцій (що виключає відповідні заходи з протидії), або нечіткістю понять, що не дозволяє на їх основі сформулювати вимоги до антивірусного захисту.

На сьогодні термінологія щодо шкідливого ПЗ може бути розширена:

1) активний вірус – вірус, програмний код або частина програмного коду якого находіт¬ся в оперативній або віртуальної пам’яті і періодично виконується;

2) відомий вірус – вірус, інформація про який міститься в бізі АВС;

3) невідомий вірус – вірус, інформація про який не міститься в АВС;

4) вірусний вплив (ВВ) – зміна стану КС, викликане як проникненням елементов програмного коду вірусу в систему, так і результатом його виконання (активізації вірусу);

5) вірусне зараження (ВЗ) – зміна стану КС або окремих її елементів, викликане поширенням вірусу;

6) вірусоподібні вплив (що руйнує програмний вплив, РПВ) – зміна стану КС, викликане виконанням коду спеціально створеного програмного суб'єкта або сукупності таких суб'єктів котрі мають властивість реплікації.

Серед всіх видів шкідливого програмного забезпечення ботнети є головною платформою, за рахунок якої зловмисники створюють потужний засіб для організації, підтримки і постійного зростання злочинної діяльності.

Одним з найпотужніших методів протидії шкідливим програмам є моніторинг поведінки системи і постійний контроль сигнатур програмних кодів на появу аномальної активності та надання інформації про це.

Усе вище наведене дозволяє зробити висновок щодо необхідності постійного оновлення антивірусного програмного забезпечення і впровадження додаткових програмних модулів протидії шкідливим програмам.

Об'єкт дипломного дослідження – процес протидії шкідливим програмам.

Предмет дипломного дослідження – програмний модуль для протидії шкідливим програмам.

Мета дипломного дослідження – розгляд основних методів моніторингу та аналізу аномалій мережевого трафіку, які спричинено діями шкідливих програм.

У дипломній розглядаються існуючі види загроз безпеки, описується один з підходів до створення моделі порушника, а також можливі сценарії побудови комплексної системи захисту від вірусних загроз.

РОЗДІЛ 1
АНАЛІЗ ДОЦІЛЬНОСТІ СТВОРЕННЯ ЗАХИСНИХ ПРОГРАМ

1.1. Аналіз принципів протидії шкідливим програмам

Назва «шкідливі програми» співвідноситься з англомовним терміном “*malware”*, утвореним від двох слів: “*malicious”* (“зловмисний”) і “*software*” (“програмне забезпечення”). Існують і інші, більш рідкісні варіанти – “*badware”*, “*computer contaminant”*, “*crimeware”*. У лексиконі деяких фахівців зустрічаються жаргонні назви “шкідливий”, “зловредів”, “малваре”. У побуті все шкідливі програми часто називають комп’ютерними вірусами, хоча це термінологічно некоректно.

До шкідливих програм відносять будь-яке програмне забезпечення, несанкціоновано проникаюче в комп’ютерну техніку. Подібні додатки наносять прямий або непрямий збиток – наприклад, порушують роботу комп’ютера або викрадають особисті дані користувача. Шкідники створюються для реалізації двох основних груп цілей. Одна з них зводиться до отримання вигоди від впровадження в комп’ютер жертви. Наприклад, зловмисник домагається можливості управляти комп’ютером, краде секретну інформацію, здійснює вимагання. Друга група цілей не пов’язана з матеріальною вигодою. Написання шкідливого коду може бути проявом бажання автора, який створив програму, утвердитися в своїх уміннях, звичайним хуліганством або жартом.

1.1.1. Типи вірусних загроз безпеці комп’ютерних систем

Основними видами загроз антивірусної безпеки є різні типи шкідливого коду, здатного завдати шкоди для компанії. Шкідливий код може бути представлений у вигляді комп’ютерних вірусів, програми типу «Троянський кінь», а також програм типу «*adware*», «*spyware»* і ін.

Відповідно до визначення, комп’ютерні віруси являють собою спеціально створений програмний код, здатний самостійно поширюватися в комп’ютерному

середовищі. В даний час можна виділити наступні типи вірусів: файлові та завантажувальні віруси, «мережеві черв'яки», безтілесні віруси, а також комбінований тип вірусів. Кожен з цих типів вірусу відрізняється типом носія, а також шляхом поширення в АС. У більшості випадків комп'ютерні віруси спрямовані на порушення працездатності автоматизованої системи компанії.

Програми типу «Троянський кінь» (*Trojan Horses*) також відносяться до шкідливого програмного коду, проте, на відміну від вірусів, не мають можливості самостійного розповсюдження в АС. Програми даного типу маскуються під штатний ПЗ системи і дозволяють порушнику отримати віддалений несанкціонований доступ до тих вузлів, на яких вони встановлені. На сьогоднішній день даний вид шкідливого коду є найбільш поширеним і одночасно найбільш небезпечним видом шкідливого коду, так як дозволяє зловмисникові отримати доступ конфіденційної інформації компанії.

Шкідливе ПЗ типу «*spyware»* призначений для збору певної інформації про роботу користувача. Прикладом таких даних може служити список *Web*-сайтів, відвідуваних користувачем, перелік програм, встановлених на робочої станції користувача, вміст повідомлень електронної пошти та ін. Зібрана інформація перенаправляється програмами «*spyware»* на заздалегідь певні адреси в мережі Інтернет. Шкідливе ПЗ даного типу може бути потенційним каналом витоку конфіденційної інформації з АС.

Основна функціональна задача шкідливих програм класу «*adware*» полягає в відображенні рекламної інформації на робочих станціях користувачів. Для реалізації цього вони, як правило, виводять на екран користувача рекламні банери, що містять інформацію про ті чи інші товари і послуги. У більшості випадків ці програми поширюються разом з іншим ПЗ, яке встановлюється на вузли АС. Незважаючи на те, що програми типу «*adware*» не уявляють безпосередню загрозу для конфіденційності або цілісності інформаційних ресурсів АС їх робота може призводити до порушення доступності внаслідок несанкціонованого використання обчислювальних ресурсів робочих станцій.

Необхідно відзначити, що разом з розвитком інформаційних технологій з'являються нові види шкідливого коду.

1.1.2. Модель порушника антивірусної безпеки

У модель порушника, яка описана в даній роботі, визначені чотири класи потенційних порушників, кожен з яких характеризується певним рівнем кваліфікації та ступенем навмисності виконуваних дій.

Порушники, що відносяться до класу «Н-1», являють собою рядових співробітників, в результаті ненавмисних дій яких може відбутися інфікування автоматизованої системи компанії. Прикладами таких дій є скачування з мережі Інтернет неперевірених файлів і запуск їх на локальному комп'ютері.

Порушники класу «Н-2» виконують навмисні дії, однак для проведення вірусної атаки використовуються відомі екземпляри шкідливого коду, а також опубліковані уразливості програмного забезпечення.

Клас порушників «Н-3» передбачає наявність у зловмисника вищого рівня кваліфікації, що дає йому можливість використовувати шкідливий код, який може детектуватиметься не всіма антивірусними продуктами.

Порушники класу «Н-4» є найбільш небезпечними і мають достатню кваліфікацію для розробки шкідливого коду, який не виявляється антивірусними програмними продуктами.

Необхідно відзначити, що в рамках описаної моделі передбачається, сто порушники «Н-2», «Н-3» і «Н-4» є зовнішніми по відношенню до компанії, що атакується і мають мінімум інформації про автоматизовану систему підприємства. Загальна характеристика описаної моделі порушника приведена в таблиці 1.1.

Таблиця 1.1

Модель потенційного порушника антивірусної безпеки

№	Клас порушника	Ступінь навмисності дій порушника	Рівень кваліфікації порушника
1	Клас «Н-1»	ненавмисні дії	-
2	Клас «Н-2»	умисні дії	низький
3	Клас «Н-3»	умисні дії	середній
4	Клас «Н-4»	умисні дії	високий

Представлена модель є лише одним з можливих прикладів класифікації порушника. Дана модель може бути значно розширена за рахунок додавання в неї наступних параметрів:

– вид порушника: зовнішній чи внутрішній;

– рівень знань про автоматизовану систему компанії і застосовуваних засобах захисту інформації;

– можливість використання потенційним порушником спеціалізованих програмних засобів в автоматизованій системі компанії.

1.1.3. Модель захисту компаній від шкідливого коду

Для захисту від порушників класу «Н-1» і «Н-2» досить використовувати стандартне антивірусне програмне забезпечення одного виробника, встановивши його на всі робочі станції і сервери в компанії. Такий підхід може застосовуватися для невеликих компаній, а також для підприємств, де обробляється невеликий обсяг конфіденційної інформації.

Для захисту від порушника класу «Н-3» необхідно використовувати многовендорний варіант антивірусного захисту, який передбачає застосування антивірусних ядер різних виробників. Це дозволить істотно підвищити ймовірність виявлення вірусу за рахунок того, що кожен файл або поштове повідомлення буде перевірятися різними ядрами. Ще однією перевагою використання багатоядерних антивірусів є більш висока надійність роботи системи антивірусного захисту. У разі, якщо в одному з скануючих ядер системи будуть перебої, то воно завжди може бути замінено іншим активним антивірусним ядром. Як правило, виділяють три рівні антивірусного захисту, на кожному з яких можуть використовуватися антивірусні продукти різних виробників:

– рівень шлюзу, на якому засоби антивірусного захисту встановлюються на межсетевом екрані або проксі-сервері. Ще одним варіантом захисту АС на рівні шлюзу може бути установка спеціалізованих програмно-апаратних комплексів (*appliance)* в точці підключення АС до мережі Інтернет;

– рівень серверів, В рамках якого антивірусні агенти встановлюються на файлові, поштові та інші сервери АС;

– рівень робочих станцій користувачів, На якому антивіруси встановлюються на всі робочі місця користувачів з можливістю централізованого управління з єдиної консолі.

В якості альтернативи використання декількох продуктів різних виробників можливе застосування програмних комплексів, які включають в себе кілька ядер з єдиною консоллю управління. Прикладом продуктів такого класу є система *ForeFront* компанії *Microsoft* і *GFI_Mail_Security* компанії *GFI.* У зв'язку з цим необхідно відзначити, що антивірусний продукт *ForeFront* може включати в себе одночасно від п'яти до дев'яти антивірусних ядер різних виробників.

Описаний вище підхід до захисту від порушників класу «Н-3» рекомендується застосовувати для компаній середніх розмірів.

Для захисту від порушників класу «Н-4» застосування одних лише антивірусних продуктів недостатньо, так як зловмисники даної категорії мають можливості створювати шкідливий код, який не виявляється антивірусами. Тому в доповненні до многовендорной антивірусного захисту в АС компанії необхідно використовувати підсистему мережевого екранування, виявлення і запобігання атак, а також виявлення вразливостей. Застосування даних підсистем дозволить істотно знизити ризик успішної реалізації інформаційної атаки на компанію.

Підсистема мережного екрануванняпризначена для захисту робочих станцій користувачів від можливих мережевих вірусних атак за допомогою фільтрації потенційно небезпечних пакетів даних. Підсистема повинна забезпечувати можливість фільтрації на канальному, мережевому, транспортному і прикладному рівнях стека *TCP/IP*. Як правило, дана підсистема реалізується на основі міжмережевих і персональних мережевих екранів. При цьому міжмережевий екран встановлюється в точці підключення АС до мережі Інтернет, а персональні екрани розміщуються на робочих станціях користувачів.

Підсистема виявлення і запобігання атак призначена для виявлення несанкціонованої вірусної активності за допомогою аналізу пакетів даних, що циркулюють в АС, а також подій, що реєструються на серверах і робочих

станціях користувачів. Підсистема доповнює функції міжмережевих і персональних екранів за рахунок можливості більш детального контекстного аналізу вмісту переданих пакетів даних. Ця підсистема включає в себе наступні компоненти:

– мережеві і хостової сенсори, призначені для збору необхідної інформації про функціонування АС. Мережеві сенсори реалізуються у вигляді окремих програмно-апаратних блоків і призначені для збору інформації про всі пакетах даних, переданих в рамках того мережевого сегмента, де встановлений сенсор. Даний тип сенсорів встановлюється у всіх ключових сегментах АС, де розташовані захищаються вузли системи. Хостової сенсори встановлюються на робочі станції і сервери АС і збирають інформацію про всі події, що відбуваються на цих вузлах системи. Хостової сенсори можуть збирати інформацію не тільки про пакетах даних, але і інших операціях, які виконуються додатками, запущеними на вузлі АС;

– модуль виявлення атак, що виконує обробку даних, зібраних сенсорами, з метою виявлення інформаційних атак порушника. Даний модуль підсистеми повинен реалізовувати сигнатурні і поведінкові методи аналізу інформації;

– модуль реагування на виявлені атаки. Модуль повинен передбачати можливість як пасивного, так і активного реагування. Пасивне реагування передбачає оповіщення адміністратора про виявлену атаці, в той час як активне – блокування спроби реалізації вірусної атаки;

– модуль зберігання даних, в якому міститься вся конфігураційна інформація, а також результати роботи підсистеми.

Підсистема виявлення вразливостейповинна забезпечувати можливість виявлення технологічних та експлуатаційних вразливостей АС за допомогою проведення мережевого сканування. Як об’єкти сканування можуть виступати робочі станції користувачів, сервери, а також комунікаційне обладнання. Для проведення сканування можуть використовуватися як пасивні, так і активні методи збору інформації. За результатами роботи підсистема повинна генерувати детальний звіт, що включає в себе інформацію про виявлені вразливості, а також рекомендації щодо їх усунення. Спільно з підсистемою виявлення вразливостей в

АС може використовуватися система управління модулями оновлень загальносистемного і прикладного ПЗ, встановленого в АС.

Підсистема управління антивірусним безпекою, Призначена для виконання наступних функцій:

– віддаленої установки і деінсталяції антивірусних засобів на серверах і робочих станціях користувачів;

– віддаленого управління параметрами роботи підсистем захисту, що входять до складу комплексної системи антивірусного захисту;

– централізованого збору та аналізу інформації, що надходить від інших підсистем. Ця функція дозволяє автоматизувати процес обробки даних, що надходять, а також підвищити оперативність прийняття рішень з реагування на виявлені інциденти, пов’язані з порушенням антивірусної безпеки.

Описаний вище підхід до захисту від порушників класу «Н-4» доцільно застосовувати в великих і територіально-розподілених компаніях, ресурси яких потенційно можуть бути схильні до цільовим атакам зловмисників, які реалізуються за допомогою шкідливого коду.

Нижче в таблиці 1.2 показані основні механізми захисту, які повинні застосовуватися в залежності від обраного класу порушника.

Таблиця 1.2

Модель захисту від потенційного порушника антивірусної безпеки

№	клас порушника	механізми безпеки
1	Клас «Н-1»	Антивірусні продуті одного виробника
2	Клас «Н-2»	
3	Клас «Н-3»	Антивірусні продукти різних виробників
4	Клас «Н-4»	Антивірусні продукти різних виробників, засоби міжмережевого екранування, засоби виявлення і запобігання атак, виявлення вразливостей

1.1.4. Комплексний підхід до захисту від вірусних загроз

Комплексний підхід до захисту від шкідливого коду передбачає узгоджене застосування правових, організаційних і програмно-технічних заходів, що перекривають в сукупності всі основні канали реалізації вірусних загроз. Відповідно до цього підходу в організації повинен бути реалізований наступний комплекс заходів:

– заходи по виявленню та усуненню вразливостей, на основі яких реалізуються вірусні загрози. Це дозволить виключити причини можливого виникнення вірусних атак;

– заходи, спрямовані на своєчасне виявлення і блокування вірусних атак;

– заходи, що забезпечують виявлення і ліквідацію наслідків вірусних загроз. Даний клас заходів захисту спрямований на мінімізацію збитку, нанесеного в результаті реалізації вірусних загроз.

Важливо розуміти, що ефективна реалізація перерахованих вище заходів на підприємстві можлива тільки за умови наявності нормативно-методичного, технологічного і кадрового забезпечення антивірусної безпеки.

Нормативно-методичне забезпечення антивірусної безпеки передбачає створення збалансованої правової бази в галузі захисту від вірусних загроз. Для цього в компанії повинен бути розроблений комплекс внутрішніх нормативних документів і процедур, що забезпечують процес експлуатації системи антивірусної безпеки. Склад таких документів багато в чому залежить від розмірів самої організації, рівня складності АС, кількості об’єктів захисту і т.д. Так, наприклад, для великих організацій основним нормативним документом у сфері захисту від шкідливого коду повинна бути концепція або політика антивірусної безпеки. Для невеликих компаній досить розробити відповідні інструкції і регламенти роботи користувачів, а також включити вимоги до забезпечення антивірусного захисту до складу політики інформаційної безпеки організації.

В рамках кадрового забезпечення антивірусної безпеки в компанії повинен бути організований процес навчання співробітників з питань протидії вірусним

загрозам. Програма навчання повинна бути спрямована на мінімізацію ризиків, пов'язаних з помилковими діями користувачів, що призводять до реалізації вірусних атак. Прикладами таких дій є: запуск додатків з неперевірених зовнішніх носіїв, використання нестійких до вгадування паролів доступу, закачування *ActiveX*-об'єктів з недовірених *Web*-сайтів та ін. В процесі навчання повинні розглядатися як теоретичні, так і практичні аспекти антивірусного захисту. При цьому програма навчання може складатися в залежності від посадових обов'язків співробітника, а також від того до жодних інформаційних ресурсів він має доступ.

В рамках комплексного підходу необхідно періодично проводити аудит антивірусної безпеки, що дозволяє отримати незалежну оцінку поточного рівня захищеності компанії від шкідливого коду. В рамках аудиту проводиться перевірка ефективності використовуваних механізмів безпеки, визначаються основні уразливості в системі заходів антивірусного захисту і виробляються рекомендації по їх усуненню.

1.2. Аналіз методів захисту від *DDoS*-атак рівня додатків

Спеціально створена *DDoS* атака рівня додатків дозволяє каскадно вивести з ладу системи, використовуючи набагато менший обсяг ресурсів у порівнянні з тими ресурсами, які необхідні для проведення традиційних *DDoS* атак. Подібний розклад можливий через складні взаємозв'язків, що існують між додатками. Традиційні *DDoS* атаки націлені на виснаження ресурсів системи на мережевому рівні. На рівні додатків увага зосереджена на ресурсовитратності *API*-виклики і взаємозв'язках, щоб спровокувати атаку системи на саму себе, і іноді з лавиноподібним ефектом. У сучасній архітектурі мікро-служб подібних підхід може виявитися особливо руйнівним.

Зловмисник може створити витончені шкідливі запити, що імітують легітимний трафік, який буде проходити через всі захисні системи, в тому числі і *WAF* (*web application firewall*; фаєрвол для веб-додатків).

Згідно зі звітом компанії *Akamai*, *DDoS* атаки на рівні додатків займають менше 1% серед всіх *DDoS* атак. Однак ця метрика не відображає ступінь впливу подібних атак. Коли зловмисник витрачає час на підготовку плану *DDoS*-атаки, ефективність цього сценарію зростає в рази. З огляду на цей факт, при захисті від подібного типу атак в першу чергу необхідно переконатися, що не відбудеться вихід з ладу систем лавиноподібним чином.

1.2.1. Аналіз програмного забезпечення для запобігання *DDoS*-атак

За останніми даними *Incapsula* [1], *DDoS* обходиться бізнесу в 40000 доларів на годину.

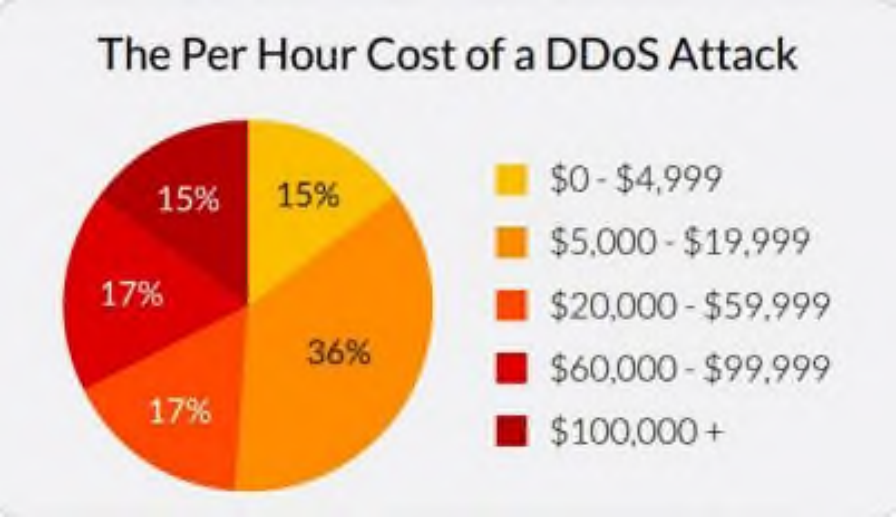

Рис. 1.1. Дані про фінансові втрати від *DDoS*-атак

Для розрахунку вартості *DDoS* атаки використовуються наступні інструменти:

– *Cost of a DDoS attack* від *Akamai*;

– *DDoS downtime cost calculator* від *Incapsula;*

– *Calculate your DDoS attack cost* від *Neustar*.

DDoS-атаки непередбачувані, а останнім часом ще й жахливо небезпечні. Потужність може бути в межах від 300 до 500 *Gbps*.

Зломщики вдаються до різних методиках, щоб нанести удар, в тому числі:

– фрагментація *UDP*-пакетів;

– *DNS, NTP, UDP, SYN, SSPD, ACK*-флуд;

– *CharGEN*-атака;

– аномалії *TCP.*

Тому необхідно захищатися не тільки від *DDoS*-атак насьомому рівні, А забезпечувати захист на всіх рівнях.

згідно з останнімзвіту про безпеку *AKAMAI* [], більшість *DDoS*-атак виробляються з Китаю.

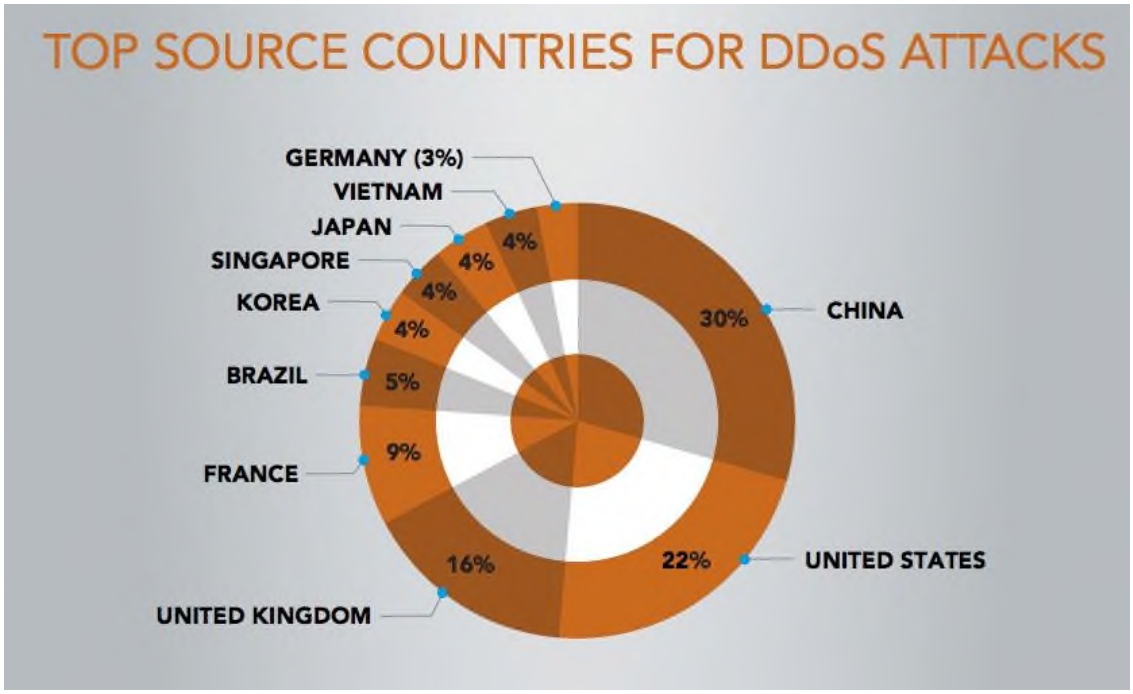

Рис. 1.2. Статистика по країнам-джерелам *DDoS*-атак

Розглянемо хмарні рішення для малого та середнього бізнесу, за допомогою яких можна підключити захист від *DDoS*-атак за лічені хвилини.

1.2.2. Аналіз можливостей системи комплексної безпеки *Incapsula*

Incapsula пропонує комплексний захист і захищає від будь-яких видів *DDoS*-атак на 3, 4 і 7 рівнях, таких як:

– *TCP SYN + ACK, FIN, RESET, ACK, ACK + PSH*, фрагментація;

– *UDP;*

– *Slowloris;*

– спуфинг;

– *CMP;*

– *GCP;*

– *HTTP*-флуд, запуск великого числа одночасних з’єднань, *DNS*-флуд;

– брутфорс;

– *NXDomain*;

– *Ping of death.*

Захист може здійснюватися в постійному режимі або включатися вручну для пошуку та усунення загроз.мережа *Incapsula* складається з 32 дата-центрів із загальною пропускною спроможністю понад 3 *Tbps*.

Доступна пробна версія *Incapsula* пакетів*Business* і *Enterprise*, В які входить захист від *DDoS*-атак, а також глобальні *CDN*, *SSL*, *WAF* і не тільки.

Рис. 1.3. Принцип протидії атаці системи безпеки *Incapsula*

1.2.3. Аналіз можливостей системи комплексної безпеки *KONA DDoS Defender*

Компанія *AKAMAI* займає провідне становище на ринку за службу безпеки і *CDN*. Зовсім недавно *AKAMAI* поставила рекорд, відбивши атаку потужністю 620 *Gbps*.

KONA DDoS Defender від *AKAMAI* захищає від *DoS/DDoS*-атак на периферію мережі.

KONA створена на основі інтелектуальної платформи *AKAMAI* для захисту веб-сайтів, а реагує на атаки глобальний центр безпеки, який працює цілодобово.

Це хмарне рішення оберігає від усіх відомих видів атак, в тому числі з шифруванням трафіку. *AKAMAI* широко представлений по всьому світу: всього у *AKAMAI* понад 1300 майданчиків в більш ніж 100 країнах.

AKAMAI захищає інфраструктуру всього дата-центру за допомогою*Prolexic Routed* або *Prolexic Connect.*

1.2.4. Аналіз можливостей системи комплексної безпеки *Cloudflare*

Cloudflare надає базовузахист від *DDoS*-атакв пакетах *FREE* і *PRO.* Однак для розширеної захисту від *DDoS*-атак (на рівнях 3, 4 і 7) потрібно купувати пакет *Business* або *Enterprise*.

Вартість послуг *Cloudflare* фіксована, а це значить, що з якою б масштабною атакою не зіткнувся користувач, платити буде однакову суму щомісяця.

Cloudflare довіряють такі відомі компанії як *Nasdaq,DigitalOcean, Cisco, Salesforce, Udacity* і ін.

Мережа *CloudFlare* доступна в 102 дата-центрах загальною пропускною спроможністю понад 10 *Tbps*, тобто вона впорається з будь-якими видами *DDoS*-атак, в тому числі:

– на рівнях 3, 4 і 7;

– *DNS amplification;*

– *DNS reflection*;

– *SMURF;*

– *ACK.*

1.2.5. Аналіз можливостей спеціалізованого хмарного рішення *SUCURI*

SUCURI – спеціалізоване хмарне рішення для захисту самих різних сайтів, в тому числі *WordPress, Joomla, Drupal, Magento, Microsoft.Net* і ін.

Захист від *DDoS*-атаквключена в пакет антивіруса і брандмауера. У разі якщо потрібна комплексна захист сайту, то в такому випадку підійде *Website*

Antivirus, який оберігає від загроз в мережі, в тому числі від *DDoS*-атак, а також включає наступні послуги:

– виявлення і видалення шкідливого коду;

– контроль безпеки;

– оптимізація швидкості;

– захист від брутфорса;

– захист від *zero-day;*

– захист від небажаних ботів.

SUCURI виявляє і блокує атаки 3,*4* і 7 рівня. Вартість обслуговування починається від 19,*8*8 доларів в місяць.

1.2.7. Аналіз можливостей вузькоспеціалізованого програмного забезпечення захисту від *DDoS*-атак

На ринку присутня величезна кількість простих додатків для протидії атакам. Серед даного програмного забезпечення є платні, умовно безкоштовні й безкоштовні рішення. Розглянемо можливості деякі з них:

1) *Myra DDoS protection*- повністю автоматизоване рішення для веб-сайтів, *DNS*-серверів, веб-додатків та інфраструктури. Воно повністю сумісний з усіма видами *CMS* і системами електронної торгівлі. *MYRA* розміщується в Німеччині, так що дані обробляються відповідно до федеральним законом Німеччини про захист даних;

2) *AWS Shield Advanced*. Інструмент для попередження *DDoS*-атак*Shield*від *AWS* доступний на всіх додатках *AWS* безкоштовно. Проте, якщо необхідно забезпечити мережу розширеної захистом, захистити транспортний і прикладний рівень, то можна підписатися на *Shield Advanced*. У *Shield Advanced* є цілий ряд переваг в порівнянні зі стандартною версією *Shield*, таких як:

– виявлення – перевірка мережевого потоку і відстеження трафіку на прикладному рівні;

– захист від атак – передова система захисту від великомасштабних атак, що включає блокування шкідливого трафіку;

– підтримка – цілодобова команда підтримки при *DDoS*-атаках;

– аналіз – проведення аналізу після атаки.

1.3. Методи виявлення атак рівня додатків

Традиційні *DDoS* атаки на рівні додатків були засновані на попередній підготовці вхідних параметрів з урахуванням можливостей системи по генерації вихідних даних. Подібні сценарії засновані на використанні ресурсовитратності викликів (наприклад, запитів до баз даних або операції з дисковими системами) з метою надмірного використання додатка до тих пір, поки не припиниться обслуговування легітимних користувачів. А оскільки архітектура додатка є частиною більш складних і розподілених систем, у нас з'являється додаткова задача по перевірці служб і складних залежностей мікрослужб, які можуть постраждати, якщо один ключовий сервіс стає нестабільним.

У сучасній архітектурі мікрослужб *DDoS* атака на рівні додатків може стати особливо ефективною, якщо ставиться завдання щодо виведення з ладу цієї служби. Щоб зрозуміти чому, розглянемо приклад мікрослужби, що використовує шлюз для взаємодії з декількома мікрослужбами на середньому рівні і бекенда, як показано на малюнку нижче.

На діаграмі вище показано, як один запит в шлюзі може перетворитися в тисячі запитів на середньому рівні і бекенда. Якщо зловмисник зможе знайти *API*-виклики, які призводять до подібного ефекту, то далі можна використовувати цю фішку проти всієї системи.

Якщо результуючі обчислення будуть споживати дуже багато ресурсів, деякі служби на середньому рівні можуть зупинитися. У підсумку, в залежності від рівня критичності цих служб, вся система піддасться загрозу в тій чи іншій мірі.

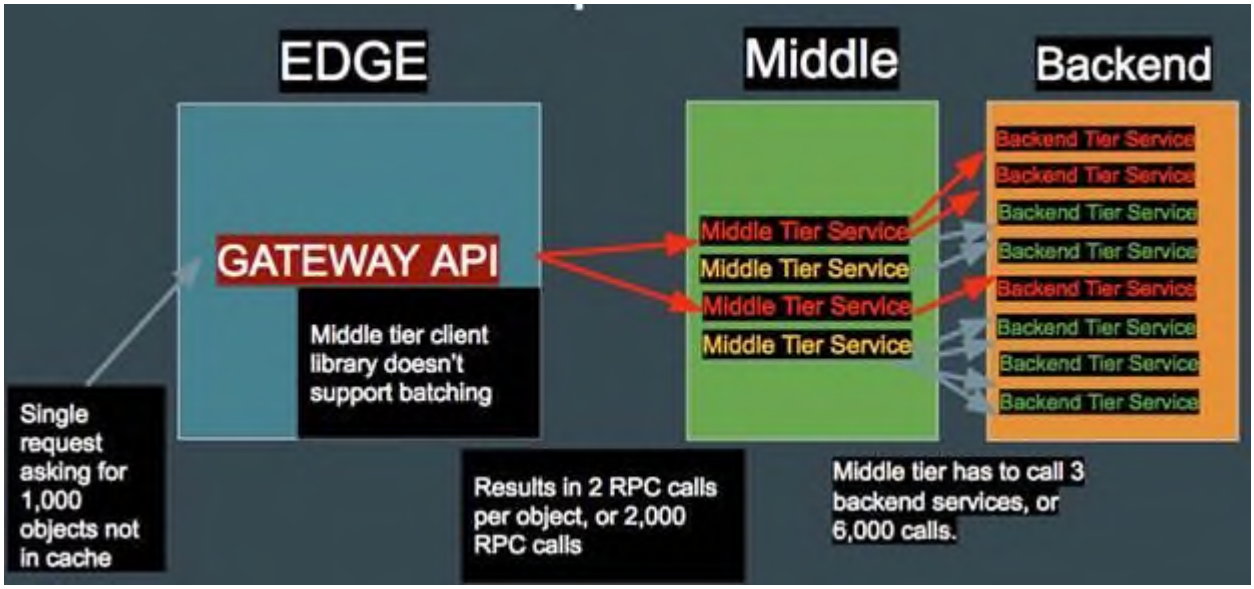

Рис. 1.4. Схема взаємодії мікрослужб на різних рівнях

Подібна схема можлива завдяки архітектурі мікрослужб, яка дозволяє зловмисникові серйозно посилити атаку проти внутрішніх систем. Тобто один запит до мікрослужбе може згенерувати десятки тисяч складних викликів до служб на серединному рівні і бекенда.

Цей факт додає головного болю фахівця з безпеки. Якщо у вашому середовищі використовується *WAF*, налаштований стандартним чином (наприклад, як *API*-шлюз), то в такому режимі можуть пропускатися запити, спеціально спрямовані для виходу з ладу служб на серединному рівні і бекенда. Фаєрвол може «не здогадуватися» про наслідки одного запиту, спрямованого до вищезазначених служб, і як результат досягнуто не спрацює фільтр по чорному списку, що може привести до плачевних наслідків.

Фахівцям з безпеки, важливо розуміти, як знайти виклики, які потенційно можуть використовуватися в *DDoS* атаках на рівні додатків.

Програма безпеки повинна знайти запити, які вимагають багато ресурсів від служб на серединному рівні і бекенда. Один способів виявлення – вимір часу виконання *API*-викликів. Найпростіший і не дуже надійний спосіб – ідентифікація *API*-викликів через браузер. Відкриваємо консоль *Chrome Developer*, вибираємо вкладку *Network*, виставляємо прапорець *Preserve log* і далі починаємо переглядати сайт. Через деякий час сортуємо запити по колонці *Time* і

дивимося виклики з найбільшим часом виконання. Тоді отримуємо таблицю, схожу з тією, яка показана на рисунку 1.5.

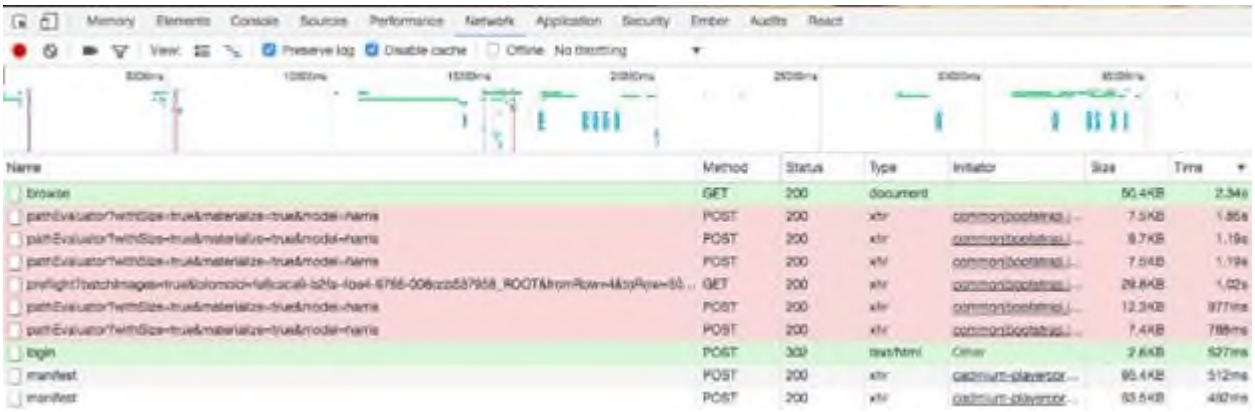

Рис. 1.5. Запити сайту, відсортовані в порядку убування за часом виконання

Ця техніка може призвести до помилкових спрацьовувань, включаючи запити, які не можна модифікувати для збільшення часу виконання. Або можна пропустити виклики, які можна змінити для збільшення часу виконання. Більш точна техніка пов’язана з моніторингом періодичності запитів до служб серединного рівня. Як тільки на серединному рівні виявлена служба, яка використовує виклики з великим часом виконання, потрібно реконструювати запит, який може пройти через *API*-шлюз для повторного виконання запиту до знайденої службі.

Після знаходження декількох підозрілих *API*-викликів наступний крок – інспекція вмісту цих функцій. Наше завдання – зробити так, щоб ці виклики стали споживати більше ресурсів. Одна з технік – збільшення діапазону запитуваних об’єктів. Наприклад, на рисунку 1.6 показано, як можна модифікувати параметри *from* і *to* для збільшення навантаження на служби серединного рівня.

Якщо провести більш глибокий аналіз, то можна знайти багато інших елементів запиту для збільшення споживання ресурсів. На рис. 1.7 показаний приклад модифікації поля запитуваної об’єкта, діапазону і навіть розміру картинки. Крім того, можна сформувати список індикаторів, що відображають успішність тестування, і які інформують вас про те, що тестування знаходиться в процесі виконання і місця збільшення / зменшення масштабу.

Рис. 1.6. Вміст одного із запитів, який потенційно може навантажити службу

Рис. 1.7. Потенційні місця запитуваної об’єкта, які можна змінити

Зазвичай індикатори включають в себе статусні *HTTP*-коди і час виконання окремих запитів під час тестування, але можуть використовуватися і інші показники: заголовки, текст відповіді, вміст стека, тощо. На рис. 1.8 показаний приклад переліку індикаторів.

Рис. 1.8. Перелік показників, які використовуються під час тестування

Ще один бажаний індикатор успішності тесту – збільшений час виконання (наприклад, *HTTP*-код 200 і час відповіді 10 секунд). Можна виміряти час виконання під час тестування або коли інші користувачі використовують додаток. Як тільки знайдені типи запитів, які призводять до збільшення неактивності і сформували індикатори успішності тестування, необхідно переналаштувати тести з урахуванням *WAF*, якщо такий використовується в середовищі.

Ідеальний обсяг трафіку повинен бути нижче, ніж допустимий поріг, коли фаєрвол починає блокування, але достатній, щоб набір запитів і кількість споживаних ресурсів вивели службу з ладу.

1.4. Аналіз програмного забезпечення для визначення наявності на комп'ютері додатків для *DDoS*-атак

1.4.1. Фреймворк *Repulsive Grizzly*

Для полегшення проведення тестування в невеликих масштабах підходить фреймворк *Repulsive Grizzly*, який підтримується компанією *Netflix* в рамках експериментального проекту з відкритим вихідним текстом. Вихідні тексти публікуються в якості доказу нашої концепції без гарантії подальшої підтримки протягом тривалого часу. Цей фреймворк написаний на *Python* з використанням бібліотеки *eventlet* для підтримки більш високої узгодженості. Також є підтримка циклічного перебору аутентифікаційних об'єктів, що може стати в нагоді при обході деяких видів *WAF'*ов.

Фреймворк *Repulsive Grizzly* не допомагає у виявленні вразливостей, які допомагають здійснити *DDoS* атаку на рівні додатків. Як і у випадку з усіма іншими утилітами, пов'язаними з тестуванням безпеки, важливо не забувати використовувати ці інструменти тільки там, де дозволено. Спочатку необхідно знайти потенційні проблеми, згадані вище. Як тільки у вас з'явився «матеріал» для тестування, фреймворк *Repulsive Grizzly* спростить увесь інший процес.

1.4.2. Фреймворк *Cloudy Kraken*

Після тестування гіпотез в невеликих масштабах можна переходити до масштабування за допомогою фреймворка *Cloudy Kraken*, який працює на базі платформи *Amazon Web Services* (*AWS*). Як і у випадку з фреймворком *Repulsive Grizzly*, *Cloudy Kraken* представлений в якості експериментального проекту з відкритим вихідним кодом.

Cloudy Kraken допомагає підтримувати глобальний набір екземплярів тестування, а *Repulsive Grizzly* відповідає за механіку всередині кожного примірника. Також цей фреймворк створює і розподіляє тестову конфігурацію і використовує розширені мережеві драйвера сервісу *AWS EC*2. *Cloudy Kraken* дозволяє масштабувати тестування серед кількох регіонів і підтримує синхронізацію за часом для агентів, запущених паралельно. На діаграмі рисунку 1.9 показана загальна схема роботи *Cloudy Kraken*:

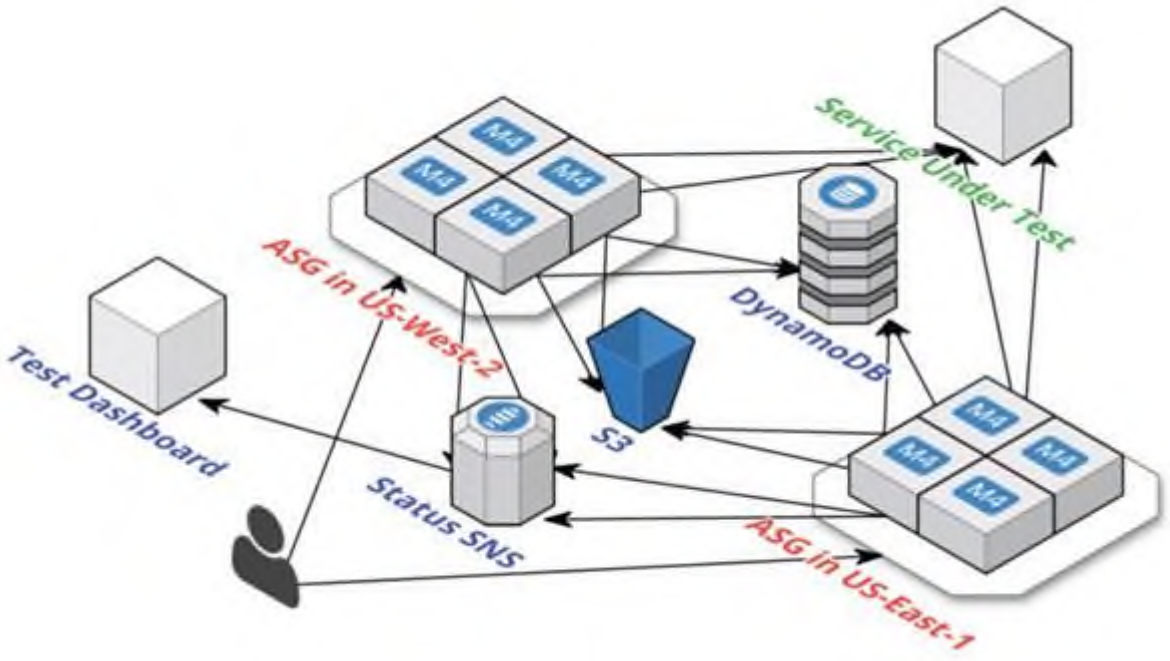

Рис. 1.9. Архітектура фреймворка *Cloudy Kraken*

Cloudy Kraken організовує ваші тести в зрозумілій і дружній формі. Все починається з конфігураційних скриптів, що визначають логіку тестування. Потім створюється *AWS*-середовище для тестування і запуску примірників. Поки проводиться тестування, *Cloudy Kraken* збирає інформацію за допомогою служби

AWS SNS. В кінці тестування *AWS*-ресурси знищуються. Вся послідовність кроків показана на діаграмі на рисунку 1.10.

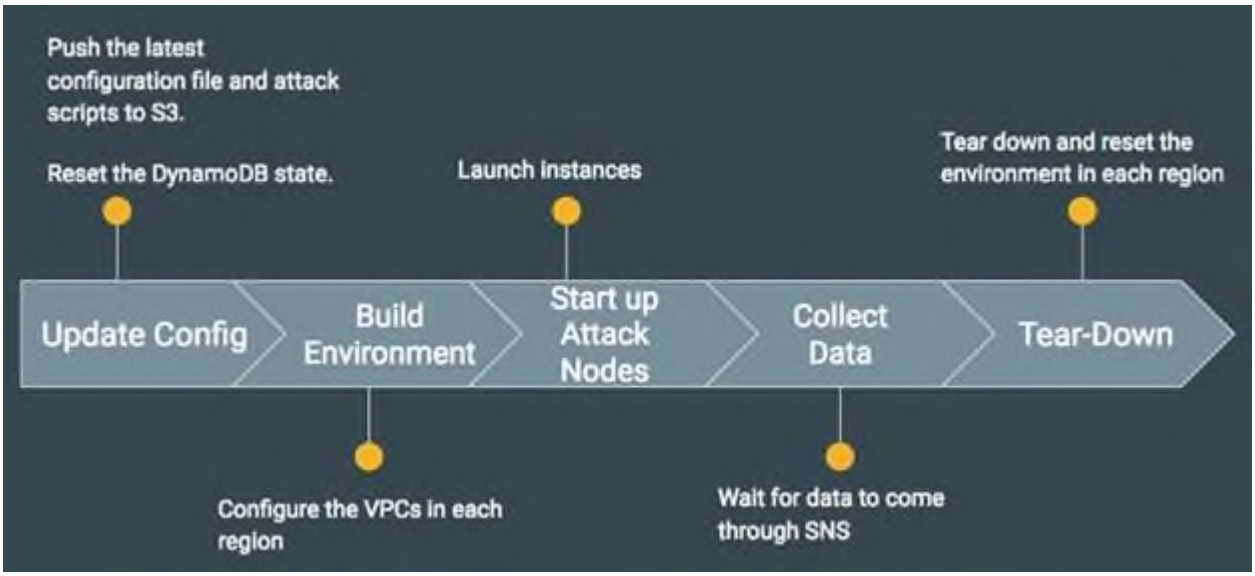

Рис. 1.10. Послідовність кроків для настройки розподіленого тестування

1.5. Висновки до розділу

Комп'ютерні віруси є в даний час однієї з найбільш значущих загроз інформаційній безпеці, про що свідчать численні дані по щорічним фінансових втрат компаній в результаті впливів вірусних атак. Вибір стратегії захисту від шкідливого коду повинен визначатися моделлю зловмисника, приклад якої розглянуто в цій роботі. Представлена модель показує, що для ефективного захисту компанії необхідно в доповненні до класичних антивірусних засобів застосовувати системи виявлення атак, виявлення вразливостей і системи міжмережевого екранування.

Перший і найважливіший метод захисту від *DDoS*-атак рівня додатків – розуміння того, як працює ваша система. Необхідно розуміти, які мікрослужби впливають на кожен аспект, пов'язаний досвідом взаємодії з покупцями / замовниками системи. Необхідно скоротити кількість взаємозалежностей з цими службами. Якщо одна служба стає недоступною, інші мікрослужби повинні продовжувати свою роботу (можливо, в більш нестабільному стані).

Важливо мати хороше розуміння про черговість роботи служб і про механіку запитів. Можливо, на серединному рівні і бекенда слід обмежити розмір черги або розмір запитуваних об'єктів, що можна зробити як в коді клієнта, так і на рівні *API*-шлюзу. Установка обмежень на обсяг дозволених запитів може значно знизити ймовірність реалізації подібних атак.

Рекомендується використовувати зворотний зв'язок, коли від служб серединного рівня і бекенда відсилаються повідомлення до *WAF*. Ця схема допоможе повідомити *WAF* про те, коли блокувати подібного роду атаки. У багатьох випадках *WAF* здійснює тільки моніторинг прикордонної зони і може не здогадуватися про вплив одного запиту на *API*-шлюз. Бажано також, щоб *WAF* моніторив промахи кешу (*cache miss*). Якщо *API*-шлюз постійно виконує виклик служб на серединному рівні через промахи кешу, значить, кеш налаштований некоректно або здійснюються шкідливі дії.

API-шлюзи та інші мікрослужби повинні віддавати пріоритет аутентифікаційних трафіку в порівнянні з неаутентіфікаціонним, оскільки від зловмисника потрібно більше ресурсів і майстерності для використання аутентифікаційних сесій. Цей метод також допомагає скоротити вплив *DDoS* атак на рівні додатків на ваших покупців.

Нарешті, упевнитися, що використовуються розумні значення тайм-аутів в клієнтських бібліотеках і переривники ланцюгів. При використанні розумних тайм-аутів і постійному тестуванні є можливість захистити служби середнього рівня від *DDoS* атак рівня додатків.

Описані вище хмарні рішення підійдуть блогерам, електронним магазинам, малому і середньому бізнесу. У разі, якщо потрібно корпоративна захист або захист дата-центру можна використовувати такі сервіси:

– *ARBOR Networks*;

– *Neustar*;

– *Rackspace*;

– *AKAMAI*;

– *F5 Silverline*;

– *Radware*.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ

2.1. Реалізація методів виявлення та аналізу *DDoS*-атаки за допомогою аналізу журналів

2.1.1. Структура *DDoS*

Перш ніж ми вивчимо способи ідентифікації *DDoS*-атаки, важливо зрозуміти, як вони організовані та працюють. Десятиліття тому декількох машин було достатньо для збою веб-сервера. Тепер із розширеною пропускною здатністю та швидшими комп'ютерними ресурсами зловмисникам потрібні тисячі машин, щоб заповнити сервер трафіком.

Зловмисники використовують бот-мережі, які складаються з тисяч машин-зомбі, які зламують окремі ПК або сервери. На цих ПК встановлено шкідливе програмне забезпечення, яке дає зловмисникові можливість керувати машинами з одного віддаленого місця. Зловмисники можуть встановлювати шкідливе програмне забезпечення на віддалену машину за допомогою зловмисного програмного забезпечення, що входить до комплектуфішингові електронні листи або з використанням веб-сторінок, які називаються «об'ємними сторінками *Java*». Якщо зловмисник може обдурити користувача, дозволивши запустити код *Java,* він може заразити машину різними руткітами та троянами.

Інфіковані зомбі-машини дають повний контроль хакеру. Коли хакер готовий атакувати, він подає сигнал легіонам зомбі-машин, щоб вони затопили певну ціль. Для добре структурованої інфраструктури хакер може зірватися. Однак більшість атак є успішними на певному рівні, або шкодячи продуктивності служби, або порушуючи безпеку.

Хакери також можуть вибрати декілька типів *DDoS,* які вони використовують.Атаки *SYN*найчастіше використовуються при великих атаках.

З меншими атаками компанії можуть додавати більше пропускної здатності та ресурсів сервера, але *DDoS*-атаки продовжують збільшувати пропускну здатність та тривалість. Невеликі власники сайтів купують лише послуги хостингу, які дозволяють кілька тисяч одночасних з'єднань, але зловмисники можуть змоделювати 100 000 з'єднань за допомогою ефективної бот-мережі.

2.1.2. Як виявити активну атаку на вашому сервері

DDoS-атаки швидко починають знищувати продуктивність на сервері. Перший підказка того, що ви зазнали атаки, – це збій сервера. У *IIS* сервер часто повертає помилку 503 "Служба недоступна". Зазвичай вона починає періодично відображати цю помилку, але важкі атаки призводять до постійних відповідей сервера 503 для всіх ваших користувачів.

Інша підказка полягає в тому, що сервер може не повністю вийти з ладу, але послуги стають занадто повільними для виробництва. Надсилання форми або навіть відтворення сторінки може зайняти кілька хвилин.

Якщо є нахил до того, що сервер підданий атаці, чи необхідна його статистика, можна розпочати розслідування за допомогою *Netstat. Netstat* – це утиліта, включена в будь-яку операційну систему *Windows.*

К командному рядку *Windows* необхідно ввести "*netstat –an*". Стандартний виведення повинно виглядати наступним чином, як зображено на рисунку 2.1.

Наведене вище зображення ілюструє вигляд сервера. Бачимо кілька різних *IP*-адрес, підключених до певних портів. Тепер проаналізуємо, як би виглядала *DDoS*-атака, якби на сервер було здійснено атаку (рис. 2.2).

Висновок на основі цього знімка екрану полягає в тому, що той самий *IP* підключається до суміжних портів, і час з'єднання вичерпується. Ми демонструємо лише декілька, але справжня *DDoS*-атака повинна показати сотні з'єднань (іноді тисячі).

Після того, як ви підтвердите, що у вас триває *DDoS*-атака, настав час переглянути журнали сервера. Ви можете витягувати необроблені журнали з *Microsoft IIS,* а можете використовуватижурнал аналізатора.

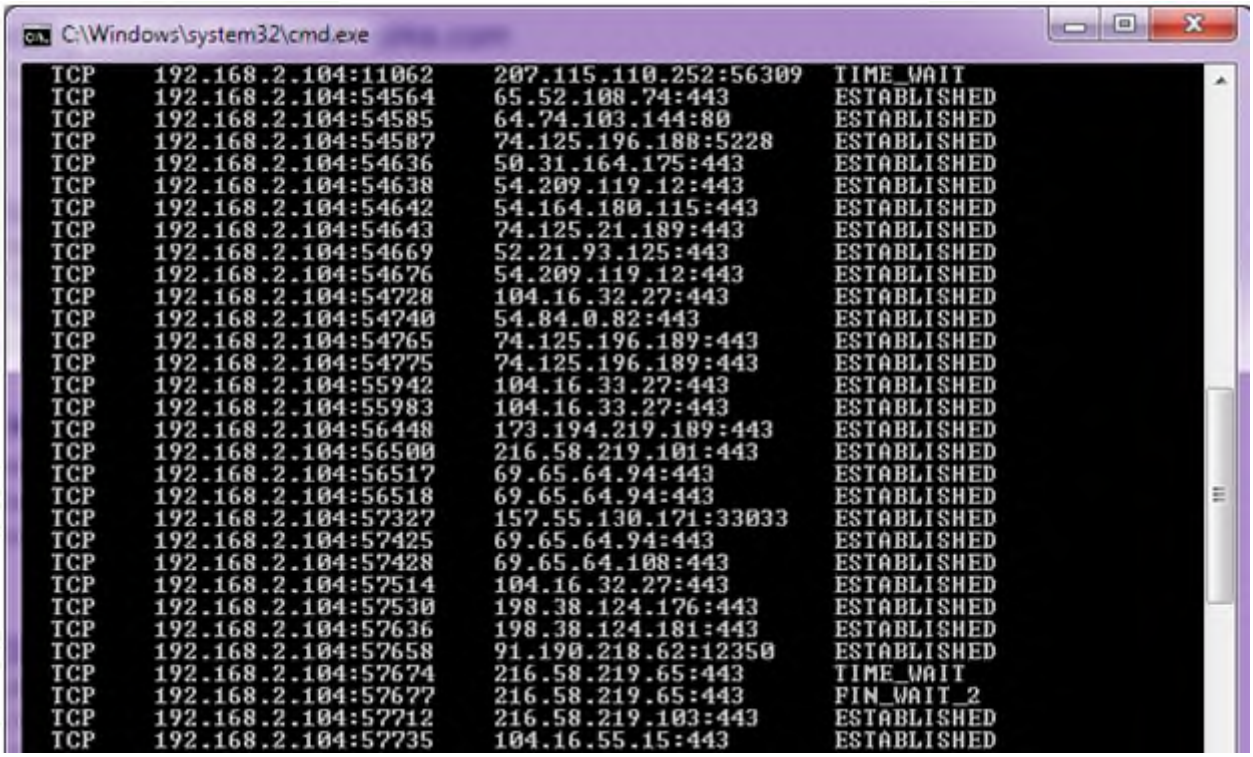

Рис. 2.1. Вигляд серверу

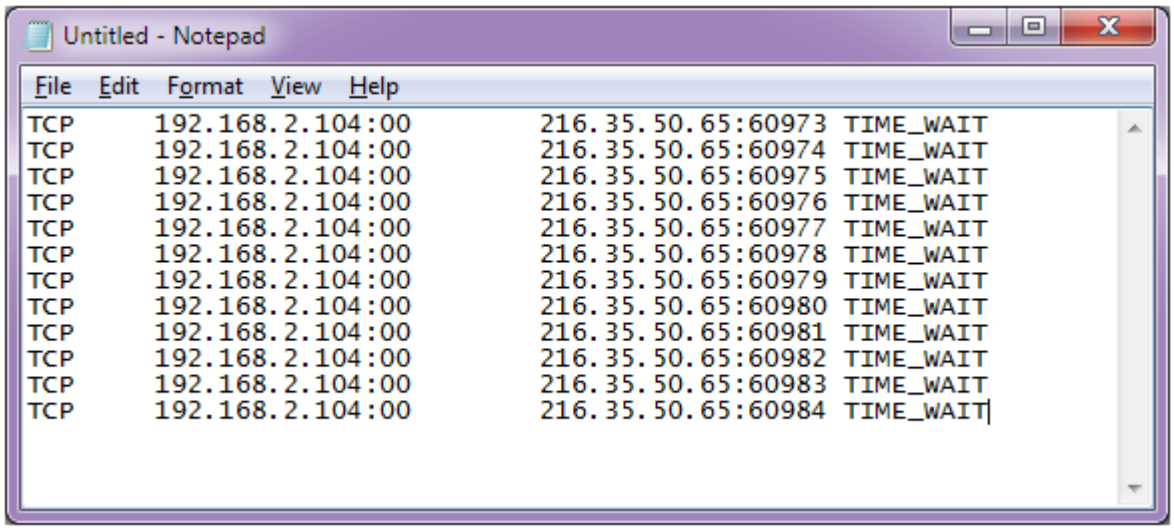

Рис. 2.2. Сервер під *DDoS* атакою

Аналізатори журналів надають візуальну інформацію про веб-трафік. У цьому прикладі маємо *IP*-адресу принаймні для одного зловмисника, але потрібно побачити більшість із них. *Loggly* надає швидку статистику відвідуваності сайту (рис. 2.3).

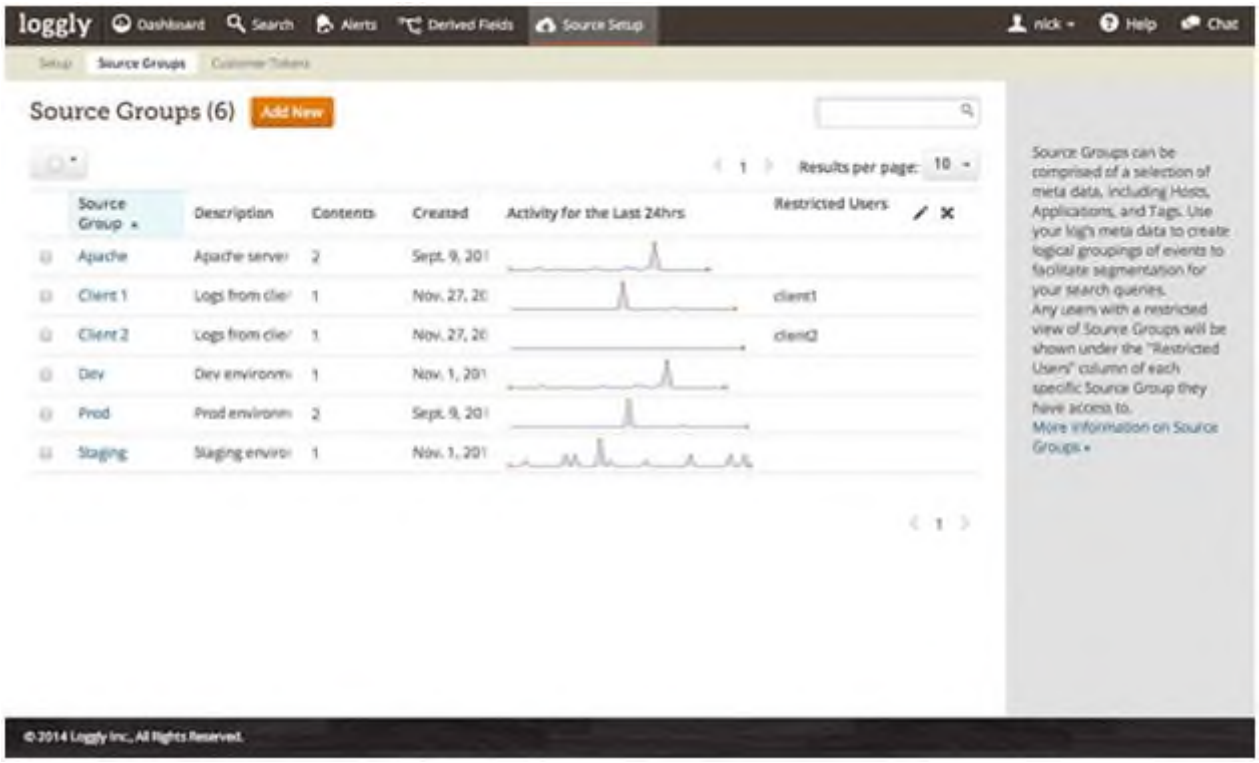

Рис. 2.3. Статистика трафіку у системі *Loggly*

Зображення показує нормальний трафік, але ознака вірусу – це величезні стрибки активності. Це повідомляє про час початку атаки, тому необхідно повернутися до журналів сервера та переглянути *IP*-активність. Оскільки *DDoS*-атака – це неймовірний обсяг трафіку, що надходить на ваш сервер, ви побачите сплеск на відміну від будь-якого дня із великим трафіком, включаючи найзайнятіші часи. По суті, ідеальний час для того, щоб зловмисник завдав удару – це коли користувачі зайняті, оскільки вірус може використовувати як наявний, так і власний трафік, щоб допомогти збити сервер.

Loggly має інструмент для виявлення аномалій. На панелі інструментів вкладки "Тенденції" ви знайдете можливість перегляду аномалій (рис. 2.4).

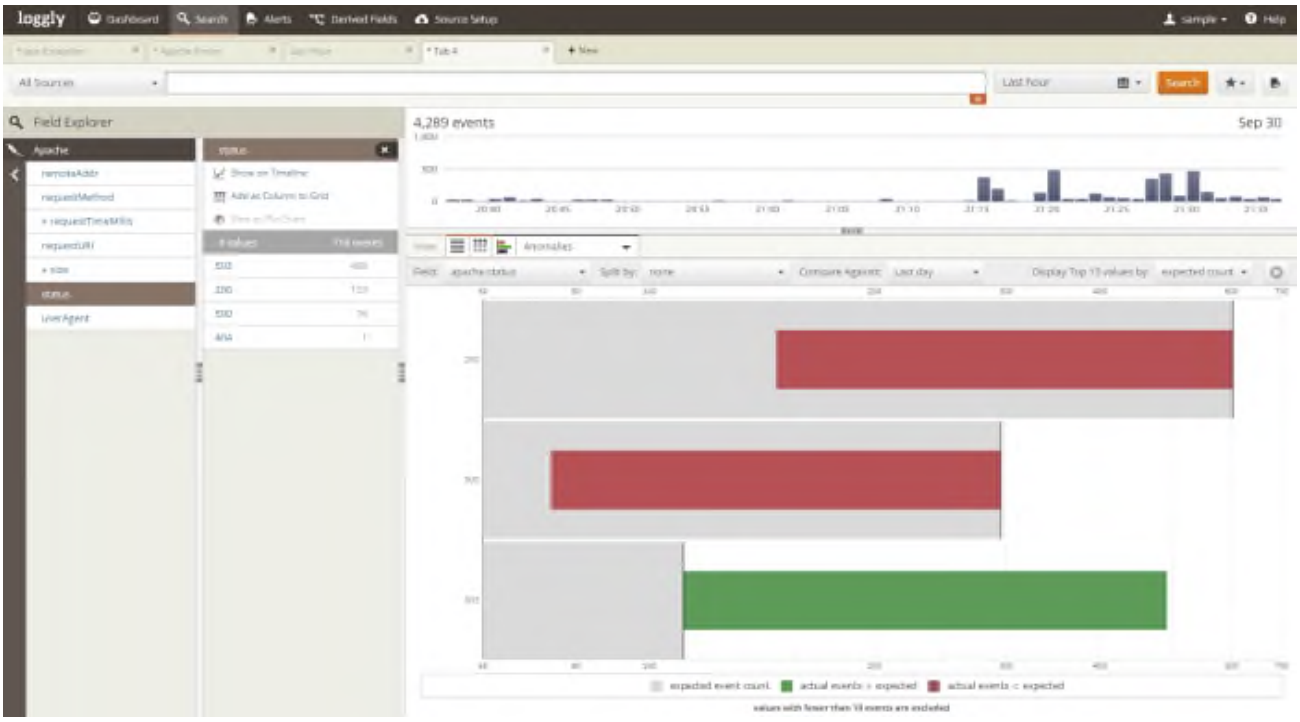

Рис. 2.5. Візуалізація аномалій

На цьому зображенні бачимо, що збільшилось число 503 кодів стану. *DDoS*-атака зазвичай робить сервер *IIS* недоступним, і він відображається як 503 для відвідувачів вашого сайту та у ваших журналах *IIS*.

Наступне зображення є прикладом файлу журналу з *IIS* (рис. 2.6).

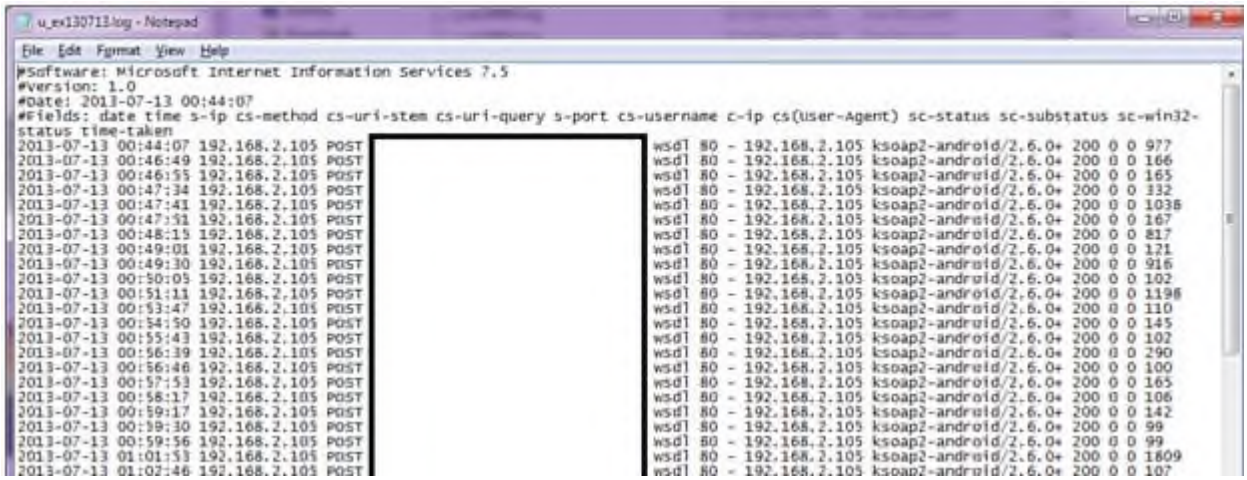

Рис. 2.6. Приклад файлу журналу з *IIS*

Оскільки виявлено стрибок обсягу трафіку в результаті аналізу *Loggly*, тепер можемо ідентифікувати *IP*-адреси в журналах *IIS* на основі часового проміжку атаки.

2.1.3. Методи зупиники атаки

Атакуючі машини, як правило, належать невинним людям, які не знають, що на їхніх комп'ютерах є шкідливе програмне забезпечення. Бажано повідомити про правопорушення у відділі зловживань провайдера зловмисників. Зазвичай електронною поштою для зловживань є "*abuse@<ispname.com>*". Це, принаймні, може зупинити деякі атаки, але це вимагає часу і терміново не допомагає.

Атаки зупиняються на маршрутизаторі. З сервером *Windows* також можна використовувати системний брандмауер, що входить до складу операційної системи. Якщо немає контролю над маршрутизаторами (якщо використовується хмарний хостинг), надзвичайним кроком буде блокування трафіку в брандмауері *Windows* і зв'язок із хостом.

Деякі хмарні провайдери *CDN* пропонують захист *DDoS*. *CloudFlare* є популярною компанією, що займається виконанням та захистом, яка забезпечує хороший захист від навіть складних атак.

Можна вибрати будь-яке програмне забезпечення для виявлення вторгнень, конфігурації маршрутизації та навіть *CDN* для пом'якшення *DDoS*-атак. Однак дуже складні атаки іноді проникають через ці захисні сили. Важливо стежити за вашим трафіком, а *Loggly* допомагає заощадити ваш час, надаючи вам графічне зображення та діаграму, які ви можете використовувати, щоб швидко помітити одну з цих атак. З *Loggly* вам потрібно лише кілька хвилин щодня, щоб переглянути будь-який незвичний трафік.

2.2. Розробка архітектури програмного модуля протидії шкідливим програмам для операційних систем родини *Microsoft Windows*

Сьогоднішній антивірус складається з декількох компонентів, кожен з яких відповідає за виявлення вірусів на певній ділянці роботи.

В процесі розвитку склад цих компонентів змінювався, додавалися нові і віддалялися що не відповідають вимогам часу.

Перший і основний антивірусний компонент - сканер (On-Demand Scanner). Завданням сканера є перевірка на вимогу користувача файлів, пам'яті і завантажувальних секторів на наявність вірусів. Сканер необхідно періодично запускати для перевірки наявних файлів і при отриманні нових.

До недавнього часу практично всі тести антивірусів включали час, за яке антивірусний сканер перевіряв жорсткий диск комп'ютера, і кількість виявлених вірусів. Компанії - розробники антивірусів йшли на різні хитрощі. Наприклад, відома історія, що в антивірусі Dr. Solomon був спеціальний режим для роботи з тестовими колекціями, що дозволяв йому перемагати. Зараз швидкість роботи не є основним показником, головне - це якість виявлення вірусів. Щоб кілька разів не перевіряти файли, які не змінювалися з часу попередньої перевірки, сканери ведуть спеціальну базу даних.

Користувачі часто забувають перевіряти принесені дискети та диски. Саме з цієї причини був розроблений страхувальний компонент антивіруса - монітор (On-Access Scanner).

На відміну від сканера, монітор постійно знаходиться в оперативній пам'яті і автоматично перевіряє файли в реальному часі. Коли користувач намагається запустити програму, монітор перехоплює запит, перевіряє файл на наявність вірусів і, якщо він чистий, дозволяє подальше виконання. Одні монітори контролюють файли тільки в процесі запуску, інші перевіряють всі, з чим працює користувач, а також змінні і закриваються файли.

2.2.1. Розробка функціональної моделі виявлення шкідливих програм

Поширеною помилкою є думка, що антивіруси ловлять шкідливі програми по сигнатурам. Ця думка зазвичай активно експлуатують програми, що реалізують «модні» типи захисту. Не так давно з цієї позиції виступали поведінкові аналізатори, потім настала черга хмарних антивірусів. «Звичайні сигнатурні антивіруси, які споживають ресурси, вимруть, на їх місце прийдуть ...». Так ось, для тих, хто в танку. Чисто сигнатурні антивіруси вимерли в 90-х роках минулого століття, коли з'явилися поліморфні віруси, що змінюються при кожному запуску і, відповідно, не ловлячи по сигнатурам (до речі, саме це стало

причиною появи одного з двох вітчизняних антивірусів - Dr.Web). Тому правильно вимірювати вірусні бази в записах.

Повернемося до кількості створюваних шкідливих файлів в день. Навіщо я все це розповідаю? Справа в тому, що практично на кожній другій конференції питають - а правда, що ви пишете віруси? Більш того, прийняті на роботу співробітники тихо-тихо натякають - ну я вже тепер свій - покажіть, де? Міф настільки укорінився, що до антивірусним компаніям постійно звертаються клієнти з проханням видати щось ще невідоме для тестування. Антивірусні компанії не розсилають ніяких зразків, західні компанії навіть не беруть на роботу колишніх хакерів з огляду на їх моральній нестійкості. Співробітників компаній не розстрілюють при звільненні, і якщо хтось колись дізнається, що компанія ххх створила семпл ...

Міф вкрай небезпечний. Виходячи з нього кількість шкідливих файлів, що випускаються в день, вкрай невелика - ну скільки вендори створять для потрапляння в новини? З міфу виходить, що антивірусні програми повинні знати всі шкідливі програми - або майже все. Але при такому потоці шкідливих файлів (а за різними оцінками на аналіз в антивірусні лабораторії потрапляє не більше 70-80 (і не сильно здивуюся, що оцінка ця завищена) відсотків шкідливих програм з випущених в «дику природу» в цей день) оперативно обробити і миттєво випустити оновлення - не можна.

Функціональна модель на рисунку 2.7 представлена у класичному вигляді, де на вході мережний трафік.

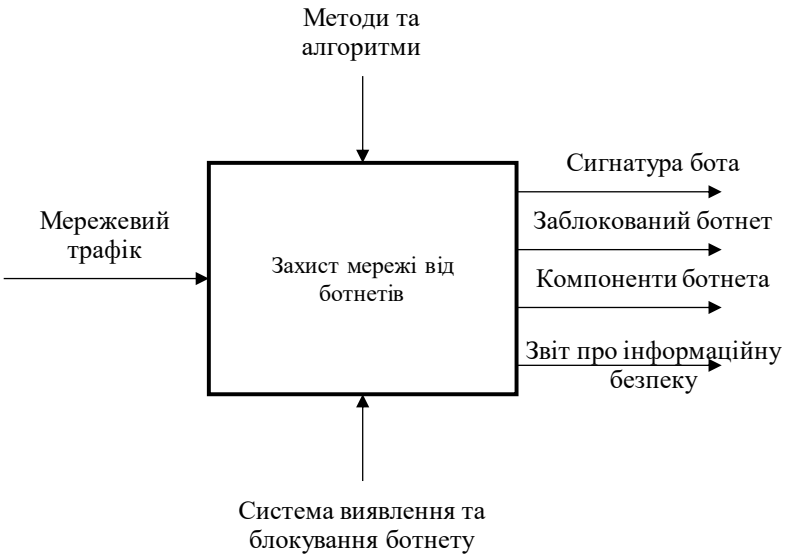

Рис. 2.7. Функціональна модель програмного модуля виявлення шкідливих програм

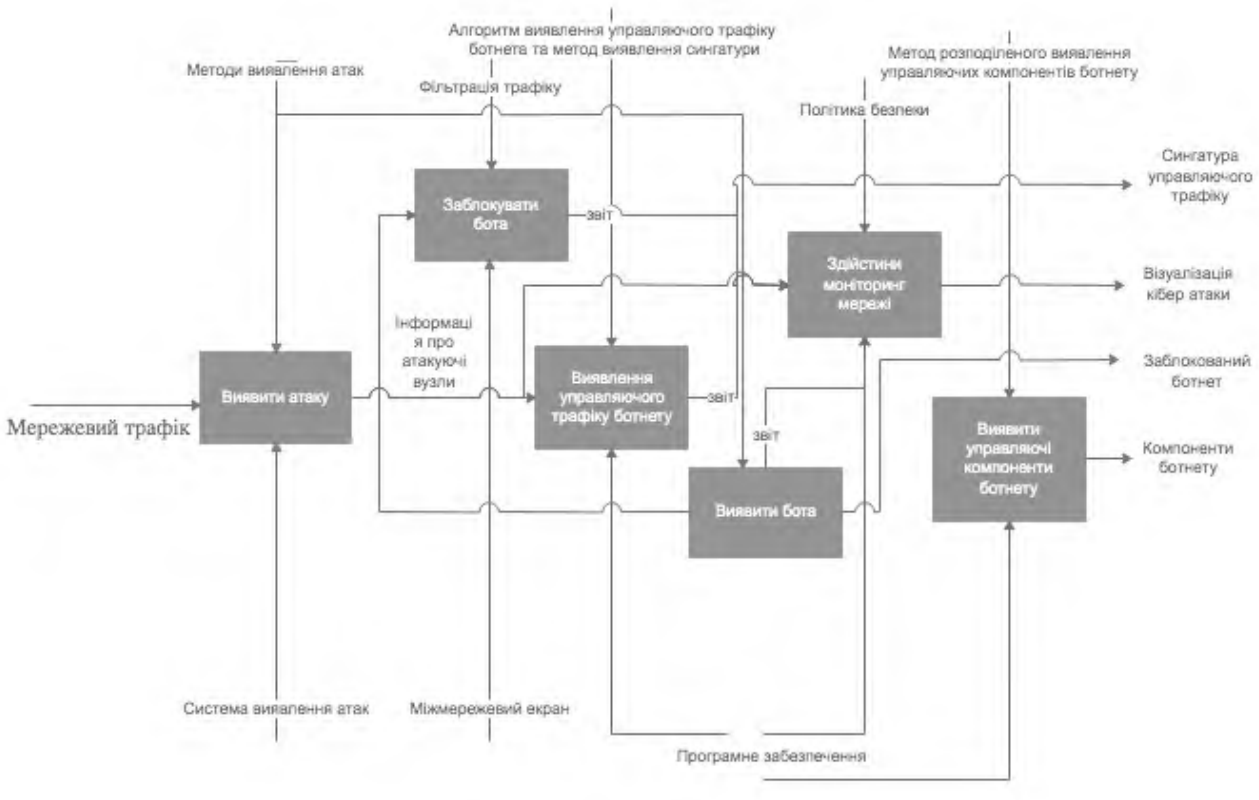

Рис. 2.8. Функціональна модель системи виявлення та блокування ботнетів

2.2.2. Розробка структури системи виявлення ботнетів

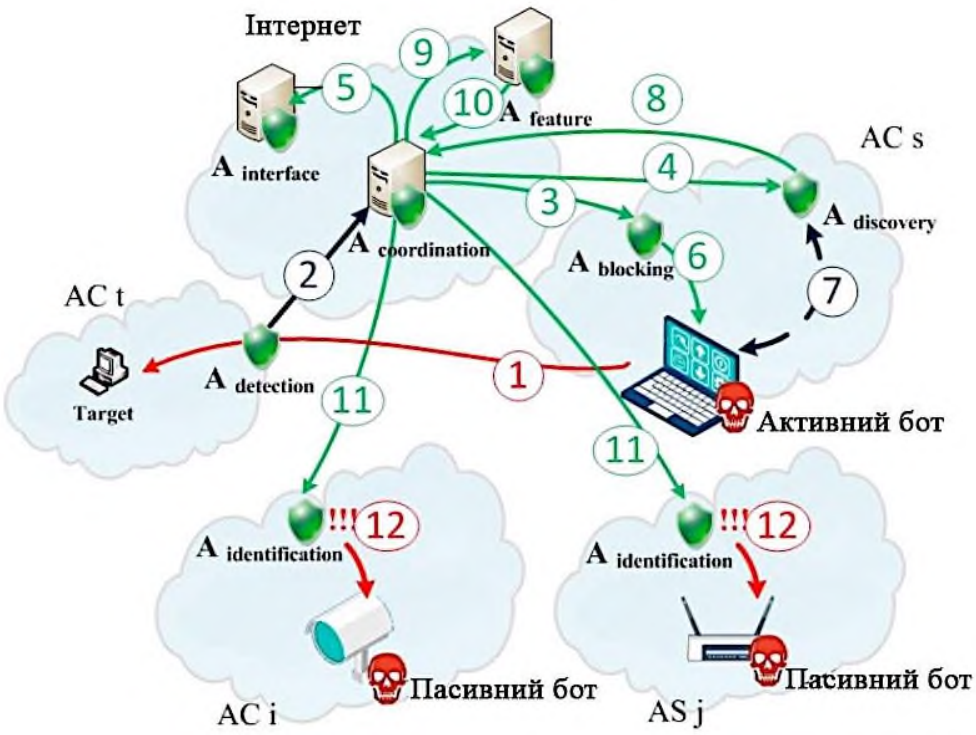

Рис. 2.9. Принцип взаємодії модулів системи виявлення шкідливих програм

2.2.3. Опис моделі агентів та їх кооперації

Рис. 2.11. Моделей системи виявлення виявлення шкідливих програм

2.2.4. Загальна модель програмного модуля виявлення шкідливих програм

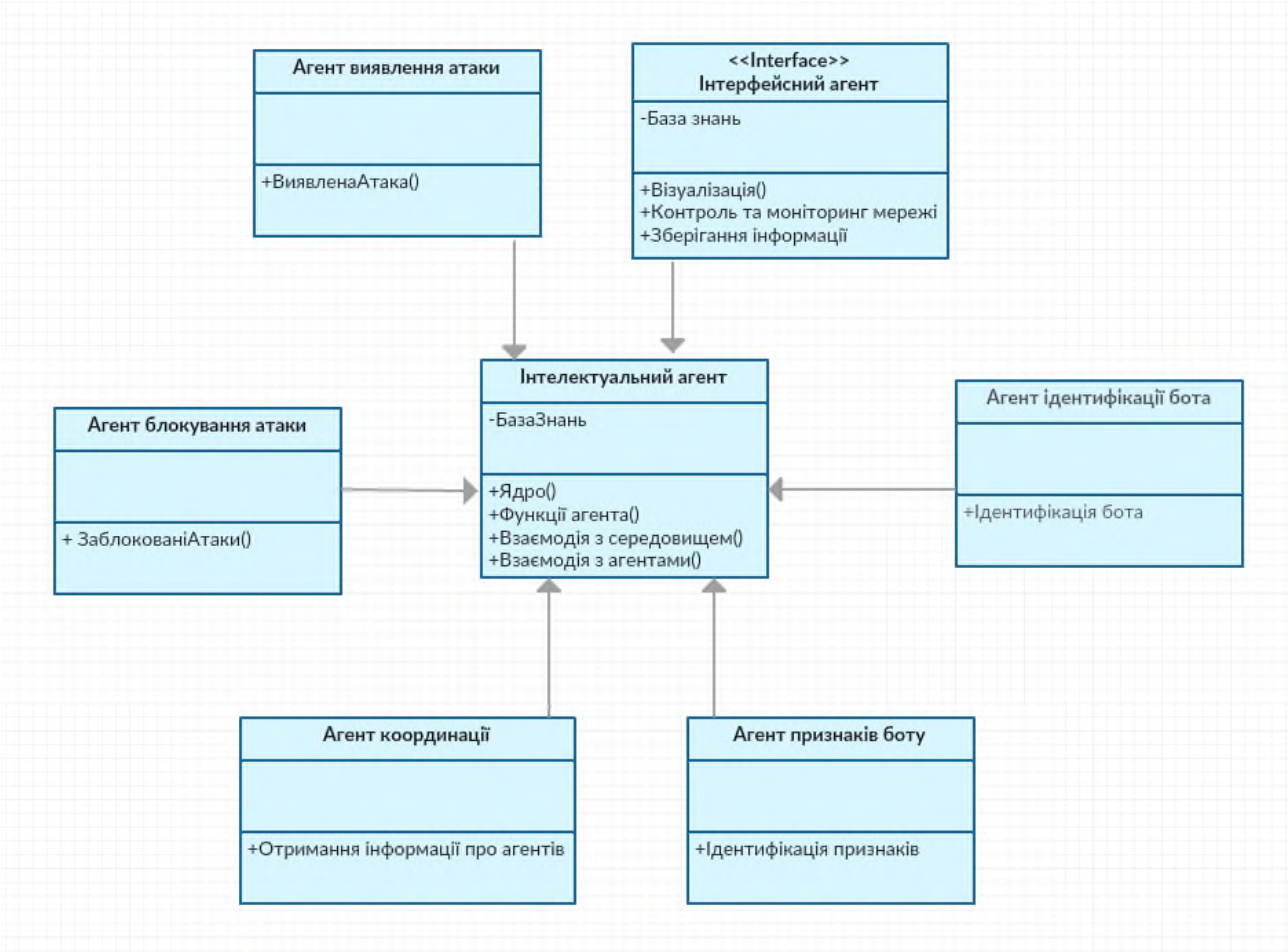

Рис. 2.11. Функціональна структура агента

2.3. Тестування модулів протидії шкідливим програмам

Тестів багато, але найбільш поширені:

- тести антивірусів на великих колекціях
- динамічні тести
- тести на евристики
- тести самозахисту
- тести на швидкодію

Але повернемося до тестів. Дійсно, для тестів на великих колекціях береться колекція за принципом «чим більше, тим краще», беруться антивіруси, які беруть участь в тесті, оновлюються до актуального стану, і запускається сканування колекції на виявлення. В общем-то логічно. Чим більше зразків знає антивірус, тим імовірніше, що він знайде якусь заразу в момент її проникнення до користувача (згадуємо міф про те, що антивіруси повинні знати всі існуючі шкідливі файли).

З точки знання звичайного користувача і, на жаль, більшості замовників - все чисто і питань немає. Але невідомі шкідливі програми псують таку красиву картинку.

Антивірус має лікувати, і лікувати живі віруси - внедрившиеся в систему користувача. Перевіряти, чи знайде антивірус щось шкідливе в файлі, теж потрібно, але в першу чергу не для сканера і файлового монітора систем, призначених для захисту робочих станцій, а для:

- систем для захисту поштових серверів і шлюзів - тільки там йде робота з вірусами виключно в їх неактивованому вигляді;
- компонентів перевірки поштового і інтернет-трафіку антивірусів для робочих станцій / файлових серверів (які повинні бути, але не з них виконують своє призначення (традиційне обіцянка - це буде описано в статті, присвяченій технологіям)). Але в тестах для робочих станцій, як правило, ці компоненти не перевіряються.

У подібних тестах перевіряється робота не систем виявлення (перехоплення запуску, наприклад), а виключно антивірусне ядро. Колекція або безпосередньо перевіряється антивірусним сканером, або файли з колекції копіюються скриптом, і контролюється їх виявлення файловим монітором. Тут цілих три підводних каменя.

По-перше, як уже говорилося, антивірус повинен лікувати. Але при великих обсягах колекції фізично неможливо перевірити, чи дійсно файл проліковано і проліковано він правильно - тобто повністю чи він працездатний після проведеного лікування. Відповідно, в разі лікування довелося б просто покладатися на чесність виробника і його повідомлення про успішність

лікування. Як результат, тестування на величезних колекціях проводяться на виявлення.

Невеликий відступ. Виявлення виявлення ворожнечу. Наприклад, не всі віруси (зараз ми говоримо про віруси яка про клас шкідливих програм) заражають файли правильно. Скажімо, аналітики «Доктор Веб» вважають, що не потрібно повідомляти про детектив в тому випадку, якщо вірус заразив файл так, що він ніколи не буде виконаний, з одного боку, а з іншого - не впливає на працездатність інфікованої програми. Програш в тесті - однозначно. Але чи знають користувачі, що програла програма піклується про них, так як результат лікування не завжди передбачуваний?

Ще один відступ і ще приклад. Поліморфні віруси - нині рідкість, так. Але тим не менше. Поліморфні віруси не визначаються по сигнатурам - вони змінюються на кожен запуск. Як їх зловити в тесті, якщо в антивірусі немає поліморфного аналізатора? Знайти всі наявні у тестувальників зразки і занести в бази. Все одно ніхто запускати їх не буде для отримання нових зразків. Реальний випадок з минулого.

По-друге, антивірус повинен лікувати в реальних умовах. Реальні віруси, які проникли в систему, активно протидіють антивірусів - шифруються, застосовують методи поліморфного зміни і обфускації, маскуються під цілком нешкідливі процеси і т.д. Видалити їх в рази складніше, ніж виявити в файлі. І тим складніше їх видалити так, щоб не пошкодити систему. Знову ж в силу величезного розміру колекцій і неможливості набрати необхідну кількість фахівців, особисто перевіряють якість лікування по кожному вірусу, провести тести на лікування неможливо. Знову ж доводиться вірити на слово виробникам.

По-третє, антивірус повинен виявляти тільки шкідливі файли. В силу відсутності потрібної кількості фахівців, здатних перевірити всю колекцію і гарантують відсутність в ній різного сміття, що не відноситься до шкідливих об'єктів, знову ж доводиться вірити на слово продукту, що він виявив вірус, а не уламок реєстру. І щоб перемогти в тесті, потрібно занести все це сміття в бази. З одного боку, бази роздуваються, а з іншого - як можна перемогти, якщо заздалегідь не отримати весь цей мотлох?

Тести на величезних колекціях:

- відкривають широкий простір для зловживань недобросовісних виробників, в рекламних цілях завищують відсоток виявлень;
- не показують, наскільки добре поводиться антивірус в реальних умовах бойового зіткнення з вірусом;
- не тестується всі компоненти антивіруса - що, до речі, вкрай важливо, тому що сучасні віруси в момент виходу тестуються на актуальних версіях антивіруса і до надходження зразків у лабораторію не виявляються ніким.

Забезпечити захист може тільки обмеження шляхів проникнення вірусів в систему - офісний контроль, закриття портів і т. Д.

Наскільки чесно пишатися такими перемогами по відношенню до своїх користувачів? Філософське питання.

Широке поширення тестів на величезних колекціях провокує виробників на створення «антивірусів», які:

- орієнтовані на виявлення тільки тих зразків, які зустрічаються в тестах;
- виявляють 100% вірусів в будь-якої колекції, навіть якщо в ній немає вірусів;
- заточені тільки на виявлення і видалення самих шкідливих програм.

Різновидом тесту на великих колекціях є тести на евристики. Виходячи з того, що антивірус повинен ловити ще не надійшли в лабораторію зразки, кожен антивірус містить механізми виявлення ще невідомих шкідливих програм. З цією метою стан антивіруса фіксується на дату, сильно віддалену назад від часу проведення тесту, а використовувана колекція містить віруси, що з'явилися після цієї дати. Такий тест добре показує, наскільки добре антивірус ловить варіанти вже відомих вірусів, але, як уже говорилося, сучасні віруси, в тому числі спрямовані на крадіжку фінансової інформації, перед релізом перевіряються на актуальних антивірусних базах - і високий рейтинг продукту в даному тесті не рятує користувача від втрати своїх грошей і даних.

Ще один вид тесту - динамічний. Встановлюється актуальна версія антивіруса, і імітується робота користувача. Проблем кілька:

- Типові користувачі не займаються налаштуванням безпеки (скажімо, батьківського контролю і поведінкового аналізатора). Відповідно, шкідливі файли, протестовані на невиявлення типовий установкою, пройдуть без проблем.
- Стандартних профілів роботи користувача в мережі особисто мені не відомо. Якщо я не правий - прохання поправити.
- Соціальна інженерія творить чудеса. Як приклад останнього часу - листи від податкової інспекції із зазначенням справжніх імен (привіт захист персональних даних!).

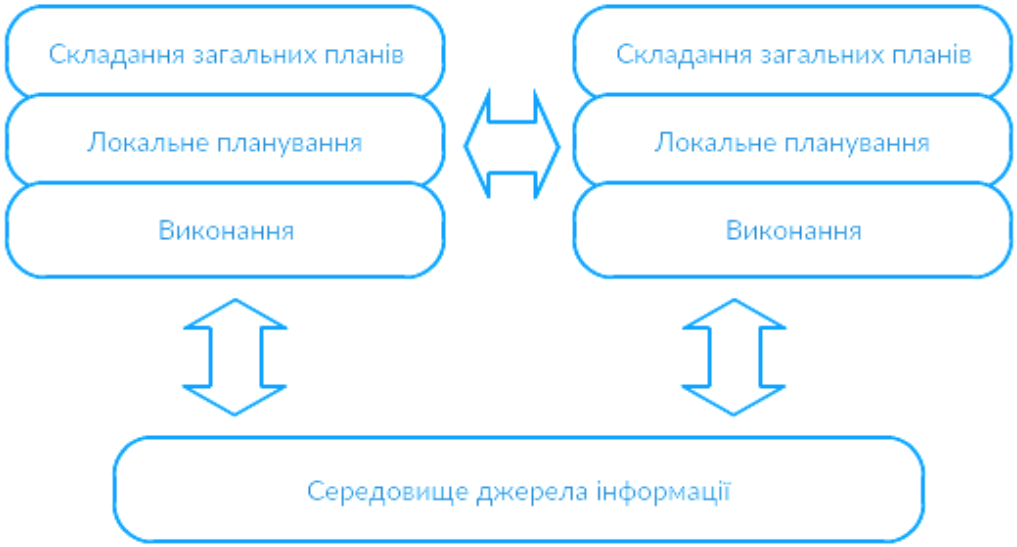

Рис. 2.12. Базовий набір функцій програмного модуля виявлення шкідливих програм

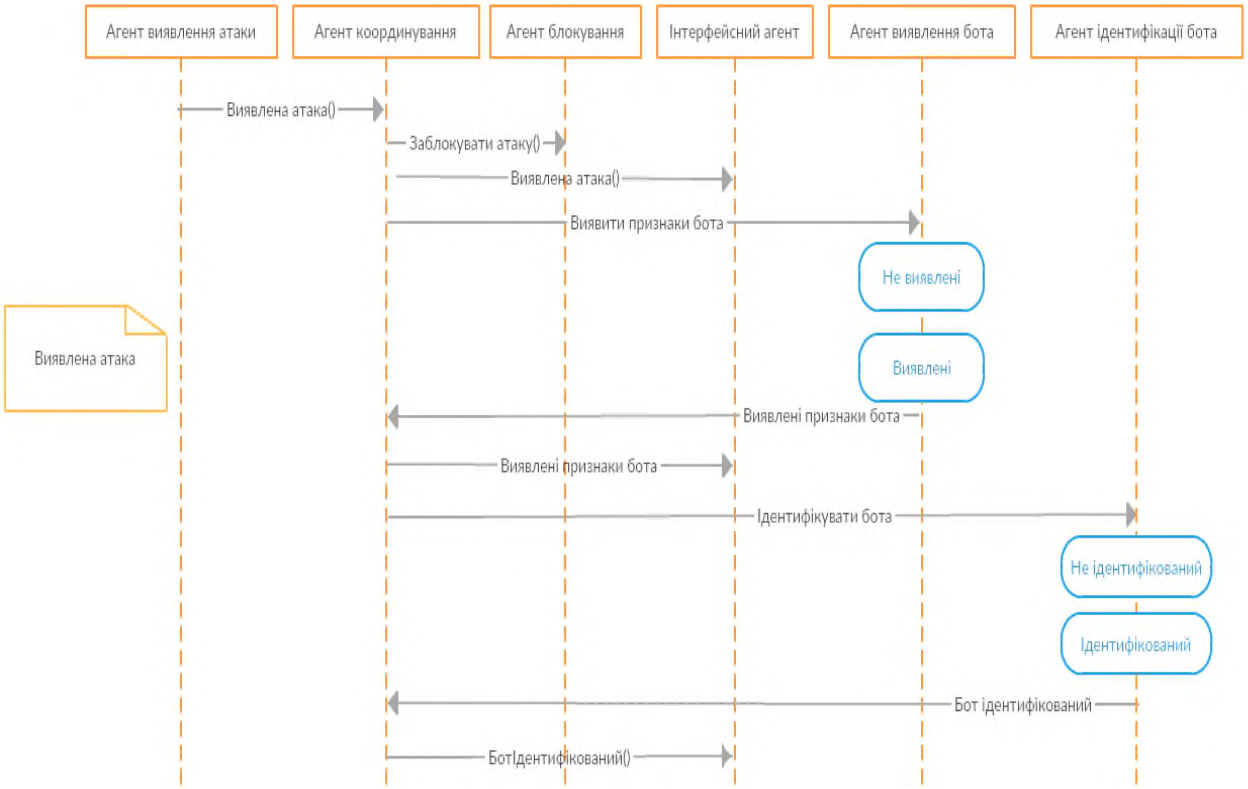

Рис. 2.13. Діаграма послідовностей взаємодій при виявленні шкідливих програм

2.4. Проактивні методи антивірусного захисту

Класичні проактивні системи виявлення шкідливих програм, незважаючи на уявну простоту реалізації і надійність, мають ряд суттєвих недоліків, а саме:

– слабка ефективність проти загроз типу 0-*day*, так як ефективність безпосередньо пов'язана з базою сигнатур шкідливого ПЗ, в яку внесено сигнатури тільки відомого, на даний момент, шкідливого ПЗ;

– необхідність постійного оновлення бази сигнатур вірусів для ефективного захисту від нового шкідливого ПЗ;

– Для визначення шкідливого ПЗ необхідна процедура сканування, яка забирає досить багато часу і системних ресурсів;

Дані причини і послужили поштовхом до розвитку т.зв. проактивних систем захисту, про які й піде мова в даній статті.

Методи проактивного захисту

Історія розвитку проактивних методів захисту

Проактивні методи захисту з’явилися майже одночасно з реактивними методами захисту, як і системи, які застосовують ці методи. Але в силу недостатньої «дружності» проактивних систем захисту по відношенню до користувача розробка подібних систем була припинена, а методи забуті. Відносно недавно проактивні методи захисту почали своє відродження. На сьогоднішній день методи проактивного захисту інтегруються в антивірусні програми, раніше використовували лише реактивні системи захисту, а так само на основі проактивних методів створюються нові антивірусні системи, що комбінують декілька методів проактивного захисту, і що дозволяють ефективно протистояти новітнім загрозам.

На сьогоднішній день найбільш відомі і часто застосовуються такі методи проактивного захисту:

- Методи поведінкового аналізу;
- Методи обмеження виконання операцій;
- Методи контролю цілісності ПЗ і ОС.

Методи запобігання вторгнень (*IPS*-методи)

HIPS

HIPS – метод контролю активності, заснований на перехопленні звернень до ядра ОС і блокування виконання потенційно небезпечних дій ПЗ, що працює в *user-mode*, виконуваних без відома користувача;

Принцип роботи

HIPS-система за допомогою власного драйвера перехоплює всі звернення ПЗ до ядру ОС. У разі спроби виконання потенційно небезпечного діяння з боку ПЗ, *HIPS*-система блокує виконання даної дії і видає запит користувачеві, який вирішує дозволити або заборонити виконання даної дії.

Переваги систем, побудованих на методі *HIPS*:

- Низьке споживання системних ресурсів;
- Чи не вимогливі до апаратного забезпечення ПК (можуть працювати на різних платформах);

- Висока ефективність протистояння загрозам 0-*day;*
- Висока ефективність протидії руткитам, які працюють в *user-mode*;

Недоліки систем, побудованих на методі *HIPS*:

- Низька ефективність протидії руткитам, які працюють в *kernel-mode*;
- Велика кількість звернень до користувача ПК;
- Користувач повинен мати знання про принципи функціонування ОС;
- Неможливість протидії активному зараження ПК;

VIPS

VIPS – метод контролю активності, заснований на моніторингу виконуваних операцій ПЗ, встановленому на ПК, і блокування виконання потенційно небезпечних дій ПЗ, виконуваних без відома користувача.

Принцип роботи

VIPS-система за допомогою технологій апаратної віртуалізації (*Intel VT-x, AMD-V)* запускає ОС в «віртуальній машині» і здійснює контроль всіх виконуваних операцій. У разі спроби виконання потенційно небезпечного діяння з боку ПЗ, *VIPS*-система блокує виконання даної дії і видає запит користувачеві, який вирішує дозволити або заборонити виконання даної дії.

Переваги систем, побудованих на методі *VIPS*:

- Низьке споживання системних ресурсів;
- Висока ефективність протистояння загрозам 0-*day;*
- Висока ефективність протидії руткитам, які працюють і в *user-mode*, і в *kernel-mode*;

Недоліки систем, побудованих на методі *VIPS*:

- Вимогливі до апаратного забезпечення ПК (для роботи *VIPS*-системи необхідна апаратна підтримка процесором технологій апаратної віртуалізації (*Intel VT-x* або *AMD-V)*;
- Велика кількість звернень до користувача ПК;
- Користувач повинен мати знання про принципи функціонування ОС;
- Неможливість протидії активному зараження ПК;

Пісочниця (*sandbox)*

Пісочниця (*sandbox)* – метод, заснований на виконанні потенційно небезпечного ПЗ в обмеженому середовищі виконання (т.зв. «пісочниці»), яка допускає контакт потенційно небезпечного ПЗ з ОС.

Принцип роботи

Пісочниця розділяє встановлене ПЗ на дві категорії: «Довірені» і «Не довірені». ПЗ, що входить до групи «Довірені» виконується без будь-яких обмежень. ПЗ входить до групи «Не довірені» виконується в спеціальній обмеженою середовищі виконання (т.зв. пісочниці), даним ПЗ заборонено виконання будь-яких операцій, які можуть привести до краху ОС.

Переваги систем, побудованих на методі пісочниця (*sandbox)*:

– Низьке споживання системних ресурсів;

– Чи не вимогливі до апаратного забезпечення ПК (можуть працювати на різних платформах);

– Мала кількість звернень до користувача ПК;

Недоліки систем, побудованих на методі пісочниця (*sandbox)*:

– Користувач повинен мати знання про принципи функціонування ОС;

– Неможливість протидії активному зараження ПК;

Методи поведінкового аналізу

Поведінковий блокує (метод активного поведінкового аналізу)

Поведінковий блокує (метод активного поведінкового аналізу) – метод, заснований на методах *IPS,* аналізу в реальному часі ланцюжків дій ПЗ і блокування виконання потенційно небезпечних алгоритмів в реальному часі.

Принцип роботи

Різні проактивні системи захисту використовують різні концепції реалізації методу активного поведінкового аналізу (тому що метод активного поведінкового аналізу може бути побудований на базі будь-якого *IPS*-методу). У загальному і цілому метод активного поведінкового аналізу являє собою *IPS*-метод з інтелектуальною системою прийняття рішень, яка аналізувала, не окремі дії, а ланцюжка дій.

Переваги систем, побудованих на методі активного поведінкового аналізу:

– Менша кількість звернень до користувача, в порівнянні з системами, побудованими на *IPS*-методах;

– Низьке споживання системних ресурсів;

– Чи не вимогливі до апаратного забезпечення ПК (можуть працювати на різних платформах);

Недоліки систем, побудованих на методі активного поведінкового аналізу:

– Інтелектуальна система винесення вердиктів може викликати «помилкові спрацьовування» (блокування роботи нешкідливого ПЗ, через здійснення операцій, схожих з діяльністю шкідливого ПЗ);

– Неможливість протидії активному зараження ПК;

Метод емуляції системних подій (метод пасивного поведінкового аналізу)

Методи емуляції системних подій (метод пасивного поведінкового аналізу) – метод визначення шкідливого ПЗ шляхом аналізу дій і / або ланцюжка дій за допомогою виконання ПЗ в спеціальній обмеженою середовищі (т.зв. емуляторі коду), що імітує реальне апаратне забезпечення.

Принцип роботи

ПЗ запускається в спеціальній обмеженою середовищі (т.зв. емуляторі коду), що імітує реальне апаратне забезпечення, де проводиться перевірка на виконання певних дій і / або ланцюжка дій. Якщо виявлена можливість виконання потенційно небезпечного діяння і / або ланцюжка дій, ПЗ позначається як шкідливе.

Переваги систем, побудованих на методі активного поведінкового аналізу:

– Не вимагають спеціальних знань або навичок з боку користувача;

– Чи не вимогливі до апаратного забезпечення ПК (можуть працювати на різних платформах);

– Відсутність звернень до користувача, за винятком випадків виявлення шкідливого ПЗ;

Недоліки систем, побудованих на методі активного поведінкового аналізу:

– У деяких випадках аналіз коду може займати досить тривалий час;

– Широке застосування методик протидії емуляції коду шкідливого ПЗ, тому системи, що використовують метод пасивного поведінкового аналізу, можуть протистояти не всім видам шкідливого ПЗ;

– Неможливість протидії активному зараження ПК.

сканер цілісності

Сканер цілісності здійснює постійний моніторинг ядра ОС, на предмет виявлення змін, які могло призвести шкідливе ПЗ. У разі виявлення змін внесених шкідливим ПЗ про це сповіщається користувач і по можливості проводиться відкат дій, вироблених шкідливими ПЗ.

Принцип роботи

Сканер цілісності здійснює моніторинг всіх звернень до ядра ОС. У разі виявлення спроби зміни критично важливих параметрів, операція блокується.

Переваги систем, побудованих на методі «сканер цілісності»:

– Не вимагають спеціальних знань або навичок з боку користувача;

– Чи не вимогливі до апаратного забезпечення ПК (можуть працювати на різних платформах);

– Мала кількість звернень до користувача;

Недоліки систем, побудованих на методі «сканер цілісності»:

– Для здійснення контролю цілісності необхідно контролювати велику кількість різних параметрів, що може негативно позначитися на продуктивності ПК;

– Слабка ефективність протидії *user-mode* і *kernel-mode* руткитам;

– Неможливість протидії активному зараження ПК;

Метод запобігання атак на переповнення буфера («емулятор *NX*-біта»)

«Емулятор *NX*-біта» здійснює моніторинг вмісту оперативної пам’яті. У разі виявлення спроби внесення критичних змін в вміст оперативної пам’яті, дія блокується.

Принцип роботи: суть атак на переповнення буфера полягає в навмисному переповненні оперативної пам’яті і як наслідок виведення з ладу ПК, на певний час. «Емулятор *NX*-біта» створює спеціальну зарезервовану область оперативної

пам’яті, куди записувати дані неможливо. У разі виявлення спроби записи даних в зарезервовану область пам’яті, «емулятор *NX*-біта» блокує дію, таким чином, запобігаючи виведенню ПК з ладу.

Переваги систем, побудованих на методі «емулятора *NX*-біта»:

– не вимагають спеціальних знань або навичок з боку користувача;

– чи не вимогливі до апаратного забезпечення ПК (можуть працювати на різних платформах);

– повна відсутність яких би то не було звернень до користувача;

Недоліки систем, побудованих на методі «емулятора *NX*-біта»:

– системи, побудовані на методі «емулятора *NX*-біта», можуть протистояти тільки проти хакерських атак на переповнення буфера, будь-яким іншим видам загроз дані системи протистояти не можуть;

– неможливість протидії активному зараженню ПК.

2.5. Метод розподіленого виявлення керуючих компонентів шкідливих програм

В силу простоти алгоритму перевірки наш сканер зможе виявляти тільки шкідливі програми, що поширюються цільним файлом, тобто не заражають інші файли, як PE-Віруси, і не змінюють своє тіло в процесі діяльності, як поліморфні віруси.

Втім, це відноситься до більшості вірусів, хробаків і практично до всіх троянам, тому написаний нами сканер має право на життя :)

Навіщо потрібен антивірус?

Відповідно, система антивірусного захисту повинна забезпечувати:

захист від проникнення всіх уже відомих типів шкідливих програм (в тому числі за допомогою технологій, що дозволяють виявляти модифікації раніше знайденого). Слово «всіх» виділено не просто так - типовою є прохання видалити з баз старі віруси. Не повірите - OneHalf ще живий!

після отримання оновлень - виявлення і знищення (але не відкат дій!) вже запущених і активно протидіють виявлення шкідливих програм.

Визначення показує, що дані в нормативних документах визначення антивірусного захисту не просто хибні, а свідомо шкодять компаніям, які орієнтуються на дані визначення. Тому і реалізовані на основі цих визначень функції, і склад систем антивірусного захисту виявляються також невірними.

Коментарі до попередніх статей показують, що багато, визначаючи завдання антивірусного захисту, забувають про другу частину.

Виконати завдання щодо запобігання впровадження шкідливих програм можна і без антивіруса - ніхто вам не заважає, і, більш того, іноді наявність антивіруса протипоказано. Але ось видалення вже активної зарази без антивіруса неможливо. Так, я знаю про наявність фахівців, які можуть зробити це вручну (і навіть є компанії, які формують такі групи швидкого реагування). Але є три але:

Більшість користувачів на таке не здатні;

сучасні шкідливі програми часто розраховані на тривалий непомітне присутність в системі (і більш того, можуть закривати уразливості, видаляти інші віруси і навіть встановлювати антивірус). Протестовані на системах захисту, вони можуть залишатися непоміченими роки;

антивірус при наявності знань про шкідливу програму видалить її швидше.

У чому різниця між антивірусом і антивірусної системою захисту?

Система антивірусного захисту - це не завжди антивірус. Це будь-яка програма / настройка ОС / процедура, за допомогою якої ви знижуєте ризик зараження.

Відповідно, ніхто вам не заважає (крім регуляторів, а й там все не так очевидно) не використовувати антивірус для запобігання зараження, але для лікування без антивіруса не обійдешся. Але є одне але (www.infosec.ru/news/8119):

Найбільш поширені уразливості і недоліки:

Чи не налаштовані або некоректно налаштовані парольні політики.

Адміністрування мережевого обладнання за допомогою небезпечного протоколу TELNET.

Аудит подій не налаштований або налаштований некоректно.

Міжмережеві екрани містять надлишкові правила.

Некоректно проведена сегментація мережі, зокрема серверні сегменти не відокремлені від призначених для користувача сегментів.

Чи не реалізований або некоректно реалізований процес управління оновленнями, в результаті чого, наприклад, використовує дуже застаріле, що містить відомі уразливості

Відсутність налаштувань безпеки комутаторів доступу, зокрема, захисту від вразливості до атак класу «ARP Cache Poisoning».

Відсутність оновлення сигнатур і налаштувань оповіщення для засоби виявлення вторгнень.

Використання небезпечних протоколів для віддаленого доступу за допомогою технології VPN.

Некоректно налаштовані обмеження прав доступу до системних файлів.

2.6. Висновки до розділу

Для захисту робочих станцій і файлових серверів антивірусне рішення повинно:

– мати систему самозахисту, що не дозволяє невідомої шкідливій програмі порушити нормальну роботу антивіруса, - антивірусне рішення повинно нормально функціонувати до надходження оновлення, що дозволяє пролікувати зараження;

– як система управління, так і система оновлення антивірусного рішення повинні бути незалежні від відповідних механізмів, які використовуються в операційних системах (ніяких WSUS і Windows Update) і включені в систему самозахисту антивіруса, що дозволяє виключити можливість перехоплення системи поновлення шкідливою програмою. Взагалі число компонентів операційної системи, незалежних бібліотек і т. Д., Які використовуються антивірусом, але природно, не розміщені під захист системи самозахисту - має бути мінімальним;

– система управління антивірусним захистом повинна забезпечувати максимально швидке отримання оновлень захищеними робочими станціями і серверами;

– антивірусна система повинна покладатися в першу чергу на технології антивірусного ядра - з працею піддаються обходу шкідливими програмами, а не на зовнішні компоненти, які так чи інакше можуть бути скомпрометовані внаслідок невірного використання та / або використання вразливостей в використовуваних спільно з ними зовнішніх програмах або бібліотеках (це положення ми розберемо, коли будемо говорити про переваги і недоліки різних технологій і вплив ЗМІ на рівень захищеності).

РОЗДІЛ 3
ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНОГО МОДУЛЯ ПРОТИДІЇ ШКІДЛИВИМ ПРОГРАМАМ ДЛЯ ОПЕРАЦІЙНИХ СИСТЕМ РОДИНИ *MS WINDOWS*

Перш ніж ввести визначення системи антивірусного захисту, ще раз визначимо можливості шкідливих програм по обходу систем антивірусного захисту (рівень ризику).

На даний момент найбільш небезпечні шкідливі програми розробляється не хакерами-одинаками – це добре організований кримінальний бізнес, який втягує в свою злочинну діяльність висококваліфікованих системних і прикладних розробників ПЗ.

Тестування на невиявлення розроблюваних шкідливих програм актуальними антивірусними рішеннями забезпечило можливість випуску великої кількості шкідливих програм, які не виявляються (до отримання оновлень) системами захистів, імовірно використовуваних групами користувачів, на яких планується атака – в тому числі за допомогою евристичних механізмів. Число таких програм, що випускаються одним угрупованням, може досягати сотень зразків – і жоден з них не буде виявлятися антивірусним ПЗ, використовуваним цільовою групою жертв.

Основні дві причини пропуску вірусів антивірусним ядром це::

– затримка виявлення нової шкідливої програми – дельта між її попаданням до жертви і надходженням на аналіз;

– затримка обробки в зв’язку з великою кількістю надійшли на аналіз програм / неправильною оцінкою небезпеки програми в момент виявлення.

Але ці дві проблеми – можуть бути вирішені. Збільшується кількість аналітиків, до надходження аналітикам потік аналізується спеціальним роботом, автоматично формує записи в бази і т.д. Частина шкідливих програм, що не потрапили на аналіз, відловлюється евристиками.

3.1. Оцінювання ступеня протидії шкідливим програмам

Перш ніж починати писати код, варто визначити всі складові частини сканера і то, як вони будуть взаємодіяти (рис. 3.1).

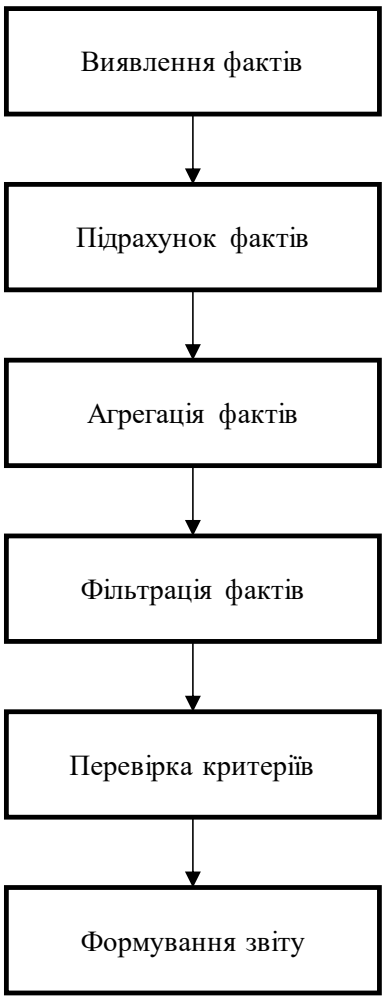

Рис. 3.1. Рівні роботи програмного модуля протидії шкідливим програмам

В силу простоти алгоритму перевірки наш сканер зможе виявляти тільки шкідливі програми, що поширюються цільним файлом, тобто не заражають інші файли, як *PE*-Віруси, і не змінюють своє тіло в процесі діяльності, як поліморфні віруси.

Втім, це відноситься до більшості вірусів, хробаків і практично до всіх троянам, тому написаний програмний модуль у достатній мірі виконує свою функцію.

Сигнатура в простому поданні є унікальною частиною (послідовністю байт) в файлі. Однак ця частина повинна максимально однозначно виділяти файл

серед безлічі інших файлів. Це означає, що обрана послідовність має бути присутня тільки одному файлі, якому вона належить, і ні в яких інших.

На практиці крім самої послідовності застосовуються ще додаткові параметри, що дозволяють максимально однозначно зіставляти сигнатуру файлу. Введення додаткових параметрів також направлено на прискорення пошуку сигнатури в файлі.

Такими параметрами, наприклад, можуть бути розмір файлу, зсув послідовності байт, тип файлу, спеціальні маски (відображення того, як приблизно має виглядати файл, щоб в ньому могла міститися шукана сигнатура) і багато інших.

У розробленому програмному модулі (рис. 3.2) в якості додаткового параметра будемо використовувати зміщення послідовності в файлі щодо початку.

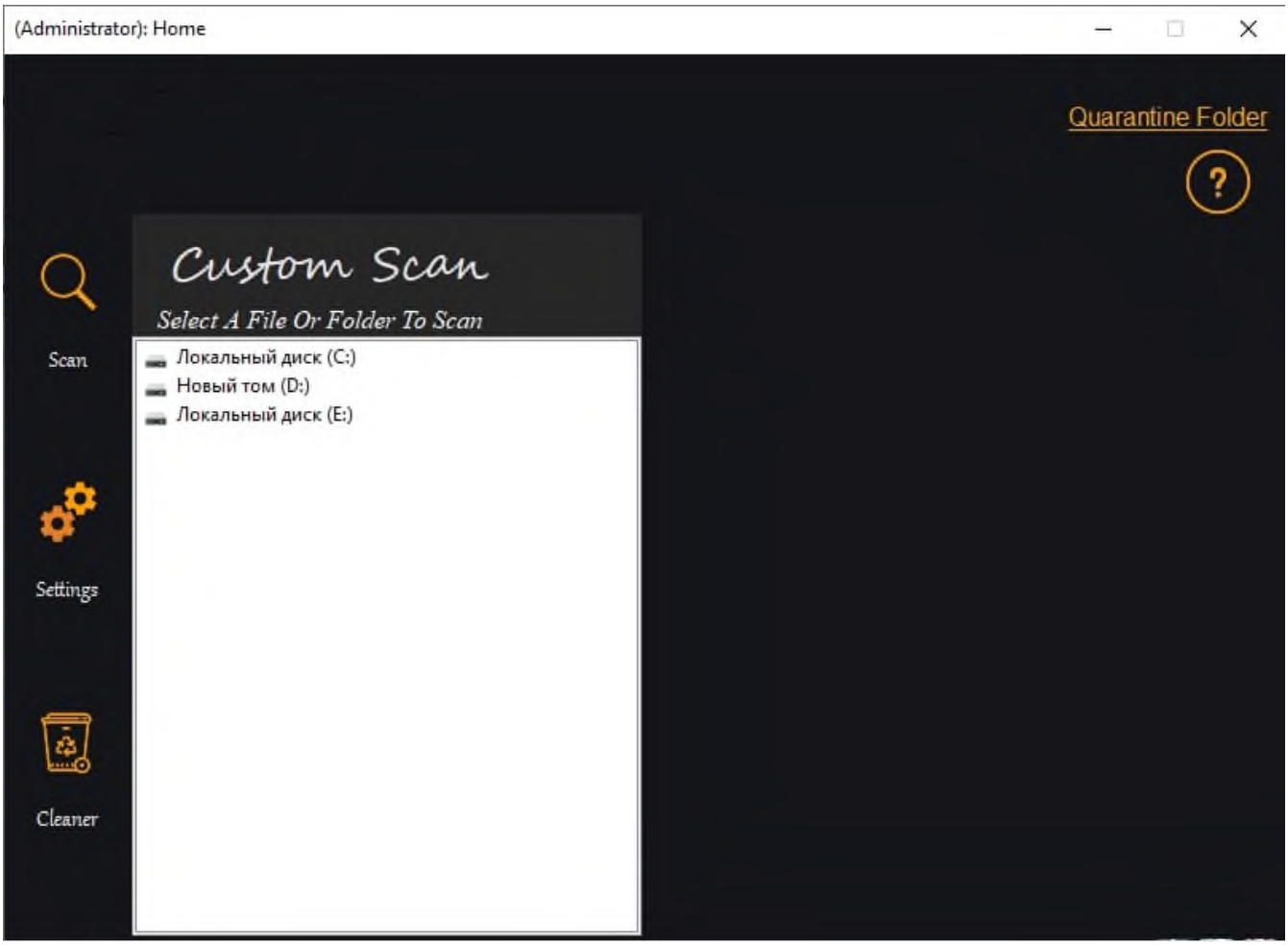

Рис. 3.2. Основне вікно розробленого програмного модуля

Даний метод досить універсальний в тому плані, що підходить абсолютно для будь-яких файлів незалежно від типу.

Однак у використання зміщення є один дуже суттєвий мінус: щоб "обдурити" сканер, досить злегка "пересунути" сукупність електронних даних у файлі, тобто змінити зміщення послідовності (наприклад, перекомпілировать вірус або додавши символ в разі скрипт-вірусу).

Для економії пам’яті і підвищення швидкості виявлення, на практиці зазвичай використовується контрольна сума (хеш) послідовності.

Таким чином перед додаванням сигнатури в базу вважається контрольна сума вибраної ділянки файлу. Це також допомагає не виявляти шкідливий код у власних базах.

Антивірусна база являє собою один файл, що містить записи про всі відомі програмному модулю віруси. Додаткові налаштування визначаються в програмному модулі (рис. 3.2).

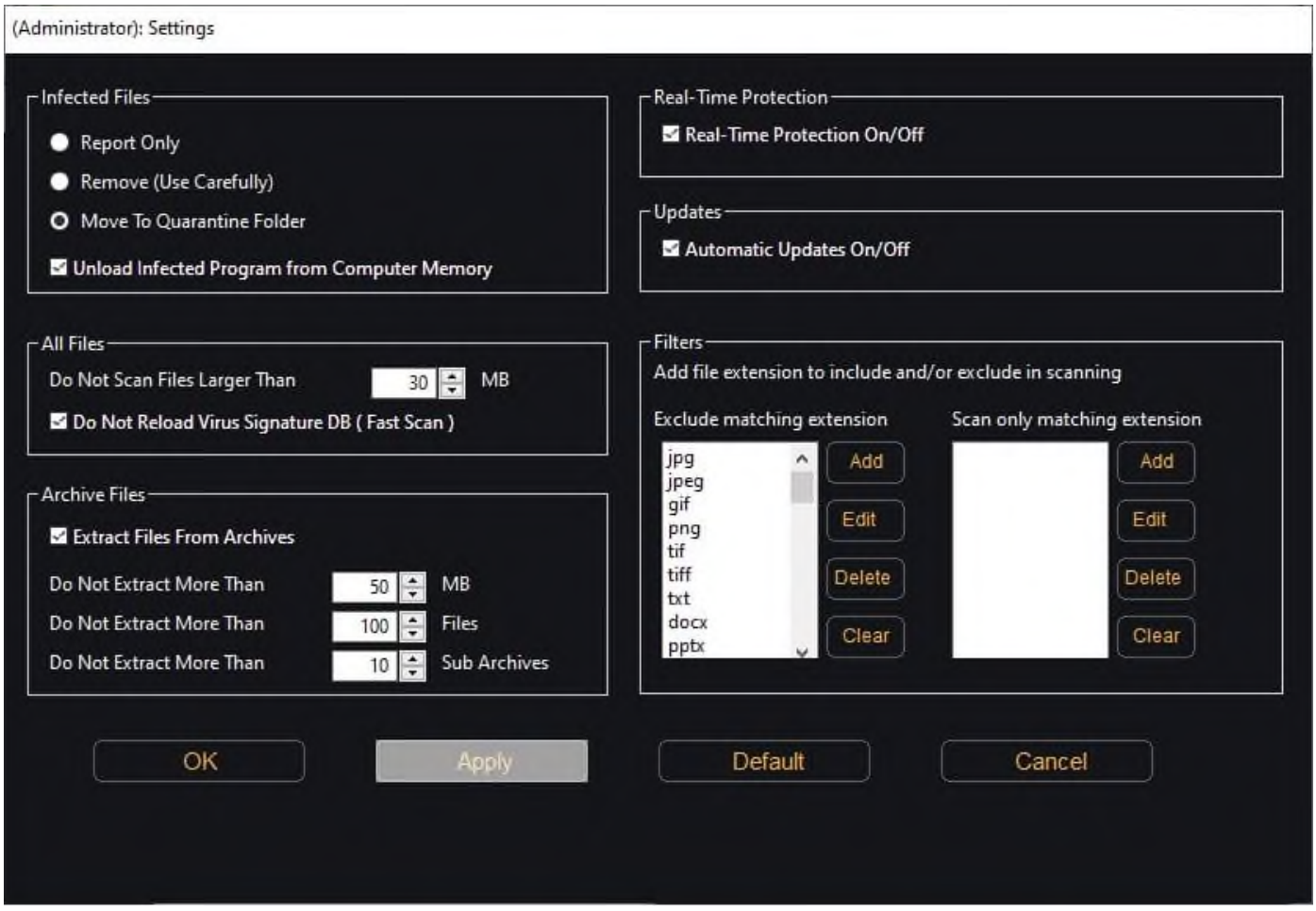

Рис. 3.3. Додаткові нааштування програмного модуля протидії шкідливим програмам

Кожен запис обов’язково містить ім’я вірусу (користувач же повинен знати, що за звір оселився у нього в системі), також в записи обов’язково присутній сигнатура, по якій виконується пошук.

Крім імені та сигнатури, запис може містити деяку додаткову інформацію, наприклад інструкції по лікуванню.

Алгоритм роботи сканера при використанні сигнатур, можна звести до кількох пунктів:

1. Завантаження бази сигнатур;

2. Відкриття і перевірка файлу;

3. Пошук сигнатури в відкритому файлі;

4. Якщо сигнатура знайдена, прийняття відповідні заходи;

5. Якщо жодної сигнатури з бази не знайдено, закриття файлу і перехід до перевірки наступного.

Отже, для виявлення шкідливих файлів нам необхідний безпосередньо сам сканер. Сканера для роботи необхідні сигнатури, які зберігаються в антивірусній базі. База створюється і наповнюється спеціальною програмою.

Однак перш ніж створювати базу, необхідно визначитися з її форматом, який в свою чергу залежить від інформації, що зберігається.

Сигнатура складається з:

– зміщення послідовності в файлі;

– Розміру послідовності;

– хеш послідовності.

Для реалізації програмного засобу було обрано мову програмування *C++* через її значні переваги над іншими [6]. Схема алгоритму виявлення шкідливих програм наведена у Додатку А.

Вимірювання – важливий аспект інформаційної безпеки, з яким рано чи пізно стикаються всі фахівці в цій галузі. Відстеження значень метрик дозволяє вчасно виявляти проблеми і вживати заходів щодо їх усунення.

Разом з тим складання каталогу метрик – індивідуальний процес для кожної організації. При правильному підході кожна метрика служить певній меті, що залежить від ситуації, і дозволяє отримати відповіді на конкретні питання.

Можна сказати, що якщо організація не знає, які метрики в області ІБ їй необхідні, вона не потребує них, так як ще не готова до проведення вимірювань.

Фахівці *R-Vision* вирішили поділитися власним списком метрик, який сформувався на основі свого досвіду, а також з практики вимірювання ІБ клієнтами. У список увійшли найбільш часто використовувані метрики, які підходять для широкого кола цілей вимірювання і які не потребують великих зусиль при зборі інформації та розрахунках.

Для аналізу програмного модуля було розділено всі метрики на наступні розділи інформаційної безпеки:

– управління інцидентами,

– управління уразливостями,

– аудит ІБ,

– управління ІБ і робота з користувачами.

Для представлених метрик організаціям слід визначити періодичність вимірювання, допустимий і критичний рівні, які дозволять інтерпретувати одержувані значення, а також цільову динаміку. Залежно від змісту, що вкладається в метрику, її цільова динаміка може бути позитивною (допустиме значення вище критичного) або негативною (допустиме значення нижче критичного).

3.1.1. Метрики у напрямку Управління інцидентами

Виділимо 3 основних метрики цієї групи:

1. Частка інцидентів за типами. Відношення числа інцидентів певного типу, виявлених за період часу, до загальної кількості виявлених за цей час інцидентів.

Цільова динаміка: негативна.

Практична цінність: оперативне виявлення проблемних областей ІБ, що вимагають уваги.

Зростання числа інцидентів певного типу може вказати на виникнення проблеми, що вимагає термінової уваги: відмова в роботі СЗІ, здійснення атаки на організацію і т.д. Відсутність інцидентів деяких типів на певних об’єктах може свідчити про некоректної налаштування *SIEM*-системи. Крім того,

результати розрахунку даної метрики можуть бути використані при обґрунтуванні керівництву організації необхідності впровадження певного СЗІ.

2. Частка інцидентів з дотриманням термінів реагування. Відношення числа інцидентів, закритих за період часу в встановлені терміни, до загальної кількості закритих за цей час інцидентів.

Цільова динаміка:позитивна.

Практична цінність:оцінка ефективності процесу реагування на інциденти, виявлення проблемних місць.

Збільшення цифри не усунених в термін інцидентів може свідчити про надто великий завантаженні групи з реагування на інциденти, порушення в комунікації між її членами, затримках на етапі розслідування інциденту.

3. Середній час реагування на інциденти (за рівнями критичності). Середній час (в годинах), за яке інцидент проходить весь встановлений в організації цикл обробки – від виявлення до закриття. Дану метрику рекомендується розраховувати окремо для кожного рівня критичності.

Цільова динаміка: негативна.

Практична цінність: оцінка ефективності процесу реагування на інциденти, визначення шляхів його оптимізації.

3.1.2. Метрики в рамках напряму управління уразливостями

1. Частка вразливостей, усунених в установлені терміни. Відношення числа вразливостей, усунених за період часу в встановлені терміни, до загальної кількості усунених за цей час вразливостей.

Цільова динаміка: позитивна.

Практична цінність: оцінка ефективності процесу усунення вразливостей, виявлення проблемних місць.

Своєчасне усунення вразливостей є важливим кроком щодо запобігання інцидентів інформаційної безпеки.

2. Середній час усунення вразливостей (за рівнями критичності). Середній час (в годинах), що пройшов з моменту виявлення уразливості до її закриття.

Цільова динаміка: негативна.

Практична цінність: оцінка ефективності процесу усунення вразливостей, виявлення шляхів його оптимізації.

Дану метрику рекомендується розраховувати окремо для кожного рівня критичності.

3. Частка усунених вразливостей (за рівнями критичності). Відношення кількості усунених вразливостей певного рівня критичності до загальної кількості виявлених вразливостей цього рівня.

Цільова динаміка: позитивна.

Практична цінність: контроль темпу усунення вразливостей, запобігання порушенню строків усунення.

Особливу увагу слід приділити високим рівням критичності. Зниження значення метрики може вказувати на високу завантаженість персоналу, неправильний розподіл навантаження, а також на загальне порушення процесу управління уразливими.

В нормальному режимі функціонування вікно програмного модуля не містить наявних ознак шкідливих програм (рис. 3.4), у разі знаходження ознак шкідливої програми буде надано попередження (рис. 3.5).

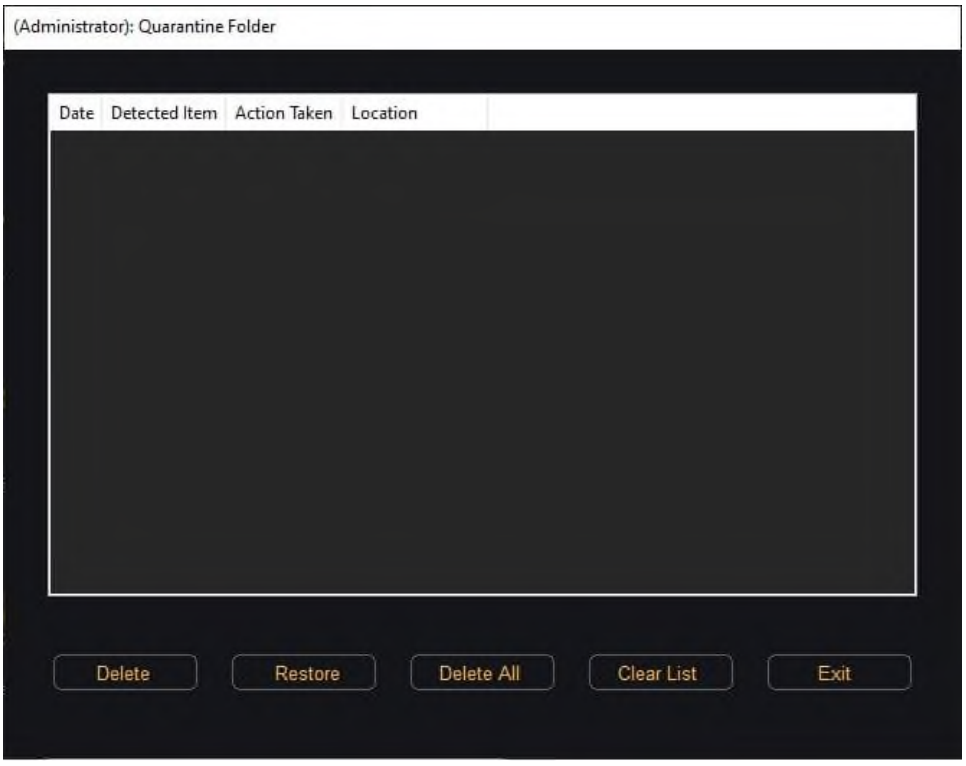

Рис. 3.4. Вікно програмного модуля в нормальному режимі

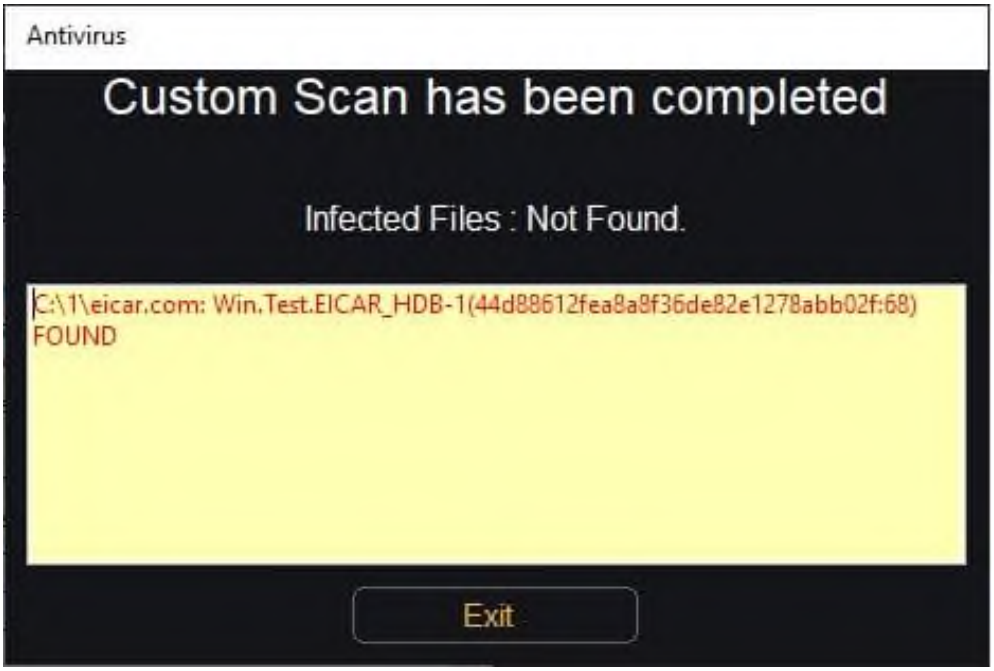

Рис. 3.5. Вікно програмного модуля з попередженням про можливу наявність шкідливої програми

3.2. Оцінювання ступеня навантаження системи

При оцінювані розроблених антивірусних програмних засобів (АВПЗ) для інформаційно-обчислювальних систем різного призначення, крім безпосередніх показників ефективності цих засобів з протидії комп'ютерним вірусам і шкідливим програмам, необхідно враховувати ступінь їх впливу на якість функціонування захищаються ними інформаційно-обчислювальних систем.

Оскільки при роботі АВПЗ забирають частину обчислювальних ресурсів (процесорний час, оперативну пам'ять і ін.), Вплив цих засобів на захищає систему виражається в зниженні продуктивності інформаційно-обчислювальної системи, що, в свою чергу, може знизити якість виконання покладених на неї завдань [1].

Виходячи з цього, ступінь впливу АВПЗ на якість функціонування інформаційно-обчислювальної системи можна оцінити через показник, що характеризує зниження продуктивності інформаційно-обчислювальної системи при виконанні кожної з покладених на неї завдань [1, 2].

В даний час використовуються два підходи до оцінки ступеня впливу АВПЗ на захищається інформаційно-обчислювальну систему.

Перший підхід [3-5] в якості показників, що характеризують вплив АВПЗ на захищає систему, використовує показники, що характеризують витрати ресурсів інформаційно-обчислювальної системи:

– використання оперативної пам'яті в стані спокою і під час сканування АВПЗ накопичувачів інформації інформаційно-обчислювальної системи;

– завантаженість процесора в стані спокою і під час сканування АВПЗ накопичувачів інформації інформаційно-обчислювальної системи;

– час читання / запису даних на накопичувачі інформації в стані спокою і під час сканування АВПЗ накопичувачів інформації інформаційно-обчислювальної системи;

– кількість дочірніх процесів АВПЗ;

– займаний АВПЗ обсяг на жорсткому диску інформаційно-обчислювальної системи після установки;

– пропускна здатність сканування.

Перераховані показники досить просто оцінити за допомогою штатних засобів оцінки продуктивності, що входять до складу практично всіх операційних систем. Однак з їх допомогою складно оцінити зниження якості виконання окремих завдань, покладених на інформаційно-обчислювальну систему. До того ж, оскільки результат оцінки представляється сукупністю кількох різнорідних показників, завдання вибору раціонального (з точки зору споживання ресурсів) АВПЗ зводиться до вирішення задачі багатокритеріального вибору, що представляє певну складність [6].

Другий підхід [7-11] передбачає використання показників падіння продуктивності інформаційно-обчислювальних систем при виконанні певних типових завдань інформаційно-обчислювальної системою з встановленим АВПЗ, наприклад:

– завантаження операційної системи;

– запуск будь-яких програм;

– завантаження будь-яких документів в програми, які їх обробляють (текстові та графічні редактори, електронні таблиці і т.п.);

– завантаження даних з мережі Інтернет та ін.

Як правило, при використанні другого підходу в якості показника, що характеризує падіння продуктивності, використовується час виконання кожної з задач до установки АВПЗ в інформаційно-обчислювальну систему і після установки АВПЗ. Для вимірювання часу виконання завдань використовуються різні програмні засоби (наприклад, для оцінки часу завантаження операційної системи – утиліта *BootRacer*, а для оцінки часу завантаження і запуску програм – утиліта *AppTimer*).

Даний підхід, на відміну від першого, дозволяє оцінити падіння продуктивності інформаційно-обчислювальної системи при виконанні окремих завдань, але також не позбавлений певних недоліків. По-перше, незважаючи на те, що в більшості випадків виміряні оцінки однорідні і являють собою значення часу виконання завдань, вони можуть мати великий розкид за кількісними показниками (від сотих часток секунд до декількох хвилин), а по-друге, як і при використанні першого підходу, завдання вибору раціонального АВПЗ зводиться до задачі багатокритеріального вибору.

Для усунення зазначених недоліків і обмежень відомих і застосовуваних способів оцінки зниження продуктивності информаци Онно-обчислювальних систем при їх захисту від комп’ютерних вірусів і шкідливих програм за допомогою АВПЗ необхідно:

– провести оцінку зниження продуктивності для кожної з покладених на інформаційно-обчислювальну систему завдань (при цьому в якості показників зниження продуктивності використовувати безрозмірні величини у вигляді коефіцієнтів зниження продуктивності);

– провести оцінку рівня значущості (коефіцієнта важливості) для кожної задачі, виконуваної інформаційно-обчислювальної системою виходячи з її призначення;

– виробити спільну оцінку ступеня впливу АВПЗ на продуктивність інформаційно-обчислювальної системи шляхом згортки отриманих коефіцієнтів

зниження продуктивності для кожної з задач з урахуванням рівня їх значимості (коефіцієнта важливості).

Відповідно, методика оцінки ступеня впливу АВПЗ на якість функціонування інформаційно-обчислювальної системи буде комплексною і включати в себе три приватні методики: методику оцінки зниження виводительности інформаційно-обчислювальної системи при виконанні кожної з задач, покладених на систему; методику оцінки рівня значущості завдань, покладених на інформаційно-обчислювальну систему; методику загальної оцінки ступеня впливу АВПЗ на якість функціонування інформаційно-обчислювальної системи.

Методика оцінки зниження продуктивності інформаційно-обчислювальної системи при виконанні кожної з задач, покладених на систему. Для оцінки зниження рівня продуктивності інформаційно-обчислювальних систем можна використовувати як тимчасові показники, так і кількісні.

Тимчасові показники характеризують збільшення часу виконання будь-якої задачі або операції інформаційно-обчислювальної системою:

$$\Delta t_{оп} = t_{ф\ оп} - t_{оп},$$

де $\Delta t_{оп}$ – приріст часу виконання завдання або операції з урахуванням впливу АВПЗ;

$t_{ф\ оп}$ – час виконання інформаційно-обчислювальної системою завдання або операції з урахуванням впливу АВПЗ;

$t_{оп}$ – час виконання інформаційно-обчислювальної системою завдання або операції без впливу АВПЗ.

Кількісні показники характеризують зменшення кількості одночасно виконуваних інформаційно-обчислювальної системою завдань або операцій за одиницю часу:

$$\Delta k_{оп} = k_{оп} - k_{ф\ оп},$$

де $\Delta k_{оп}$ – зниження числа одночасно виконуваних завдань або операцій з урахуванням впливу АВПЗ;

$k_{оп}$ – число одночасно виконуваних завдань або операцій до введення до складу інформаційно-обчислювальної системи АВПЗ;

$k_{¢\ оп}$ – число одночасно виконуваних завдань або операцій після введення до складу інформаційно-обчислювальної системи АВПЗ.

Перехід до безрозмірних коефіцієнтів зниження продуктивності (*КП*С) проводиться таким чином:

– для визначення коефіцієнта зниження продуктивності за кількісними показниками:

$$K^{k}_{\text{ПС}} = \frac{\Delta \text{k}_{\text{оп}}}{\text{k}_{\text{оп}}};$$

– для визначення коефіцієнта зниження продуктивності за часовими показниками:

$$K^{t}_{\text{ПС}} = \frac{\Delta \text{t}_{\text{оп}}}{\text{t}_{\text{оп}}}$$

Методика оцінки рівня значущості завдань, покладених на інформаційно-обчислювальну систему. Рівень визначається шляхом опитування експертів з використанням методу парних порівнянь. Даний метод дозволяє зробити почергове порівняння двох елементів, ігноруючи всі інші, що значно полегшує процес прийняття рішення [12].

Для цього список всіх завдань заноситься в таблицю парних порівнянь (табл. 3.1), а далі кожен з експертів приймає рішення по почергової оцінки значущості кожного завдання при порівнянні її з іншими шляхом розподілу свого го- Лосано між двома порівнюваними завданнями (наприклад, якщо значимість порівнюваних двох завдань, на думку експерта, однакова, в таблицю заносяться числа 0,*5* і 0,*5*, або, якщо значимість першого завдання, на думку експерта, значно перевершує значущість другого завдання, в таблицю заносяться числа 0,*9* і 0,*1*).

Кожен експерт заповнює свою таблицю, після чого формується загальна таблиця парних порівнянь всіх експертів, яка має такий же вигляд, що і таблиця парних порівнянь для одного експерта, при цьому значенням кожної комірки даної таблиці є сума значень цієї ж комірки в таблиці.

Таблиця 3.1

Приклад заповненої експертом аналітичного відділу таблиці парних порівнянь для п’яти завдань

Найменування завдання	Завдання № 1	Завдання № 2	Завдання № 3	Завдання № 4	Завдання № 5
Завдання № 1	-	0,*7*	0,*4*	0,*2*	0
Завдання № 2	0,*3*	-	0,*5*	0,*4*	0,*1*
Завдання № 3	0,*6*	0,*5*	-	0,*9*	0,*3*
Завдання № 4	0,*8*	0,*6*	0,*1*	-	0,*2*
Завдання № 5	1	0,*9*	0,*7*	0,*8*	-

Методика загальної оцінки ступеня впливу АВПЗ на якість функціонування інформаційно-обчислювальної системи. Виходячи з того, що для кожного окремого показника на попередньому етапі визначені їх коефіцієнти важливості, які визначаються значущістю завдань, покладених на інформаційно-обчислювальну систему, а також з однорідності цих показників (виражені у вигляді безрозмірних коефіцієнтів), загальну оцінку ступеня впливу АВПЗ на якість функціонування можна визначити за допомогою середньозваженого арифметичного узагальненого показника [12, 13]:

За допомогою отриманих загальних оцінок для різних АВПЗ можливий вибір такого з них, яке чинить найменший вплив на продуктивність інформаційно-обчислювальних систем і, відповідно, на якість виконання завдань, покладених на захищається інформаційно-обчислювальну систему.

Оцінка ступеня впливу розробленого АВПЗ на типову інформаційно-обчислювальну систему за допомогою запропонованих методик. Під типовий інформаційно-обчислювальної системою в даному випадку розуміється персональний комп’ютер, орієнтований на вирішення спільних завдань (перегляд і редагування документів, перегляд інтернет-сторінок, робота з архівами документів і т.п.). Конфігурація комп’ютера, який використовувався при оцінці ступеня впливу АВПЗ на якість рішення задач: процесор – *Intel Core i3*-2120 *CPU* 3.30 *GHz;* обсяг і тип оперативної пам’яті – 2,*0*0 Гб, *DDR3*; відеокарта – *Intel HD Graphics Family*; обсяг накопичувача на жорсткому магнітному диску – 250 Гб; операційна система – *Microsoft Windows* 7 32 *bit.*

В якості завдань, покладених на типову інформаційно-обчислювальну систему, було обрано такі:

- завантаження операційної системи;

- завантаження інтернет-сторінки в веб-браузер (інтернет-сторінка – *nau.edu.ua*, веб-браузер – *Opera* 12.0);

- завантаження документа в *Microsoft Excel* 2013 (документ обсягом 13 483 байта);

- завантаження документа в *Microsoft Word* 2013 (документ обсягом 11 888 байт, включаючи таблиці і малюнки);

- архівування файлів (архівувалися 3 файлу загальним обсягом 129 536 байт, тип архіву *rar*, архіватор *WinRAR* 5.0.1);

- очищення одного з локальних дисків (обсяг диска 3 Гб).

Разом з розробленим програмним модулем було оцінено сім найбільш поширених АВПЗ: *Kaspersky Internet Security* (*KIS), Dr. Web Security Space, Eset NOD32 Smart Security*.

Для кожного АВПЗ проводилася серія вимірювань по кожній задачі, що включає тридцять вимірювань.

Вимірювалися час виконання кожного завдання без встановленого АВПЗ в інформаційно-обчислювальної системи і час виконання кожного завдання з кожним з вищевказаних АВПЗ. Після проведення серії вимірювань для одного АВПЗ проводилися його повне видалення та встановлення в інформаційно-обчислювальну систему наступного АВПЗ.

Час завантаження операційної системи вимірювалося за допомогою утиліти *BootRacer* 4.9, час завантаження інтернет-сторінки в веб-браузер, час завантаження документів в *Microsoft Excel* і *Microsoft Word*, час очищення локального диска – за допомогою програми *AppTimer* 1.0, час архівування файлів – за допомогою вбудованого в програму архівування таймера.

Отримані точкові оцінки середнього часу виконання кожного завдання і розраховані коефіцієнти зниження продуктивності для кожного завдання наведені в таблиці 3.2.

Таблиця 3.2

Точкові оцінки середнього часу виконання кожного завдання і розраховані коефіцієнти зниження продуктивності для кожного завдання

Задача	Без антивіруса		Власна розробка		*KIS*		*Dr Web*		*Eset NOD32*	
	t	*K*	*t*	*K*	*t*	*K*	*t*	*K*	*t*	*K*
Завантаження ОС (задача №1)	22,065	-	26,453	0,199	36,547	0,656	30,789	0,395	36,533	0,655
Завантаження сайту до веббраузера (задача №2)	0,2312	-	0,3501	0,514	0,4022	0,739	1,0601	3,585	0,2654	0,148
Завантаження файлу в *MS Excel* (задача №3)	5,5577	-	7,4221	0,336	7,0168	0,263	7,8613	0,415	6,0468	0,088
Завантаження файлу в *MS Word* (задача №4)	6,4398	-	7,6463	0,187	7,7245	0,199	7,6281	0,184	6,7961	0,055
Архівування файлів (задача №5)	18,94	-	22,15	0,169	20,56	0,086	19,23	0,015	20,01	0,057
Очищення одного з локальних дисків (задача №6)	13,67	-	15,34	0,122	15,79	0,155	16,4	0,2	14,89	0,089

Оцінка коефіцієнта значущості завдань типовий інформаційно-обчислювальної системи проводилася п’ятьма експертами. Результати оцінювання та розрахунків рівня значущості завдань представлені в таблиці 3.3.

Таблиця 3.3

Результати оцінювання та розрахунків рівня значимості завдань

	Задача №1	Задача №2	Задача №3	Задача №4	Задача №5	Задача №6	Сумарний рівень значимості *j*-ї задачі	Нормоване значення рівня значимості *j*-ї задачі (R_j)
Задача №1	0	1,2	0,9	0,8	1,7	2,2	6,8	0,08983
Задача №2	3,8	0	2,6	2,5	2,9	3,9	15,7	0,2074
Задача №3	4,1	2,4	0	2,4	4,2	4,5	17,6	0,2325
Задача №4	4,2	2,5	2,6	0	4,2	4,5	18	0,23778
Задача №5	3,3	1,9	0,7	0,7	0	2,9	9,5	0,1255
Задача №6	2,8	1,1	0,5	0,5	3,2	0	8,1	0,107

Результати загальної оцінки рівня впливу оцінених АВПЗ представлені на рисунку 3.6 у вигляді гістограми.

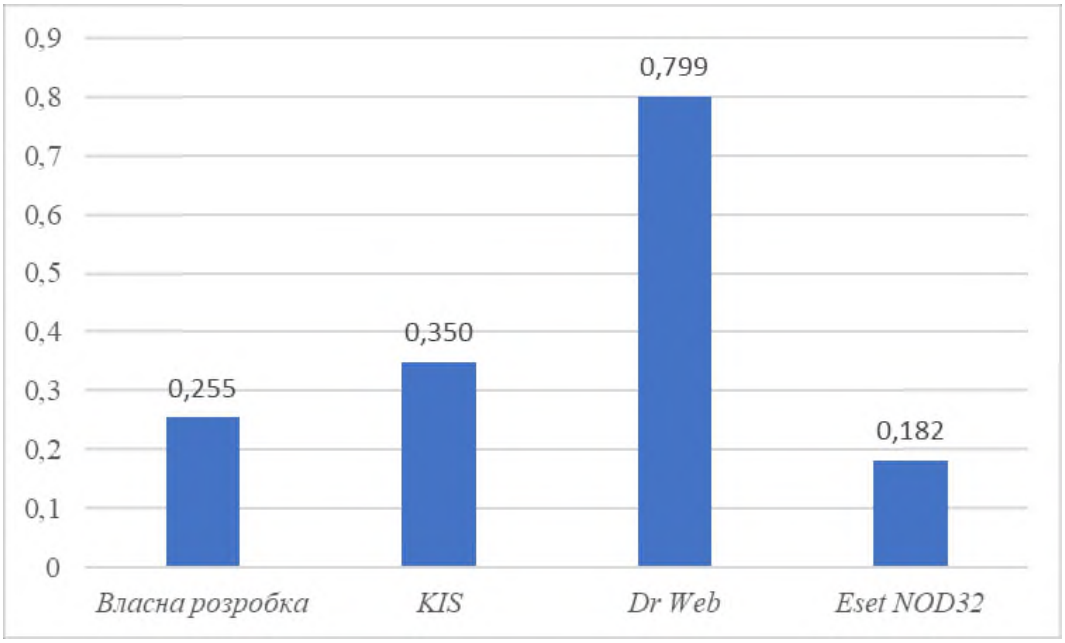

Рис. 3.6. Загальна оцінка ступеня впливу ФВПЗ на типову ІС

З отриманих результатів оцінки можна зробити висновок, що найменший вплив на інформаційно-обчислювальну систему загального застосування надає *Eset NOD32*, а на другому місці розроблений програмний модуль, але це обумовлено меншим функціоналом у порівнянні з іншими системами.

3.3. Висновки до розділу

Використовуючи для оцінки програмні метрики, необхідно враховувати, що в процесі оцінки ефективності програмного модуля оцінюється саме забезпечення інформаційної безпеки. При цьому розрахунок метрик не є самоціллю процесу. Весь процес спрямований на те, щоб надавати інформацію, на основі якої будуть прийматися рішення по видозміні алгоритмів. Вибудовуючи процес оцінки ефективності, необхідно відразу ж продумати, як будуть аналізуватися результати роботи, яким чином і ким будуть прийматися рішення щодо поліпшення безпеки, як будуть оцінюватися результати внесених змін. Без цього цінність процесу оцінки ефективності можна поставити під сумнів.

В результаті проведених досліджень було оцінено реалізовану архітектуру програмного модуля, визначено рівні ступенів протидії шкідливим програмам та

навантаження на систему. Було описано можливості використання програмного протидії шкідливим програмам для операційних систем родини *Microsoft Windows*.

При аналізі вірусної бази було визначено, що найбільш небезпечні шкідливі програми роблять не хакери-одинаки, а це добре організований кримінальний бізнес, побудований за образом і подобою компаній, що розробляють звичайні програми.

До складу таких фірм входять:

– розробники шкідливого ПЗ;

– тестувальники створеного ПЗ;

– дослідники вразливостей в операційних системах і прикладному ПЗ в злочинних цілях;

– «фахівці» з використання вірусних пакувальників та шифрування;

– розповсюджувачі шкідливого ПЗ, фахівці з соціальної інженерії;

– системні адміністратори, що забезпечують розподілене безпечну роботу всередині злочинного співтовариства і управління ботсеті;

– дроппери.

Запропонована комплексна методика дозволяє проводити оцінку ступеня впливу АВПЗ на якість функціонування інформаційно-обчислювальних систем різного призначення. Отримані оцінки дають можливість при виборі АВПЗ керуватися їх ефективністю з протидії комп'ютерним вірусам і шкідливим програмам, а також враховувати ступінь їх впливу на захищає інформаційно-обчислювальну систему. Це, в свою чергу, дозволить з усіх АВПЗ, що задовольняють вимогам по ефективності протидії комп'ютерним вірусам і шкідливим програмам, вибрати найбільш раціональний (з точки зору потребляваних ресурсів захищається інформаційно-обчислювальної системи і впливу на якість її функціонування).

ВИСНОВКИ

Число програм, що випускаються одним угрупованням, може досягати сотень зразків в день – і жоден з них не буде виявлятися антивірусним ПЗ, використовуваним цільовою групою жертв.

В даний час в будь-який момент часу жодна антивірусна програма – без застосування додаткових засобів захисту – не може забезпечити захист від проникнення ще невідомих шкідливих програм – які не надійшли на аналіз в антивірусну лабораторію.

Якщо компанія / користувач / адміністратор покладаються в справі виявлення шкідливих програм тільки на антивірусне програмне забезпечення – вони повністю беззахисні. Саме цим визначається успіх шифрувальників (а в недавньому минулому і *WinLock'*а).

Типовою є думка про те, що антивірус повинен визначати все і на вході до комп'ютера. Але це не так. Антивірус може забезпечити виявлення відомих шкідливих програм (в тому числі і за допомогою евристичних механізмів) в момент їх проникнення, а після отримання оновлень – виявити і знищити (але не повернути систему, на момент до вчинення шкідливих дій).

Можна зробити загальні висновки щодо можливостей додаткового програмного забезпечення для протидії шкідливим програмам:

1) вибір антивіруса на основі перемог в тестах – не має сенсу. Всі тести проводяться на вже відомих шкідливих програмах;

2) функція антивіруса – знищення вже активних, раніше пропущених усіма засобами захисту шкідливих програм. І ніхто крім антивіруса виконати це завдання не може. А ось функцію протидії проникненню невідомих шкідливих програм на себе повинні взяти системи контролю доступу, системи обмеження запуску, поведінкові аналізатори (плюс антивірус як аналізатор трафіку), тощо;

3) На даний момент використовувати для захисту від шкідливих програм тільки антивірус не можна. Потрібно реалізовувати систему антивірусного захисту системно в складі власне антивіруса, брандмауера, поведінкового

аналізатора, засобів офісного / батьківського контролю, системи резервування, тощо;

4) Доступ користувача повинен бути обмежений тільки реально необхідними ресурсами локальної мережі та мережі Інтернет. Повинна бути обмежена можливість використання змінних носіїв. Повинна бути заборонена установка користувачем будь-яких програм, що не дозволить вірусу, що обійшов захист засобів безпеки, встановитися на комп'ютері. Склад встановленого і використовуваного програмного забезпечення повинен відповідати відомого переліку;

5) повинна використовуватися система централізованої установки оновлень. Повинна бути виключена ситуація, коли рішення про необхідність установки оновлення приймає користувач. Використання централізованої системи оновлень дозволяє в режимі реального часу контролювати відсутність відомих вразливостей на захищаються робочих станціях і серверах;

6) робота користувача повинна відбуватися тільки під обліковим записом з обмеженими правами.

Використовуючи для оцінки програмні метрики, необхідно враховувати, що в процесі оцінки ефективності програмного модуля оцінюється саме забезпечення інформаційної безпеки. При цьому розрахунок метрик не є самоціллю процесу. Весь процес спрямований на те, щоб надавати інформацію, на основі якої будуть прийматися рішення по видозміні алгоритмів. Вибудовуючи процес оцінки ефективності, необхідно відразу ж продумати, як будуть аналізуватися результати роботи, яким чином і ким будуть прийматися рішення щодо поліпшення безпеки, як будуть оцінюватися результати внесених змін. Без цього цінність процесу оцінки ефективності можна поставити під сумнів.

В результаті проведених досліджень було оцінено реалізовану архітектуру програмного модуля, визначено рівні ступенів протидії шкідливим програмам та навантаження на систему. Було описано можливості використання програмного протидії шкідливим програмам для операційних систем родини *Microsoft Windows*.

При аналізі вірусної бази було визначено, що найбільш небезпечні шкідливі програми роблять не хакери-одинаки, а це добре організований кримінальний бізнес, побудований за образом і подобою компаній, що розробляють звичайні програми.

До складу таких фірм входять:

– розробники шкідливого ПЗ;

– тестувальники створеного ПЗ;

– дослідники вразливостей в операційних системах і прикладному ПЗ в злочинних цілях;

– «фахівці» з використання вірусних пакувальників та шифрування;

– розповсюджувачі шкідливого ПЗ, фахівці з соціальної інженерії;

– системні адміністратори, що забезпечують розподілене безпечну роботу всередині злочинного співтовариства і управління ботсеті;

– дроппери.

Запропонована комплексна методика дозволяє проводити оцінку ступеня впливу АВПЗ на якість функціонування інформаційно-обчислювальних систем різного призначення. Отримані оцінки дають можливість при виборі АВПЗ керуватися їх ефективністю з протидії комп'ютерним вірусам і шкідливим програмам, а також враховувати ступінь їх впливу на захищає інформаційно-обчислювальну систему. Це, в свою чергу, дозволить з усіх АВПЗ, що задовольняють вимогам по ефективності протидії комп'ютерним вірусам і шкідливим програмам, вибрати найбільш раціональний (з точки зору потребля-ваних ресурсів захищається інформаційно-обчислювальної системи і впливу на якість її функціонування).

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1.

Додаток А

Схема алгоритму виявлення шкідливих програм

Рис. А.1. Схема алгоритму виявлення шкідливих програм

Додаток Б

Лістинг коду основного програмного модуля