НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено
В.о. завідувача кафедри
_____ **Микола ГРАЙВОРОНСЬКИЙ**
(підпис)
“____”_____ 2021 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи, технології та математичні методи Кібербезпеки»
спеціальність: 125 «Кібербезпека»

на тему: Модифікація загальнодоступних програм з віртуальним середовищем для вдосконалення безпеки корпоративних мереж з неоднорідною інформаційною структурою

Виконав: здобувач вищої освіти **IV** курсу, групи __ФБ-72__
(шифр групи)
Курт Олег Владиславович
(Підпис)

Керівник Гальчинський Леонід Юрійович
(Підпис)

Рецензент
(Підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис) (Підпис)

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Микола ГРАЙВОРОНСЬКИЙ
«__»_____2021 р
(підпис

ЗАВДАННЯ
на дипломну роботу здобувача вищої освіти

Курт Олег Владиславович
(прізвище, ім'я, по батькові)

1. Тема роботи Модифікація загальнодоступних програм з віртуальним середовищем для вдосконалення безпеки корпоративних мереж з неоднорідною інформаційною структурою
керівник роботи Гальчинський Леонід Юрійович, к. е. н. доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «____» _____ 2021 р. №

2. Термін подання здобувачем вищої освіти роботи 07 червня 2021 р.

3. Вихідні дані до роботи:

4. Зміст роботи: аналіз технологій антивірусних програм, модифікація програмного забезпечення, практичне використання та аналіз результатів

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

Презентація до захисту дипломної роботи, яка відображає ключові моменти

6. Дата видачі завдання «12» квітня 2021 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Отримання завдання	01.12.2020	виконано
2	Вивчення рекомендованої літератури та наукових робіт	18.01.2021	виконано
3	Огляд та аналіз методів антивірусних програм	01.02.2021	виконано
4	Огляд та аналіз методів створення ізольованого середовище	17.02.2021	виконано
5	Порівняльна характеристика методів ізолювання	17.03.2021	виконано
6	Виконання програмної реалізації обраного методу	20.04.2021	виконано
7	Перевірка роботи та ефективності програми	10.05.2021	виконано
8	Підготовка ілюстративного матеріалу. Оформлення текстової і графічної частини	25.05.2021	виконано
9	Оформлення дипломного проекту	29.05.2021	виконано

Здобувач вищої освіти _____ Олег КУРТ

Керівник роботи _____ Леонід ГАЛЬЧИНСЬКИЙ

РЕФЕРАТ

Робота складається з 3 розділів, містить 10 ілюстрацій, 1 таблицю, 22 літературних посилань, обсяг роботи – 49 сторінок.

Метою дипломної роботи є проведення порівняльного аналізу антивірусних методів та технологій забезпечення ізольованої області для модифікації загальнодоступних програм з віртуальним середовищем, щоб забезпечити захист від потрапляння шкідливого програмного забезпечення через архіви або виконувані файли завантажених з недовірених джерел

Об’єктом дослідження є виявлення шкідливого навантаження на систему чи шкідливого програмного забезпечення в підозрілих файлах, методом ізолювання таких об’єктів від головної системи.

Предметом дослідження є спостерігання та аналіз поведінки завантажених файлів з недовірених джерел.

Актуальність роботи полягає в тому, що на сьогодні, на просторах інтернету з’являться велика кількість виконуваних файлів, які можуть ховати в собі різноманітне шкідливе ПО, вірус, троян та інші. Програма дозволяє перевірити такі файли не тільки на одній операційній системі, а на декількох одночасно і виявити загрозу на старих версіях ОС, так як вони мають слабкішу систему захисту і є більш вразливими.

Наукова новизна полягає в тому, що за даними порівняльного аналізу, та отриманого в роботі програмного застосунку отримати кращий рівень захисту кінцевих точок, та корпоративної мережі в цілому.

Практичне застосування полягає в тому, що за допомогою програми, можливо перевіряти різноманітні підозрілі об’єкти, на наявність шкідливого програмного забезпечення, імітуючи при цьому інфраструктуру компанії, тобто її сервери з різною операційною системою. Це створено для додаткового рівня

безпеки від потрапляння різного виду шкідливого програмного забезпечення через завантаженні файли з недовірених джерел.

Ключові слова: ізольоване середовище, віртуалізація, контейнеризація, хостова ОС, гостьова ОС.

ABSTRACT

The work consists of 3 sections, contains 10 illustrations, 1 table, 22 references, the volume of work - 49 pages.

The aim of the thesis is to conduct a comparative analysis of antivirus methods and technologies to provide an isolated area to modify public programs with a virtual environment to provide protection against malware through archives or executable files downloaded from untrusted sources

The object of the study is to detect malicious load on the system or malicious software in suspicious files, by isolating such objects from the main system.

The subject of the study is the observation and analysis of the behavior of downloaded files from untrusted sources.

The relevance of the work is that today, on the Internet there will be a large number of executable files that can hide a variety of malware, viruses, Trojans and others. The program allows you to check such files not only on one operating system, but on several at once and detect the threat on older versions of the OS, as they have a weaker protection system and are more vulnerable.

The scientific novelty is that according to comparative analysis, and obtained in the work of the software application to get the best level of protection of endpoints, and the corporate network as a whole.

The practical application is that with the help of the program, it is possible to check various suspicious objects for malware, while simulating the company's infrastructure, ie its servers with different operating systems. It is designed for an extra level of security against getting various types of malware through downloading files from untrusted sources.

Keywords: isolated environment, virtualization, containerization, host OS, guest OS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

EDR – Endpoint Detection and Response

SOAR – Security Orchestration, Automation and Response

NGFW – Next-Generation Firewall

ІТ – Інформаційні технології

ЦП – центральний процесор

ПЗ – Програмне забезпечення

ОС – Операційна система

Атака нульового дня – використання вразливості програмного забезпечення, яке ще невідоме користувачам чи розробникам програмного забезпечення.

Platform as a Service (PaaS) - платформа як послуга.
Software as a Service (SaaS) - програмне забезпечення як послуга.

ВСТУП

За останні декілька років, атаки на інформаційну систему значно підвищили свій рівень складності і фінансову сторону, а самі зловмисники мають високу компетенцію. Але це ще не межа, так як засоби і методи цифрового криміналу продовжують активно розвиватися, і вдосконалювати свої атаки.

За даними від аналітиків компанії Angara, відзначається винахідливість кіберзлочинців в приховувані слідів присутності шкідливого програмного забезпечення в корпоративній інфраструктурі, високу швидкість використання вразливостей нульового дня, ускладнювання технік фішингу та першого проникнення.

Також, за даними компанії Symantec, котра опублікувала звіт про атаки хакерського угрупування Thrip. Ця група використовувала для злому легітимні програмні забезпечення. Зокрема, для запуску віддалених процесів хакерами були використані легітимні утиліти PsExec і LogMeIn, а для крадіжки файлів - легальний FTP-клієнт WinSCP. Використання таких утиліт дозволяє хакерам уникати від всевидющого ока антивірусів, тому що легітимні програми не можуть позначатися як шкідливі, навіть якщо їх використовують хакери для своїх цілей. Контроль додатків тут теж не допомагає – ці утиліти, як правило, дозволені для використання в організаціях.[7]

Взявши до уваги навіть ці данні про кібератаки, можливо прийти до висновку, що захист інформаційних систем лише одним антивірусом, та лише сканування системи – є недостатнім, для виявлення більш складних, націлених атак. Тому, є сенс в розробці нового або модифікації вже існуючого програмного забезпечення, що дозволить покращити захист на кінцевих пристроях.

Тож ***метою*** даної дипломної роботи є проведення порівняльного аналізу антивірусних методів та технологій забезпечення ізольованої області для

модифікації загальнодоступних програм з віртуальним середовищем, щоб забезпечити захист від потрапляння шкідливого програмного забезпечення через архіви або виконувані файли завантажених з недовірених джерел.

Для досягнення даної мети було поставлено такі ***завдання***:

- Аналіз методів, що використовуються в антивірусних програмах;
- Аналіз технологій та рішень, що забезпечують створення ізольованого середовища;
- Модифікація та доповнення загальнодоступних програм для автоматизованого розповсюдження та перевірки підозрілих об’єктів на різних типах ОС.

Актуальність роботи полягає в тому, що на сьогодні, на просторах інтернету з’являться велика кількість виконуваних файлів, які можуть ховати в собі різноманітне шкідливе ПО, вірус, троян та інші. Програма дозволяє перевірити такі файли не тільки на одній операційній системі, а на декількох одночасно і виявити загрозу на старих версіях ОС, так як вони мають слабкішу систему захисту і є більш вразливими.

Об’єктом дослідження – є виявлення шкідливого навантаження на систему чи шкідливого програмного забезпечення в підозрілих файлах, методом ізолювання таких об’єктів від головної системи.

Предметом дослідження – є спостерігання та аналіз поведінки завантажених файлів з недовірених джерел.

Практичне застосування полягає в тому, що за допомогою програми, можливо перевіряти різноманітні підозрілі об’єкти на наявність шкідливого програмного забезпечення, імітуючи при цьому інфраструктуру компанії, тобто її сервери з різною операційною системою. Це створено для додаткового рівня безпеки від потрапляння різного виду шкідливого програмного забезпечення через завантаженні файли з недовірених джерел.

1 АНАЛІЗ ТЕХНОЛОГІЙ АНТИВІРУСНИХ ПРОГРАМ

1.1 Статистичні антивірусні методи

Статистичні антивірусні методи поділяються на декілька технологій, а саме: технології сигнатурного аналізу, статистичний та евристичний аналіз.

Технології сигнатурного аналізу

Сигнатурний аналіз – це метод виявлення вірусів, що полягає в пошуку відомих характерних ознак атаки чи вірусу[2].

Сигнатурний аналіз є найбільш відомим методом виявлення вірусів і використовується практично у всіх сучасних антивірусах. Для проведення перевірки, антивірусу необхідно мати набір вірусних сигнатур, що зберігаються в антивірусній базі.

Через те, що сигнатурний аналіз припускає перевірку файлів на наявність сигнатур вірусів, антивірусна база має потребу в періодичному відновленні для підтримки актуальності антивірусу. Сам принцип роботи сигнатурного аналізу також визначає границі його функціональності, а саме можливість виявляти лише відомі віруси, проте пошук нових – сигнатурний сканер неспроможний. [2]

Приклад сигнатури з тіла вірусу *Email-Worm.Win32.Happy*, опублікований в журналі Virus Bulletin:

Happy New Year 1999 !!

Сигнатури антивірусів створюються за допомогою кропіткого аналізу декількох копій файлу, що належать одному вірусу. Сигнатура повинна містити тільки унікальні рядки цього файлу, настільки характерні, щоб гарантувати мінімальну можливість помилкового спрацьовування, що є головним

пріоритетом для будь-якої компанії, що займається антивірусами.[3] Роздивимось 3 основні, найпоширеніші, типи виявлення сигнатурних підписів: хеші, байт-підписи та евристика.

- Хеш-підписи

Найбільш основним і простим типом підпису є хеш-значення. Хеш-значення створюється за допомогою хеш-функції, яка є процедурою або математичною функцією. Така функція перетворює вхідні дані будь-якого розміру в дані фіксованого розміру. Найзнаменитіші хеш-функції – це MD5 і SHA.

Кожна така функція повинна задовольняти дві властивості:

- Швидко обчислюватися;
- Мінімізувати кількість колізій.

- Байт-підписи

Підпис байтів або виявлення байтів – це підпис, заснований на послідовності байтів файлів, які присутні у файлі або потоці даних. Байтові підписи є дуже поширеною формою виявлення і використовуються з часу першого антивірусного сканера. Їх корисність обумовлена точністю, яку вони забезпечують для виявлення послідовності байтів.

- Евристика

Евристика – загальний термін для різних методів, що використовуються для виявлення шкідливих програм за їх поведінкою. Зазвичай її застосовують, коли програмне забезпечення занадто складне для хеш та байтових підписів. На сучасному рівні розвитку системи захисту від вірусів евристичні методи в основному використовуються в аналітичному компоненті захисту, а не реалізуються у вигляді визначених технологій. Іншими словами, під «евристичним виявленням» виробник

передбачає деяку систему захисту. Аналітичний компонент, який працює за принципом сліпого пошуку рішення. При цьому технічний компонент захисту може бути яким завгодно, а можливість збору інформації для подальшого аналізу може здійснювати будь-які методи - від роботи з файлами до роботи з подіями або стану операційної системи. У загальному вигляді евристичний аналізатор в антивірусі представляє собою визначений набір правил. Система використовує ці правила, щоб на основі даних, передбачених технічним компонентом, винести рішення про те, чи являється програма або подія, що аналізується - шкідливою. Метод евристичного сканування повинен поліпшити здатність сканерів застосовувати сигнатури і розпізнавати модифіковані віруси в тих випадках, коли сигнатура збігається з тілом невідомої програми не на 100%. Однак, цю технологію слід застосовувати в сучасних програмах дуже обережно, так як вона може підвищити кількість помилкових спрацьовувань.

Тобто, підводячи підсумки, ми можемо виділити головні переваги та недоліки сигнатурного аналізу, а саме:

- Дозволяють визначати конкретну атаку з високою точністю і малою часткою помилкових викликів
- Беззахисні перед поліморфними вірусами і зміненими версіями того ж вірусу
- Вимагають регулярного і вкрай оперативного оновлення
- Вимагають ретельного ручного аналізу вірусів
- Нездатні виявити будь-які нові атаки

Статичний евристичний аналіз

На відміну від сигнатурного підпису, статистична евристика не ідентифікує шкідливе програмне забезпечення, за допомогою сигнатурного сканування. Натомість, він аналізує поведінку, структуру та інші атрибути за вірусоподібними якостями. Цей метод можна використовувати для виявлення зловмисного програмного забезпечення нульового дня, а також наявного зловмисного програмного забезпечення. Статистичний евристичний аналіз визначає ймовірність зараження файлу шляхом поглибленого аналізу інструкцій, логіки програми, використовуваних даних та загальної структури двійкового файлу, який він сканує. Це допомагає визначати гібриди й нові версії раніше відомих вірусів без додаткового відновлення антивірусної бази.[4]

Більшість евристичних антивірусних процесів використовують правило або систему на основі ваги, щоб визначити кількість небезпеки, яку може представляти функціональність програми. Якщо ці правила перевищують заздалегідь визначений поріг, спрацьовує сигнал тривоги та вживаються запобіжні дії. Залежно від налаштувань антивірусу, цей сигнал може просто надіслати попередження адміністратору сервера або автоматично помістити файл у карантин.

Як було зазначено вище, евристичні антивірусні засоби використовують ряд різних методів сканування, зокрема:

- Аналіз файлів

  Під час аналізу файлів програмне забезпечення для сканування уважно перевіряє файл, щоб визначити його призначення та намір. Наприклад, якщо метою файлу є видалення певних файлів, він може бути позначений як вірус.

- Емуляція файлу

Також відома як динамічне сканування або тестування пісочниці, емуляція файлу тестує файл у контрольованому віртуальному середовищі, щоб побачити, що відбувається. Якщо файл поводиться як вірус, це, мабуть, вірус![5]

- Виявлення генетичного підпису.

Призначене для виявлення різних варіацій вірусу. Виявлення генетичного підпису використовує попередні визначення вірусів для того щоб виявити віруси в одній родині.

Аналізуючи вищезазначене, можна виділити з плюсів: статичний евристичний аналіз має можливість виявляти шкідливе програмне забезпечення навіть до того, як заразить комп'ютер. З іншого боку, неефективний евристичний сканер може призвести до декількох помилкових спрацьовувань, коли доброякісний файл помилково ідентифікується як шкідливе програмне забезпечення.

1.2 Динамічні антивірусні методи

Як і статистичні методи, динамічні антивірусні методи також мають декілька технологій, а саме: блокатори поведінки та емулятори.

Блокатори поведінки

Блокатори поведінки пильно стежать за підозрілою діяльністю. Якщо виявляється підозріла активність, блокатор зупинить програму. На відміну від статистичних методів виявлення, методи динамічного детектування повинні враховувати лише виконання інструкції. Гарною аналогією може бути безпека в аеропорту. Неважливо, що ти найкращий в світі хірург, лікар не може взяти ніж

до літака. Блокатори поведінки зазвичай не переймаються тим, що це за програма, якщо вона спробує виконати певну дію, він її зупинить.[6] Наприклад, якщо блокатор налаштований на зупинку запису до реєстру, то багато шкідливих програм не зможуть працювати, але й багато хороших програм також стануть повністю непридатними для використання.

Опираючись на поведінку та дерева обробки загроз, можливості поведінкового блокування можуть стримувати загрози навіть при їх виконанні.

Емулятори

На відмінну від блокаторів поведінки, котрі запускають шкідливе програмне забезпечення на комп’ютері, емулятори це роблять у спеціально виділених зонах (емуляційні середовища), яке є безпечним віртуальним середовищем. За допомогою саме віртуального середовища, шкідливе ПЗ не зможе нанести шкоду безпосередньо системі, а лише віртуальному простору, так як вони забезпечують повну емуляцію власного обчислювального середовища, включаючи віртуальний ЦП, віртуальну оперативну пам'ять і віртуальну ОС разом з її підсистемами. Використовуючи такий підхід, емулятори збирають інформацію про роботу шкідливого ПЗ, методом динамічної евристики. Динамічна евристика пильно стежить за операційною системою, та має значну перевагу, при виявленні вірусів, над статичною евристикою.

Недоліком є те, що процес емуляції процесора набагато повільніший, ніж процес сканування підписів, а отже, динамічна евристика повільніша, ніж статична. Крім того, деякі віруси використовують логічні прийоми і приховують свою логіку зараження від емулятора, виконуючи зараження лише за певних умов, що заважає динамічному сканеру виявляти зловмисне програмне забезпечення.

1.3 Огляд технологій для забезпечення кращого захисту IT

EDR (Endpoint Detection and Response)

EDR (Endpoint Detection and Response) – виявлення атак на кінцеві пристрої і реагування на них[7]. Ця технологія визначається як одна з трьох основних технологій SOC (Security operations center). Таке рішення являється більшим, ніж просто захист робочих станцій і серверів від важких загроз. Вона допомагає захистити робочі місця, котрі являються ключовою цілю зловмисників (на протязі вже багатьох років) і найпоширенішими точками входу в інфраструктуру організацій.

Потрібно розуміти, що важкі загрози, як правило, це ряд цілеспрямованих заходів, створених на проникнення в структуру організації, вони можуть включати атаки з використанням невідомого шкідливого коду, скомпрометованих облікових записів, без файлових методів, законних додатків та дій, які не несуть в собі нічого підозрілого[8].

Саме з такими атаками бореться EDR, доповнюючи функціональні можливості антивірусів, ця технологія дозволяє узагальнити відомості про шкідливу активність на всьому підприємстві, виявити підозрілі дії, виявити схильні атаці пристрої, а також локалізувати шкідливу активність.

Працює EDR за допомогою спеціального агента, котрий встановлюється на кінцеві пристрої і вимагає, щонайменше 4 типи можливостей, та повинен:

- Вміти виявити інциденти з безпекою у кінцевій точці;
- Утримувати інцидент у кінцевій точці;
- Підтримувати розслідування інциденту;
- Забезпечувати механізм для усунення вражених кінцевих точок.

Такий агент, збирає інформацію про активність користувача і програм, аналізує їх, допомагає провести розслідування і покращити захист.

Хороша система EDR використовує безліч методів виявлення для знаходження загроз в кінцевих точках, для визначення того, коли ці загрози стали постійними, а також для встановлення, які зміни чи наслідки відбулися в результаті загрози.

Сюди входять такі методи:

- Виявлення IOC

  Метод визначає зміни стану системи та порівнює їх із внутрішнім IOC (Indicator of Compromise). Іноді може знадобитися надіслати зміни стану або підозрюваний файл до служби розвідки загроз для аналізу та оцінки.

- Виявлення аномалій.

  Зміни в системі з відомих хороших базових конфігурацій також можуть допомогти виявити загрози.

- Виявлення поведінки.

  Виявлення поганої, дивної або незаконної поведінки в системі може свідчити про загрозу. Запис таких подій допомагає визначити загрозу і може визначити час, коли інцидент стався або почався.

- Порушення політики

  Зміни системи (наприклад, планове технічне обслуговування або оновлення, нове встановлення програмного забезпечення, нові користувачі або зміни облікового запису).

Дуже важливо виявити успішні атаки якомога швидше після їх виникнення. Хороша система EDR ловить їх, коли вони починають розгортатися, ідентифікує їх автоматично та допомагає негайно реагувати на дії. Чим коротший період активності атак, тим менше шкоди вони можуть заподіяти.

SOAR (Security Orchestration, Automation and Response)

SOAR (Security Orchestration, Automation and Response) - це комплекс сполучених програмних засобів, що дозволяють організувати збір даних про

загрози безпеці відразу з декількох джерел, при цьому реагуючи на низькорівневі події безпеки без допомоги людини[9]. Інструменти SOAR дозволяють організації визначати процедури аналізу інцидентів та реагування на них у цифровому форматі робочого процесу.

Orchestration (оркестрація) - інтегрує різні технології та з'єднує інструменти безпеки для покращення можливостей реагування на аварії.

Automation (автоматизація) - забезпечує автоматизовані інструменти розслідування та реагування, щоб зменшити час, який необхідний командам безпеки для виявлення та вирішення інцидентів безпеки та зменшення навантаження на них.

Response (відповідь) - команди безпеки можуть використовувати «Playbooks» для запуску автоматизованих робочих процесів і виконання багатьох дій, таких як запуск розслідувань, утримання та пом'якшення загроз

Очевидними перевагами автоматизації управління інцидентами або їх частин – є рушійними чинниками зростаючої популярності платформ SOAR. Компонент оркестровки платформи SOAR сприяє інтеграції широкого спектру інструментів та технологій для управління інцидентами. Організація може приймати обґрунтовані рішення, автоматизувати дії реагування на основі позиції ризику, застосовувати захисні заходи та формалізувати процес реагування на інциденти. Як тільки платформа SOAR виявляє наявність шкідливого програмного забезпечення, вона реалізує достатні заходи для запобігання розповсюдженню інфекції та надсилає попередження команді безпеки.

NGFW (Next-Generation Firewall)

Міжмережеві екрани нового покоління (Next-Generation Firewall, NGFW) - являють собою інтегровані платформи мережевої безпеки, в яких традиційні

брандмауери поєднуються з іншими мережевими рішеннями для фільтрації трафіку, такими як системи глибокого аналізу трафіку Deep Packet Inspection (DPI), система запобігання вторгнень (IPS) і ін.

Сам термін «брандмауер наступного покоління» (Next-Generation Firewall - NGFW) був запропонований аналітиками Gartner Research і має на увазі використання в пристроях технологій мережевого брандмауера третього покоління[10].

На додаток до контролю доступу, NGFW можуть блокувати сучасні загрози, такі як просунуте шкідливе програмне забезпечення та атаки на рівні додатків. Згідно з визначенням Gartner, брандмауер наступного покоління повинен включати[11]:

- Стандартні можливості брандмауера, такі як перевірка стану
- Комплексне запобігання вторгненню
- Поінформованість про програми та контроль, щоб бачити та блокувати ризиковані програми
- Джерела розвідки загроз
- Шляхи оновлення, щоб включити майбутні інформаційні канали
- Методи вирішення нових загроз безпеці

Основна перевага NGFW - це можливість безпечно використовувати Інтернет-додатки, які дозволяють користувачам працювати більш продуктивно, блокуючи менш бажані програми. Міжмережеві екрани наступного покоління досягають цього за рахунок використання глибокої перевірки пакетів для ідентифікації та управління додатками незалежно від IP-порту, використовуваного додатком.

Висновки до розділу 1

У цьому розділі були розглянуті методи, які використовують антивірусні програми. Визначені їх переваги та недоліки, а також був здійснений огляд рішень, за допомогою яких можливо покращити безпеку корпоративних мереж від важких загроз. Тобто від цілеспрямованих заходів, створених на проникнення в структуру організації.

2 МОДИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2. 1 Методи реалізації ізольованого середовища

Одним із методів захисту від шкідливого ПЗ та уникнення атаки нульового дня є використання ізольованого середовища. Воно допомагає повністю забезпечити захист головної системи від пошкодження чи втрати даних, або будь-якого виконання зловмисного плану.

Така реалізація може бути різноманітною. Найпростіший спосіб – це виділення частини пам’яті, яка не має зв’язку з іншими програмами, її можливо контролювати і в будь-який момент видалити все, що там є. На рисунку 2.1 показана ілюстрація роботи ізольованої області в пам’яті.

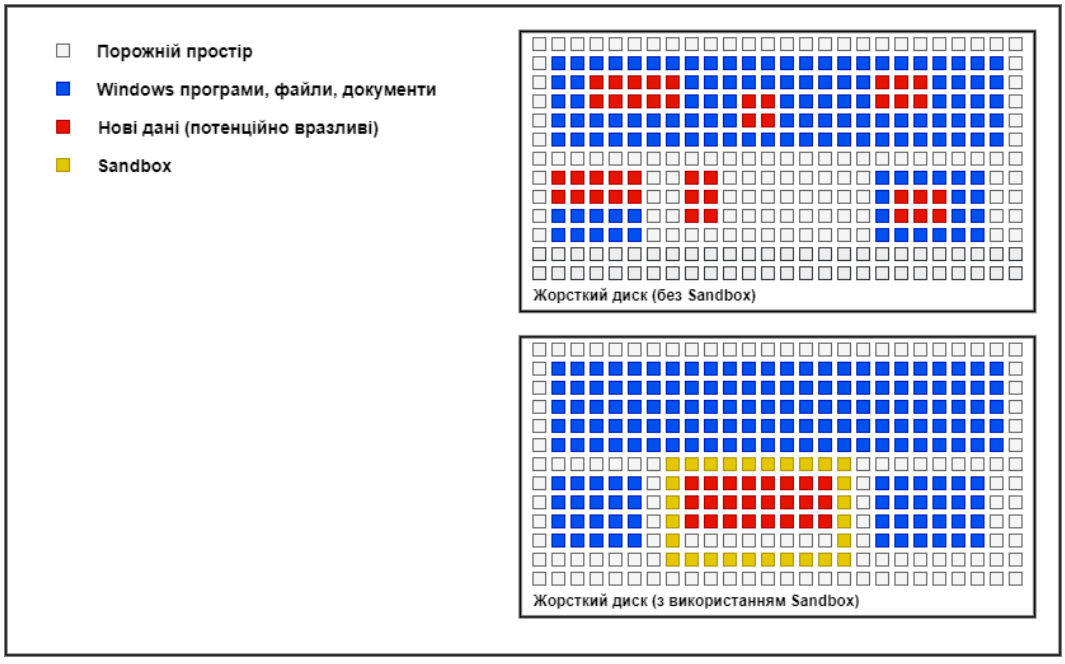

Рисунок 2.1 – Ізоляція області пам’яті на прикладі Sandbox рішення

Насамперед, є кілька варіантів отримати ізольоване середовище, деякі з них наводяться нижче.

Повна емуляція

Емуляція – часткове використання ПЗ для забезпечення іншого середовища виконання або архітектури. Проекти, що виконуються за допомогою повної емуляції працюють як інтерпретатори. Вони послідовно вибирають код гостьової операційної системи і емулюють поведінку кожної окремої взятої поведінки. За допомогою такого рішення, емуляції ЦП і пам'яті, забезпечують глибоку видимість поведінки і впливу програми.

Існує можливість запускати різні операційні системи перебуваючи лише в одній. Гарним прикладом є емулятор ОС Linux який працює в середовищі (в коробці) Windows, чи іншої операційної системи. Недоліком такого підходу є те, що можлива втрата продуктивності гостьової ОС. Гостьові програми можуть втратити свою продуктивність в 10 чи навіть в 100 разів, що в свою чергу несе практичну неможливість нормальної роботи з гостьовою ОС в середині емулятора. Повні емулятори найчастіше використовуються в якості низькорівневих відладчиків для дослідження і трасування операційних систем. [12]

Часткова емуляція

У випадку з частковою емуляцією, віртуальні машини відображають лише необхідну кількість апаратного забезпечення. Це використовується для того, щоб була можливість працювати ізольовано. Такий підхід дозволяє запускати гостьові операційні системи тільки для тієї ж архітектури, що і у хоста. [13] Він має можливість запускати екземпляри системи відвідувачів одночасно і може значно підвищити продуктивність гостьової системи при повному навантаженні. Крім того, для підвищення продуктивності, віртуальні платформи використовують спеціальний «прошарок» між гостьовою операційною системою і устаткуванням (тобто гіпервізор), який також має

назву «Монітор віртуальних машин» він, в свою чергу, дає відвідувачам прямий доступ до ресурсів устаткування[14].

Додатки, зазвичай, працюють в ізольованій адресній області і взаємодіють з інтерфейсом прикладного програмування (API), що надаються операційною системою. Якщо обидві ОС сумісні по інтерфейсу свого застосування (наприклад, Windows 2000 і Windows 98), то програми, розроблені для однієї з таких, зможуть працювати й на інших. Але з іншого боку, якщо дві операційні системи не відповідають інтерфейсу додатку один (наприклад, Windows 2008 і Linux), то тоді, для того щоб вони змогли працювати на інших системах, необхідно одночасно захопити виклики застосунків в інтерфейсі гостьовий операційної системи і змоделювати їх роботу і тільки таким чином ми можемо встановити операційну систему і одночасно працювати з різними додатками ОС.

Таким чином, до недоліків можливо віднести залежність віртуальних машини від архітектури апаратної платформи.

Віртуалізація

На відмінну від емуляції, віртуалізація безпосередньо взаємодіє з апаратним обладнанням.

У віртуалізації є кілька вимірів, які умовно можна назвати «Тип» і «Спосіб». Показані на рисунку 2.2.

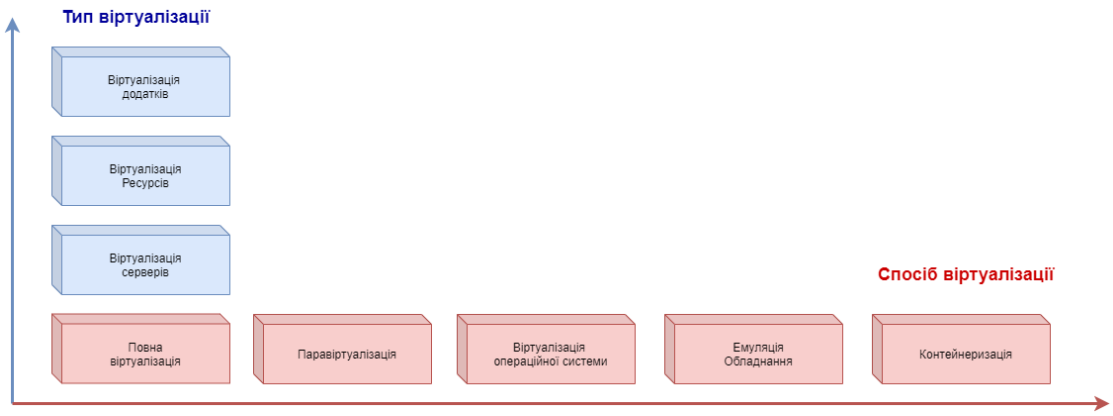

Рисунок 2.2 – Типи та способи віртуалізації

Тип віртуалізації:

- Віртуалізація ресурсів

Віртуалізація оперативної пам’яті, жорсткого диску, процесорів. Ці ресурси можуть бути нарізані для використання користувачами.

- Віртуалізація серверів

Розміщення декількох віртуальних серверів в рамках одного фізичного. Також є можливість об’єднання декількох фізичних серверів у один логічний, для вирішення складних задач.

- Віртуалізація додатків

Іншими словами PaaS і SaaS.

Спосіб віртуалізації:

- Повна віртуалізація

Процес, який повністю емулює роботу комп’ютера, його процесорів, пам’яті, відеокарт, жорстких дисків та інше.

- Паравіртуалізація

Схоже на повну віртуалізацію, але деякі компоненти, наприклад мережеві або дискові (і інші) пристрої можуть бути доступні безпосередньо через виклики назовні віртуальної машини.

- Віртуалізація операційної системи

Особливість такого способу в тому, що гостьова ОС може бути лише однією. Гарним прикладом такої віртуалізації є Linux-VServer.

- Емуляція обладнання

Особливість такого способу в тому, що відбувається повна емуляція роботи певного обладнання.

Контейнеризація

Контейнеризація - це підхід до розробки програмного забезпечення, при якому додаток або служба, їх залежності і конфігурація (абстрактні файли маніфесту розгортання) упаковуються разом в образ контейнера. Контейнерний додаток може тестуватися як єдине ціле і розгортатися як екземпляр способу контейнера в операційній системі (ОС) вузла[15].

За допомогою такого рішення, можливо переносити будь-які стандартні модулі для розгортання програмного забезпечення, які, в свою чергу, можуть мати різноманітний програмний код та залежності.

Однією з переваг контейнерів – це ізолювання не лише себе, а ще й додатків між собою, які знаходяться в загальній операційній системі, якщо їх явно не зв’язати. Таким чином, забезпечується додатковий рівень захисту.

Ще однією перевагою контейнеризації є масштабованість. Можливо швидко здійснювати горизонтальне масштабування, створюючи контейнери для короткострокових завдань.

Іншими словами, контейнери надають такі преваги, як ізоляція, переносимість, гнучкість, масштабованість і контроль, протягом усього життєвого циклу програми. Найважливішою ж перевагою - є ізоляція середовища від хостової системи[16].

Говорячи про безпеку контейнерів, можливо виділити той факт, що є деякі ризики запустити їх з неповною ізоляцією, тоді з’являється вірогідність, шо якийсь зловмисний код з мережі отримає доступ до пам’яті сервера. Щоб такого уникнути та покращити безпеку контейнерів потрібно:

- Ізолювати процеси
- Не завантажувати вже готові контейнери з недовірених джерел
- Використовувати внутрішні функції сервісу контейнеризації з пошуку вразливостей

В додаток до цього, існує так званий «життєвий» цикл, котрий був продемонстрований на конференції 11 березня 2021 року компанією «ANTI-MALWARE», рисунок 2.3 показує забезпечення захисту для контейнерів.

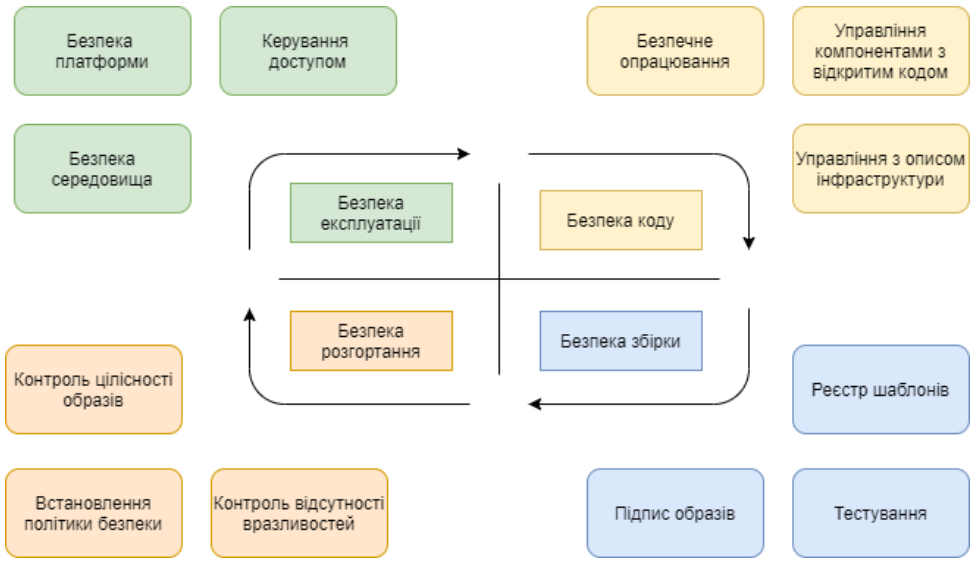

Рисунок 2.3 – Керування безпекою контейнерів

2.2 Порівняльний аналіз методів ізолювання

Як було розглянуто у попередньому підрозділі, варіантів забезпечити ізольоване середовище для захисту хостової машини та її даних – велика кількість. Але який метод все ж таки краще?

Віртуалізація системи чи емуляція?

У процесів емуляції і віртуалізації дуже багато спільного, але також присутні і помітні експлуатаційні відмінності.

Якщо клієнту необхідно працювати з більш старої операційною системою всередині сучасної технічної архітектури, він вибере виключно варіант з емуляцією.

У той же час, всі віртуалізовані системи функціонують в незалежності від використовуваного базового оснащення.

Суть емуляції полягає в тому, що одна система технічно може імітувати іншу.

Приклад - якщо структура ПО працює в системі А, але не в системі В, ми створюємо всередині системи В емуляцію роботи системи А. Внаслідок цього, ПО спокійно працює на емуляцію системи А.

Даний приклад можна перенести і на віртуалізацію, яка, крім системи А. розділена ще на 2 виділених сервера (В і С).

Обидва сервера є незалежними технічними контейнерами, що володіють персоналізованим доступом до програмних ресурсів - ОЗУ, ЦП і сховищ пам'яті - їх можна вільно перезавантажити незалежно один від одного. Їх «поведінка» цілком ідентична поведінки справжнього програмного обладнання.

Кожна з таких технологій має свої переваги і недоліки. Розглянемо деякі з них.

Емуляція

Стосовно до вищеописаного прикладу, емуляція виступає особливим «заповнювачем» для апаратного забезпечення - створення особливого технічного середовища, яке функціонує в апаратному порядку.

Емуляція може мати особливий ефект при таких призначених для користувача сценаріях:

- Старт операційної системи, спочатку призначеної для іншого системного обладнання (запуск консольної гри на ПК, робота з Windows на Mac OS);

- Старт застарілої версії ПЗ після того як порівнянне з нею обладнання застаріє.

Також процеси емуляції корисні при створенні деякого ПО для декількох систем одночасно. Процес написання програмного коду може виконуватися на одній машині, а процес емуляції на декількох операційних системах (все функціонує одночасно і без явних збоїв).

Віртуалізація

Найголовніша перевага віртуалізації полягає в тому, що вона безпосередньо взаємодіє з апаратним обладнанням.

До найбільш явних переваг такого способу взаємодії з програмними компонентами можна віднести:

- Відмінна системна сумісність яка використовується зараз архітектурою процесора x86;

- Функції демонстрації роботи фізичного пристрою як для виділеної частини апаратного і програмного забезпечення;

- Автономність на будь-якій стадії використання.

До слова, між емуляцією і віртуалізацією спокійно можна поставити знак рівності, так як ці процеси допомагають досконально проаналізувати роботу будь-якого програмного забезпечення.

Розглянемо декілька віртуальних машин та операційних емуляторів для повного розуміння чим вони відрізняються або чим схожі. (*Головним джерелом для перерахування переваг та недоліків наступних програм є – [17])*

- VMWARE

VMWare Workstation – дуже популярна і зручна у використанні віртуальна машина, що використовується на професійній основі.

Переваги продукту:

- Є некомерційна версія під назвою VMWare Workstation Player, яку можна використовувати для ознайомлювальних цілей;
- Простий і інтуїтивно зрозумілий графічний інтерфейс;
- Установка нової ОС істотно спрощена, в порівнянні з установкою традиційної версії програмного забезпечення на ПК;
- Програма дозволяє робити скріншоти системи, за допомогою яких можна відновлювати попередній стан;
- Відмінна технічна надійність і стабільність роботи;
- Швидка робота і хороша продуктивність;
- Стабільна підтримка 3D графіки.

Недоліки:

- VMWare Workstation Player - платний продукт для комерційних цілей;
- VMWare Workstation Pro - можна використовувати тільки після реєстрації;
- Окремі компоненти програми функціонують з різними операційними системами.

- VIRTUAL BOX

Поширена віртуальна машина з пристойним набором корисного технічного функціоналу.

Переваги:

- Virtual Box – дозволяє взаємодіяти з великим переліком операційних систем, як для цілей безпосередньої установки Virtual Box, так і для установки «гостьових» продуктів;
- Можна зробити скріншоти операційної системи, що дозволяють відновити попередній стан системи;
- В Інтернеті поширюється на безкоштовній основі разом з відкритим програмним кодом, а також в додатку до ліцензії GPLv2;

Недоліки:

- Препарат не можна вважати максимально продуктивним в порівнянні з іншими, платними аналогами;
- Постійно зустрічаються помилки, різні баги, креші та летальні зависання;
- Мінімальна технічна підтримка 3D графіки;
- Дуже складний графічний інтерфейс, в порівнянні з платними програмами і компонентами.

- HYPER-V

Даний продукт спочатку позиціонувався як пряма заміна компонентів Microsoft Visual PC.

Переваги:

- Доставляється разом з великою кількістю варіацій систем Windows 10;
- Підтримує процес установки гостьових операційних систем, а також всі старі версії операційної системи Windows;
- Є функція установки гостьових операційних систем Linux і FreeBSD.

Недоліки:

- Немає можливості запустити з більш ранніх і «старих» версій операційної системи Windows;
- Не вийде встановити продукт під Mac OS;
- Не дуже зручний і інтуїтивно зрозумілий графічний інтерфейс, якщо порівнювати цю програму з компонентами Virtual Box і VMWare.

- NOX

Спеціалізований емулятор під операційну систему Android.

Переваги продукту:

- Відносно «безкоштовна» програма;
- Продукт Nox дуже «легкий» і технічно «швидкий»;
- Спеціалізований маппинг клавіш під жести Android;
- Деталізовано конфігурується.

Недоліки:

- Необхідне встановлення інших додатків і компонентів;
- Можна використовувати виключно для роботи з Android системою.

- BLUESTACKS

Сучасний емулятор операційної системи Android.

Переваги:

- Відносно «безкоштовний» продукт, в якому є ознайомча версія на пробний період;
- Велика розмаїтість параметрів для тонкої настройки;
- Легке встановлення додатків;
- Функції для взаємодії на вкладках з можливістю швидкого перемикання.

Недоліки:

- Заточений під роботу тільки з гостьовим Андроїд;
- Багато реклами і сторонніх додатків;
- «Їсть» багато оперативної пам'яті, а значить, запросто може змусити гальмувати відносно не потужний або слабкий комп'ютер.

Однозначно, у віртуальних машин і емуляторів є свої сильні і слабкі сторони, що все ж таки не дозволяє поставити знак рівності між ними і традиційними способами тестування функціональності і працездатності програмного забезпечення.

Віртуальна машина чи контейнеризація?

Гостьова операційна система, така як Linux або Windows, працює поверх операційної системи хоста з віртуалізованим доступом до базового обладнання. Як і віртуальні машини, контейнери дозволяють упаковувати застосунок разом з бібліотеками та іншими залежностями, забезпечуючи ізольовані середовища для запуску програмних сервісів[18]. Однак, як можна побачити на рисунку 2.4, на цьому схожість закінчується, оскільки контейнери пропонують набагато легшу одиницю для роботи розробників та IT-відділів, несучи безліч переваг.

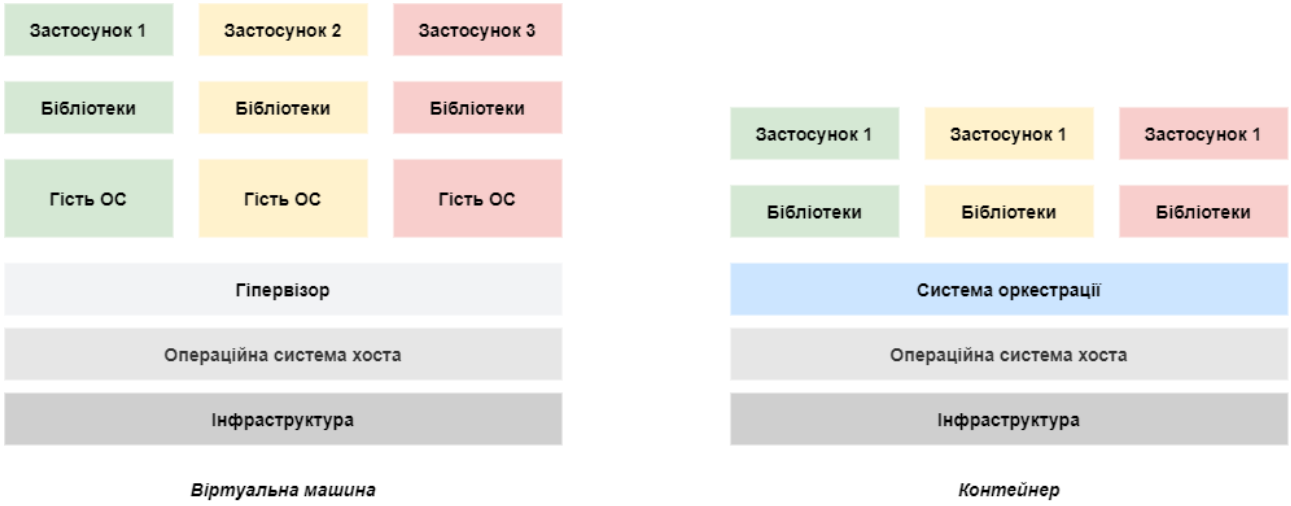

Рисунок 2.4 – Різниця між віртуальними машинами та контейнерами

Контейнери віртуалізують ЦП, пам'ять, сховище і мережеві ресурси на рівні ОС, надаючи розробникам ізольоване уявлення ОС, яке логічно ізольоване від інших додатків. Також деякі преваги контейнера від віртуальної машини наведені в таблиці 2.1.

Таблиця 2.1 – Порівняння переваг контейнера від віртуальної машини

	Переваги контейнера	Переваги віртуальної машини
Узгоджене середовище виконання	✓	✓
Можливість використовувати мікросервісні архітектури	✓	✓
Маленький розмір диску	✓	
Низькі накладні витрати	✓	

- Узгоджене середовище виконання

Налагоджений на одному комп'ютері додаток можна розгорнути на іншому.

- Можливість використовувати мікросервісні архітектури

Можливо перевірити працездатність кожного контейнера/машини, обмежити кожну службу певними ресурсами, запускати і зупиняти їх незалежно один від одного.

- Маленький розмір диску

Завдяки тому, що кожен контейнер не містить образ ОС.

- Низькі накладні витрати

Контейнери спільно використовують системне ядро операційної системи сервера, отже, запуск контейнера не вимагає запуску окремого примірника ОС для кожної програми. Це підвищує ефективність сервера і знижує витрати на сервер і ліцензування.

Підтримка операційної системи віртуальної машини і контейнерів (наприклад Docker) сильно відрізняється. На рисунку 2.3 можна бачити, що кожна віртуальна машина має свою гостьову операційну систему над основною, що робить їх важкими. З іншого боку, контейнери використовують загальну операційну систему хоста, і тому вони легковажні, а спільне використання ОС хоста між контейнерами, робить їх дуже легкими і допомагає їм завантажуватися всього за кілька секунд.

Тобто, якщо необхідно запускати максимум додатків на мінімальній кількості серверів, тоді краще скористатися контейнерами. Але все ж необхідно додатково подбати про безпеку. Якщо ж потрібно виконувати безліч і / або підтримувати різні ОС - краще брати віртуальні машини. І якщо питання безпеки для організації стоїть на першому місці, то теж краще залишитися при VM. У віртуальних машин є чимало переваг але у міру «дорослішання» технології контейнеризації на нас чекає таке майбутнє, де контейнери і

віртуальні машини спільно становитимуть гнучке і портативне хмарне середовище. [19]

Висновки до розділу 2

У цьому розділі були розглянуті методи створення ізольованого середовища і проведений між ними порівняльний аналіз. І тому, базуючись на вищесказаному та беручи до уваги всі аспекти переваг і недоліків, я прийшов до висновку, що реалізація ізольованого середовища засобами Docker, а саме docker compose, є найбільш раціональним для вирішення поставлених задач.

3 ПРАКТИЧНЕ ВИКРИСТАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Розробка основної логіки програми

За основу програми, яка відповідає за частину віртуалізації, було взято програмне забезпечення «Docker», а саме docker compose. Такий підхід дозволяє ізолювати середовище тестування шкідливого програмного забезпечення і при цьому використовувати невелику кількість ресурсів комп’ютера, що, в свою чергу, дає змогу додавати ще нові контейнери з іншою операційною системою.

Налаштування і програмування подій відбувається у файлі docker-compose.yml, а більш детальне налаштування кожного з контейнерів вже програмується в спеціальному файлі «Dockerfile». На рисунку 3.1 наведений приклад файлу docker-compose.yml який містить налаштування об’єднання різних контейнерів в одну систему. На рисунку 3.2 наведений програмний код для створення одного з docker-контейнерів з ОС Ubuntu:18.04.

```yaml
version: '3.9'
services:
  LinuxServer:
    container_name: Server
    build:
      context: Docker/LinuxServer
    image: server:web
    ports:
      - "9001:9000"
      - "8000"
    volumes:
      - "D:\\Учёба\\Диплом\\Практика\\Project\\box:/usr/src/app/box"
    networks:
      - net

  DebianClient:
    container_name: DebianClient
    build:
      context: Docker/DebianClient
    image: debian:unstable
    ports:
      - "9002:9000"
      - "8000"
    links:
      - LinuxServer
    networks:
      - net

  UbuntuClient:
    container_name: UbuntuClient
    build:
      context: Docker/UbuntuClient
    image: ubuntu:18.04
    ports:
      - "9003:9000"
      - "8000"
    links:
      - LinuxServer
    networks:
      - net

networks:
  net:
    driver: bridge
```

Рисунок 3.1 – Програмне об’єднання різних docker-контейнерів до однієї системи

```
FROM ubuntu:18.04
#FROM python:3.7.1

RUN mkdir -p /usr/src/app/
WORKDIR /usr/src/app/
COPY ubuntu_main.py /home/ubuntu_main.py
COPY ../.. /usr/src/app/


RUN pip install --upgrade pip && \
    pip install --no-cache-dir -r requirements_ubuntu.txt


ENV TZ Europe/Kiev

CMD ["python", "ubuntu_main.py"]
```

Рисунок 3.2 – Створення docker-контейнеру з ОС Ubuntu:18.04

Головною ідеєю є автоматична перевірка програмного файлу, тобто інсталяторів, на наявність зловмисного коду, що може призвести до непередбачуваних подій.

Користувач, для того щоб перевірити завантажений інсталятор, копіює його у спеціальну директорію, після цього програма самостійно починає запускати та аналізувати все, що знаходиться в середині. За допомогою рекурсивного перебору, алгоритм спроможний знаходити виконуваний файл навіть, якщо був доданий до директорії якийсь архів. На рисунку 3.3 наведений алгоритм автоматичної перевірки папки на наявність інсталяторів (.exe), наявність архівів, розпакування їх та запуск у віртуальному середовищі.

```python
class CheckInstaller:
    def __init__(self):
        self.existence = False

    def check_archive(self, checkpath):
        for root, dirs, files in os.walk(checkpath):
            for file in files:
                if file.endswith(".zip"):
                    os.system("unzip -d /usr/src/app/box/archives/zip {}".format(os.path.join(root, file)))
                    os.system("rm {}".format(os.path.join(root, file)))
                    self.existence = True
                if file.endswith(".tar"):
                    os.system("tar -xvf {} -C /usr/src/app/box/archives/tar".format(os.path.join(root, file)))
                    os.system("rm {}".format(os.path.join(root, file)))
                    self.existence = True
                elif file.endswith(".rar"):
                    os.system("unrar e {} /usr/src/app/box/archives/rar".format(os.path.join(root, file)))
                    os.system("rm {}".format(os.path.join(root, file)))
                    self.existence = True

        if self.existence:
            self.existence = False

            self.run_installer("/usr/src/app/box/archives")
        else:
            return "No installer"

    def run_installer(self, checkpath):
        for root, dirs, files in os.walk(checkpath):
            for file in files:
                if file.endswith(".exe"):
                    os.system("{}".format(os.path.join(root, file)))

        self.check_archive(checkpath)
```

Рисунок 3.3 – Алгоритм автоматичного пошуку та запуску .exe файлів

В архітектурі програми знаходиться так званий контейнер-сервер, який приймає дані про статистику кожного контейнеру-клієнта. На контейнерах-клієнтах знаходиться різні версії операційних систем, як показано на рисунку 3.2. Це зроблено для того, щоб процес перевірки одного виконуваного файлу відбувався на різних типах ОС, та збиралась інформація про поведінку саме на різних інформаційних структурах. На рисунку 3.4 проілюстровано мережеву архітектуру взаємодії контейнерів між собою.

```
[
    {
        "Name": "project_net",
        "Id": "77115c9be66ebc65610126c351d8d20e54deab5ce492a7f8eb3d365bab458a36",
        "Created": "2021-06-02T08:25:47.9341689Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.23.0.0/16",
                    "Gateway": "172.23.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": true,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "123345e7b2c766f5468f4c4a8bfc9db1af0a46c972b1032f982212e2d48e051b": {
                "Name": "Server",
                "EndpointID": "ee83c1148b616b3afee777faea274c5e6198520ff78de5f79e62b066e0195e17",
                "MacAddress": "02:42:ac:17:00:02",
                "IPv4Address": "172.23.0.2/16",
                "IPv6Address": ""
            },
            "513817fb69f15a65c6b017182aa34801c619bf608edef8720afab048394e31f4": {
                "Name": "DebianClient",
                "EndpointID": "eb9e76f27e146fcdff82a1cd4f7b7e52f2066e52487a30fd4d033146bb26e279",
                "MacAddress": "02:42:ac:17:00:04",
                "IPv4Address": "172.23.0.4/16",
                "IPv6Address": ""
            },
            "d239b368298dbe031f65f1d3011ece6bde714215abb38d4d25d649b99c3b5808": {
                "Name": "UbuntuClient",
                "EndpointID": "711760db0f1a6f834a87ca23f38cac2f7532dd0b5d217139675a0cbf13605280",
                "MacAddress": "02:42:ac:17:00:03",
                "IPv4Address": "172.23.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {
            "com.docker.compose.network": "net",
            "com.docker.compose.project": "project",
            "com.docker.compose.version": "1.29.1"
        }
    }
]
```


Рисунок 3.4 – Ілюстрація мережевої архітектури

3.2 Аналіз і дослідження роботи програми

Для того щоб розмножуватися, вірус повинен здійснювати будь-які конкретні дії, копіювання в пам’ять, запис в сектори і т. д. Це в свою чергу несе навантаження на систему. Програма, після запуску перевірочного файлу, збирає дані з CPU, Memory, Disk, Network I/O .

Для перевірки роботи було розроблено невеликий .exe файл який містить в собі шкідливе навантаження, наприклад постійне навантаження на CPU процесора.

З кожного docker-контейнеру збирається інформація про стан системи, що знаходиться в ізольованому контейнері, та виводитися на екран, для того, щоб користувач, який перевіряє той чи інший файл, зміг зробити висновок про наявність або ні, шкідливого навантаження на операційну систему. На рисунку 3.5 та 3.6 показані дані системи до і після запуску виконуваного файлу, який містив в собі шкідливий код.

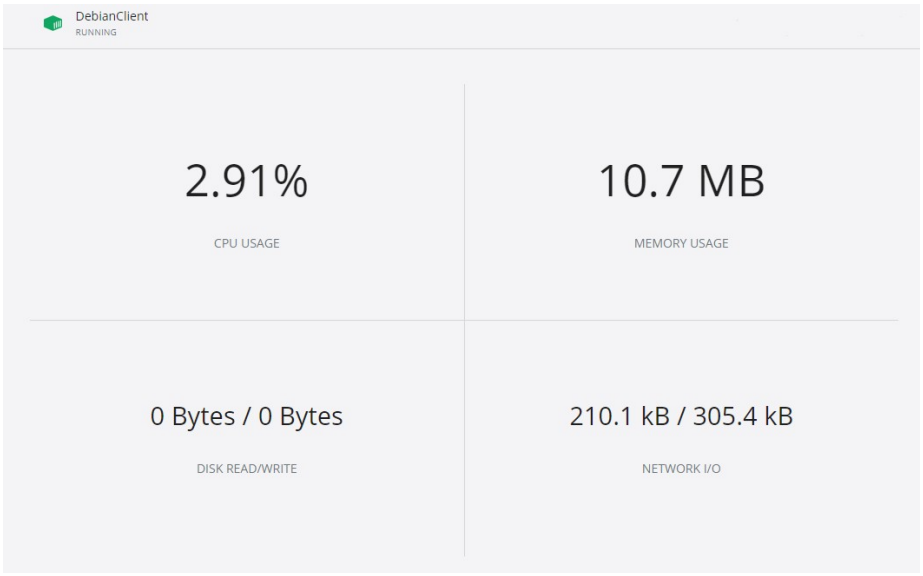

Рисунок 3.5 – Дані про роботу системи в нормальному режимі

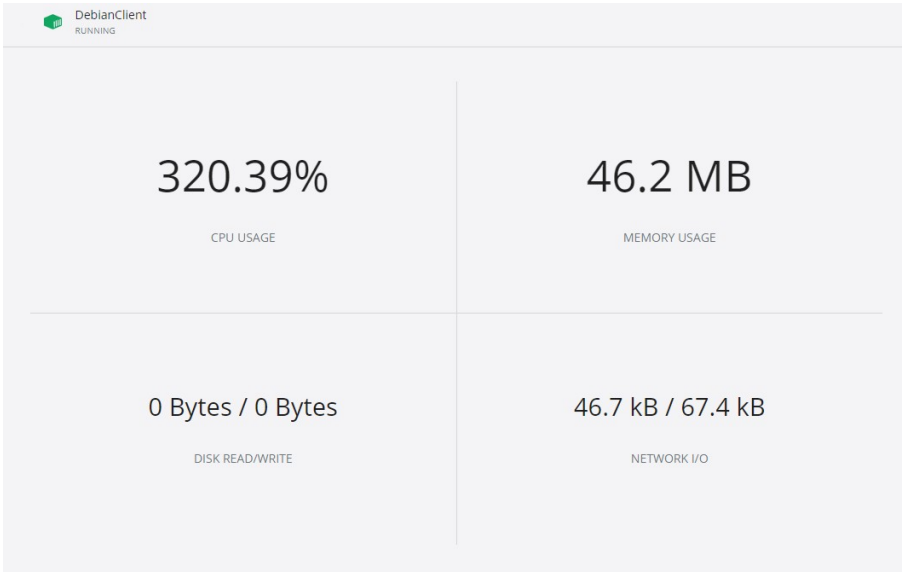

Рисунок 3.6 – Дані про роботу системи після запуску програми зі шкідливим навантаженням

Висновки до розділу 3

У цьому розділі була розглянута програмна реалізація застосунку для автоматичного запуску підозрілих об’єктів в ізольованому середовищі, з метою пошуку аномалій у системі.

Так як вірогідність потрапляння шкідливого програмного забезпечення до системи через файли, архіви чи інсталятори, котрі були завантажені з недовірених джерел або іншими засобами, – досить висока. Є сенс перевіряти їх не тільки на одній системі, імітуючи при цьому роботу ОС, а одразу на декількох. Таке рішення дає змогу уникнути вразливостей на більш ранніх версіях операційних систем.

ВИСНОВКИ

У роботі був проведений порівняльний аналіз антивірусних методів та технологій, які створюють ізольоване середовище в хостовій системі для забезпечення кращого рівня захисту від шкідливого програмного забезпечення, що може потрапити до системи через різноманітні файли, застосунки, архіви чи виконувані файли.

Опираючись на інформацію, що була отримана з порівняльного аналізу, було вибрано за основу створення ізольованого середовища загальнодоступну програму docker, з використанням docker-compose, та розроблено алгоритм який спроможний автоматизовано відправляти і відкривати виконувані файли у контейнерах, що заздалегідь зв’язані між собою в одну систему, та мають різні типи операційних систем.

Таке рішення дає змогу додати ще один рівень захисту в корпоративних мережах, а саме: можливість змоделювати інфраструктуру компанії, використовуючи ізольоване середовище, та перевіряти всі нові, завантажені з недовірених джерел, файли на наявність шкідливого ПО та шкідливого навантаження. Позбутися ризику занесення до системи вірусів, троянів, та іншого ПО таким способом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Теоретичні і прикладні проблеми фізики, математики та інформатики: матеріали XIX Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених (13 − 14 травня 2021 р., м. Київ, Україна) / Уклад.: Василенко О. Д., Пономаренко С. М., Бех С. В., Степаненко В. М., Мірошникова І. Ю., Деркач О. Г., Козленко О. В. – Київ : КПІ ім. Ігоря Сікорського, Видавництво «Політехніка», 2021. – 228 - 231 с. —[Електронний ресурс] — Режим доступу: https://drive.google.com/file/d/1V9zQB2Jvgcaw3o7eEkzMmviLRrhrHbpD/view

2. Антивіруси – [Електронний ресурс] – Режим доступу: https://ua.kursoviks.com.ua/metodychni_vkazivky/article_post/834-tema-antivirusi-navchalniy-posibnik-internet-dlya-koristuvacha-chastina-2-nudpsu

3. Openarchive – [Електронний ресурс] – Режим доступу: https://openarchive.nure.ua/bitstream/document/11001/1/2019_M_EOM_Ponomaryov_V_V.doc

4. Захист локальної мережі/Антивірусні програми – [Електронний ресурс] – Режим доступу: https://sites.google.com/site/zahistlokalnoiemerezi/zahist/ antivirusni-programi

5. Malware Detection Techniques Description [Електронний ресурс] – Режим доступу: https://malwaretips.com/threads/malware-detection-techniques-description.14028/

6. What is a Behavior Blocker? [Електронний ресурс] – Режим доступу: https://www.welivesecurity.com/2006/09/11/what-is-a-behavior-blocker/

7. Роман Ванерке: EDR/обнаружение и реагирование [Електронний ресурс] – Режим доступу: https://www.dialognauka.ru/upload/EDR-detection_and_response_IS_%E2%84%964-2018_Vanerke.pdf
8. Andrey Kopelyan: Что такое Endpoint Detection and Response (EDR)? [Електронний ресурс] – Режим доступу: https://kopelyan.kz/index.php/2021/01/24/endpoint-detection-and-response/
9. SOAR (Security Orchestration, Automation and Response) [Електронний ресурс] – Режим доступу: https://www.anti-malware.ru/security/SOAR
10. KO: NGFW — в чем преимущества файрволов следующего поколения? [Електронний ресурс] – Режим доступу: https://ko.com.ua/ngfw_v_chem_preimushhestva_fajrvolov_sleduyushhego_pokoleniya_127672
11. CISCO: What Is a Next-Generation Firewall? [Електронний ресурс] – Режим доступу: https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-next-generation-firewall.html
12. Полная эмуляция [Електронний ресурс] – Режим доступу: https://studbooks.net/2170053/informatika/polnaya_emulyatsiya
13. Віртуальна машина [Електронний ресурс] – Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:fe14c425ee98949440c8a0fefb3fa44c30863b75/latest/101568/lec3-virtualPC.pdf
14. Виртуализация: новый подход к построению IT-инфраструктуры [Електронний ресурс] – Режим доступу: https://www.ixbt.com/cm/virtualization.shtml
15. Общие сведения о контейнерах и Docker [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/architecture/microservices/container-docker-introduction/
16. textovod.com [Електронний ресурс] – Режим доступу: http://textovod.com/unique/link?url=https%3A%2F

%2Fwww.liveinternet.ru%2Ftags%2Fjupyter%2Fpage2.html&key=c92e32ce62f6801d4d926ad5ba77fea6

17. Virtualization, emulation and Containerization: Distinctive characteristics [Електронний ресурс] – Режим доступу: https://intellect.ml/emulators-of-operating-systems-virtualization-emulation-containerization-3121
18. cloud.google.com [Електронний ресурс] – Режим доступу: https://cloud.google.com/containers?hl=ru
19. Контейнеры или виртуальные машины – что лучше выбрать для своей компании? [Електронний ресурс] – Режим доступу: https://www.it-grad.ru/blog/kontejnery-ili-virtualnye-mashiny-chto-luchshe-vybrat-dlya-svoej-kompanii
20. Віртуальна машина [Електронний ресурс] – Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:fe14c425ee98949440c8a0fefb3fa44c30863b75/latest/101568/lec3-virtualPC.pdf
21. Malware Detection Using Dynamic Analysis [Електронний ресурс] – Режим доступу: https://core.ac.uk/download/pdf/70410162.pdf
22. CLASSIFICATION OF MALWARE USING REVERSE ENGINEERING AND DATA MINING TECHNIQUES [Електронний ресурс] – Режим доступу: https://etd.ohiolink.edu/apexprod/rws_etd/send_file/send?accession=akron1311042709