ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації

Кафедра прикладної математики та інформатики

«До захисту допущено»

Завідувач кафедри ПМІ

_____ Ольга ДМИТРІЄВА

(підпис) (ім'я та прізвище)

«____»_____2022 р.

Випускна кваліфікаційна робота бакалавра

на тему Розробка програмного комплексу для зняття та аналізу образів Оперативної пам'яті

зі спецчастиною Розробка архітектури та модулів комплексу на базі віртуальної машини QEMU/KVM

Виконав: студент 4 курсу, групи КН-18

(шифр групи)

спеціальності 122 Комп'ютерні науки

(шифр і назва спеціальності)

Хоменко Р.В. _____

(прізвище та ініціали) (підпис)

Керівник ст. викладач каф. ПМІ Костін В.І. _____

(посада, науковий ступінь, вчене звання, ім'я та прізвище) (підпис)

Рецензент _____ _____

(посада, науковий ступінь, вчене звання, ім'я та прізвище) (підпис)

Нормоконтроль:

_____ к.т.н., доц. Ірина НАЗАРОВА

(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис)

Луцьк – 2022 р.

ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики та інформатики
Освітній ступінь бакалавр
Спеціальність 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПМІ
_____ /Ольга ДМИТРІЄВА/
«____»_____2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Хоменко Руслану Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного комплексу для зняття та аналізу образів-оперативної пам'яті
Спецчастина Розробка архітектури та модулів комплексу на базі віртуальної машини QEMU/KVM
керівник роботи Валерій КОСТІН ст.викл. кафедри ПМІ
(ім'я та прізвище, науковий ступінь, вчене звання)

затверджені наказом від « 06 » травня 20 22 року № 180
2. Строк подання студентом роботи 8 червня 2022 року
3. Вихідні дані до роботи результати переддипломної практики
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) аналіз предметної області та формулювання задачі; 2) проєктування розроблюваної інформаційної системи; 3) опис процесу програмної реалізації інформаційної системи; 4) опис процесу роботи та результатів тестування інформаційної системи

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) 1) загальна постановка задачі; 2) обґрунтування тематики роботи; 3) функціональне представлення системи; 4) статична структура моделі системи; 5) фізична організація системи; 6) інформаційна модель системи; 7) засоби реалізації системи; 8) екранні форми розробки; 9) висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 9 травня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Опрацювання літературних джерел за темою роботи	09.05.22-15.05.22	
2	Огляд та аналіз теоретичних основ побудови алгоритмів …. та методів оцінки їх ….	16.05.22-22.05.22	
3	Формування вимог до експериментального програмного додатку	23.05.22-29.05.22	
4	Розробка програмного додатку	30.05.22-03.06.22	
5	Оформлення пояснювальної записки та опис створеної системи. Проходження нормоконтролю	04.06.22-07.06.22	
6	Перевірка пояснювальної записки антиплагіатною системою. Оформлення презентації до роботи, графічного матеріалу та рецензування роботи	08.06.22-21.06.22	
7	Захист роботи	22.06.22	

Студент _____ **Руслан ХОМЕНКО**
(підпис) (ім'я та прізвище)

Керівник роботи _____ **Валерій КОСТІН**
(підпис) (ім'я та прізвище)

ЗМІСТ

ВСТУП

З досягненням інформаційних технологій зростає програмне та апаратне забезпечення, що дозволяє вирішувати багато проблем у різних сферах людської діяльності. Люди почали активно використовувати платіжні системи, WEB-ресурси для передачі даних, накопичувачі для запису та зберігання важливої. Все це завдяки комп'ютерній техніці.

Проте активна інтеграція сучасних інформаційних технологій практично в усі сфери людської діяльності призвела до того, що через програмно-апаратні та системні засоби все ширше використовуються різні види злочинів: авторські та суміжні права, крадіжки, шахрайство тощо. псевдобізнес, продаж конфіденційних даних тощо.

Комп'ютерні злочини становлять серйозну загрозу як для комерційних організацій, так і для державних органів. Цей вид злочинів називається кіберзлочинністю [1]. Наука, що вивчає такі злочини, називається комп'ютерною криміналістикою (англійської computer forensics) [1]. Сам термін форензика походить від латинського «foren», що означає «промова перед форумом».

В українську мову це слово походить з англійської. Повна форма цього терміна англійською мовою: «computer forensic science», що буквально означає «computer forensic science». За визначенням комп'ютерна криміналістика – це прикладна наука про розкриття злочинів, пов'язаних з комп'ютерною інформацією, вивчення цифрових доказів, методів пошуку, отримання та об'єднання таких доказів, технічних засобів, що використовуються для цього [1].

Ця область криміналістики повною мірою існує в розвинених країнах: опубліковано ряд наукових праць (Windows Forensic Analysis Toolkit, Harlan Carvey; The Art of Memory Forensics, Michael Hale Ligh, Andrew Case, Jamie

Levy, AAron Walters та ін. .). , існують офіційні рекомендації, яких слід дотримуватися під час судово-медичних досліджень.

На ранніх стадіях цифрових та мережеву форензику вважали спорідненою технологією, але виявилося так насправді вони дуже різні. Цифрова криміналістика розвивається переважно для потреб правоохоронних органів та повинна надавати достовірні (неспростовні) докази для виявлення кіберзлочинності. Технологія мережевої криміналістики розвивалася в процесі боротьби з хакерськими загрозами і пов’язана з архітектурою інформаційної безпеки. Виявлення порушень ІС, оцінка загроз, запобігання модифікування даних до НСД.

Традиційні розділи криміналістики розвивалися протягом тривалого часу, було накопичено великий доказовий досвід і методи дослідження, а також деякі особливості криміналістичних технологій закріпилися на законодавчому рівні.

Комп'ютерна криміналістика - це нова наукова галузь, де тільки починають накопичуватися практичний досвід і технології і вирішуються такі основні проблеми [14]:

- розробка криміналістичних характеристик кримінальних правопорушень, пов'язані з інформаційними відносинами;

- розробка тактики оперативно-розшукових дій (ОРЗ), агентурна робота та розслідування, пов'язані з інформаційними відносинами;

- розробка методів, програмно-апаратного забезпечення для збору доказів кіберзлочинності.

Основним напрямком комп’ютерного криміналістики є збір, дослідження та створення доказів для суду, а також збирання інформації, яка отримана в результаті ОРЗ та агентурної діяльності, які не буде використано як доказ у суді.

Комп’ютерна криміналістика також використовує спеціальні методи дослідження, які є унікальними в цій області:

- створення спеціалізованих криміналістичних ІС, реконфігурація та використання ІС в ОРЗ та слідчих діях подвійного призначення [15];

- створення віртуального користувача для ОРЗ та агентурної роботи;

- збір хеш-функцій відомих файлів для їх розрізнення від файлів, які містять оригінальну користувацьку або модифіковану інформацію;

- архівування певного вмісту фізичного носія для подальшого розслідування ймовірних інцидентів;

- емуляція мережевих сервісів корпоративної комп’ютерної мережі (КМ) і VPN-мереж віддаленого доступу для експертного дослідження вилученого програмно-технічного обладнання.

В інших країнах комп’ютерна криміналістична експертиза тільки починає розвиватися.

Україна, на жаль, лише одна з них, незважаючи на те, що в ній багато провідних комп’ютерних експертів. Одним із показників є, наприклад, масове виробництво програмно-апаратних систем, що спеціалізуються на зборі, обробці та аналізі цифрових доказів для забезпечення цілісності даних під час вилучення та допитів (наприклад, карта Tribble PCI, проект Windows SCOPE, LiMe тощо).

В Україні такі пристрої тільки починають виробляти, але їх купують все частіше. Така ситуація пояснюється кількома обставинами: нерозвиненістю теоретичних і прикладних основ кіберзлочинності, невеликою кількістю публікацій у цій галузі, а також системою (вищої) освіти, яка не пристосована до ефективного навчання. основи судової експертизи.

Центральними технічними завданнями комп'ютерної криміналістики є отримання та аналіз інформації, що зберігається в пам'яті комп'ютера [1]. Перше завдання також складається з вилучення даних з: (а) нестабільної (випадкової) пам’яті (англійською – memory acquisition) [2] та (б) жорстких дисків (на диску) (англійською – data carving) [3].

Слід зазначити, що дані на комп'ютері можуть зберігатися і змінюватися в таких частинах пам'яті комп'ютера:

• регістри процесора;

• ієрархічний кеш процесора;

• оперативна пам’ять;

• зовнішня пам'ять.

Перші два типи пам’яті є специфічними та частково не адресованими для прикладного програмного забезпечення, а тому проблема надійного отримання даних не має для них надійного рішення.

Тематика цієї роботи — клас методів для виділення та аналізу оперативної пам’яті (ОП).

В даний час технології віртуалізації набувають все більшої популярності. Пропускна здатність фізичних інтерфейсів комп’ютера зростає, а також ємність і відгук систем зберігання даних.

В результаті потужність одного сервера може бути ефективно відображена у віртуальне середовище на багатьох емульованих серверах, гнучко масштабуючи інфраструктуру для конкретних бізнес-завдань комерційних організацій та державних установ [4].

Однак при всіх перевагах віртуальних середовищ є й відчутні недоліки – розслідування протиправних дій, скоєних за допомогою віртуалізації, часто пов’язане зі значними труднощами в отриманні прийнятних доказів через відсутність криміналістичних інструментів для роботи з віртуальними середовищами.

Аналогічно, потреба у комплексних інструментах моніторингу віртуалізації для судово-медичної реєстрації інцидентів у контексті комп’ютерного криміналістичного аналізу задоволена не повністю.

Підводячи підсумок, можна сказати, що проблема аналізу динаміки інформації у віртуальному середовищі є одним з найважливіших напрямків комп’ютерного криміналістичного аналізу.

У даній роботі увага приділяється одному аспекту цієї проблеми, а саме методам вилучення даних та аналізу оперативної пам’яті віртуальної машини.

У статті також розглядаються методи моніторингу пам’яті віртуальних машин як побічного продукту технології аналізу ОП.

У центрі уваги роботи є формулювання пропозицій щодо практичного застосування голосових методів і методів на віртуальних машинах, які використовують 64-розрядні операційні системи сімейства Windows NT. (Цей вибір пов’язаний з тим, що ці операційні системи популярні серед більшості користувачів.)

Як зазначалося вище, у класичному середовищі існує багато інструментів для ізоляції/аналізу ОП. На перший погляд, їх можна було б використати і в нашому випадку. На жаль, це неможливо через безліч обмежень, які є для нас принциповими. Серед них висока ціна, відсутність відкритого коду за ліберальними ліцензіями і, як наслідок, неможливість модифікувати програмні компоненти під певні завдання, платна технічна підтримка тощо.

Метою даної роботи є створення інтегрованої з платформами віртуалізації криміналістичної архітектури програмного пакета в контексті проблеми інтроспективного аналізу/моніторингу динаміки оперативної пам’яті в гостьових операційних системах.

Для досягнення цієї мети були сформульовані наступні завдання:

1) вивчення внутрішньої організації фізичного та віртуального сховища в операційних системах сімейства Windows NT (наприклад, Windows 7, k86_64);

2) огляд відомих методів і підходів для видалення та аналізу образів RAM;

3) розробка архітектури програмного комплексу.

1 ОГЛЯД ПІДХОДІВ ДО ВИЛУЧЕННЯ ТА АНАЛІЗУ/МОНІТОРИНГУ ОПЕРАТИВНОЇ ПАМ'ЯТІ

1.1 Класифікації методів

Існує багато методів вилучення та аналізу необхідних даних оперативної пам’яті. Однак не всі підходять для правильного збору доказів з точки зору комп’ютерної криміналістики. У цьому відношенні можна виділити наступні класи методів:

1) криміналістично правильний,
2) криміналістично сумнівний,
3) криміналістично неправильний.

Також, безумовно, важливий і факт призупинення функціонування обладнання, яке піддається криміналістичному дослідженню, що вводить другу класифікацію

• методи (засоби), які не вимагають зупинки цільової системи (онлайн),

• методи (засоби), що вимагають зупинки цільової системи (офлайн).

Для технологій віртуалізації, а саме серверів, які є основою віртуальної системи, недоцільно розглядати підходи другої групи - система повинна працювати безперервно, а зупинка для аналізу її даних неприпустима.

Іншим важливим аспектом є інвазивність дослідницьких дій. Якщо пам'ять цільової системи змінюється під час зняття або аналізу інформації, то цей криміналістичний метод неприйнятний. Такі методи у кращому разі можна віднести до криміналістично підозрілих, хоча вони, як правило, криміналістично неправильні. Таким чином, вводиться третя класифікація наявних методик – за ознакої інвазивності виконуваних дій:

• інвазивні,

• неінвазивний.

Розглянемо та класифікуємо кілька інструментів, які використовуються для зняття фізичної пам’яті систем.

FireVire-копіювання.

FireVire (IEEE 1394A, IEEE 1394B) — це послідовна високошвидкісна інформаційна шина, яка зазвичай використовується пристроями зберігання даних і має можливість гарячого підключення. Коли дані знімаються через інтерфейс FireVire, код не вставляється в пам’ять операційної системи, оскільки доступ до фізичної пам’яті здійснюється безпосередньо, минаючи центральний процесор. У цьому сенсі метод відноситься до неінвазивних і онлайн-засобам. Однак він являється криміналістично сумнівним: одні області пам’яті копіюються на фоні інших, що принципово не дозволяє отримати моментальну копію вмісту пам’яті та порушує вимогу цілісності даних при зборі цифрових доказів.

Копіювання засобами гіпервізора QEMU/KVM.

Дана методика застосовується до вилучення пам’яті віртуальних машин. Вона не вимагає зупинки системи, тому її можна відносити до класу онлайн-засобів. Крім того, вона неінвазивна, оскільки пам'ять гостьової операційної системи не піддається модифікації. Нарешті, методика криміналістична правильна, оскільки гіпервізор гарантує цілісність знятого образу.

1.2 Віртуалізація

Віртуалізація – це технологія, в якій розподілене використання обладнання через програмне забезпечення або на тому самому фізичному обладнанні можна розміщувати і керувати кількома серверами. Наприклад: два сервери Windows і один сервер Linux можуть працювати на трьох різних комп’ютерах використання віртуалізації для примусового запуску одного сервера.

1.2.1 Віртуалізація рівня ОС

Створення багатьох незалежних користувацьких середовищ на основою операційної системи, саме таке поняття може бути визначити віртуалізацію на рівні операційної системи. Концепція ця технологія заснована на заміні базового каталогу при завантаженні системи. Команда chroot використовується для зміни каталогів.

Механізм підміни кореневого каталогу дозволяє системі запускати віртуальні сервери, на яких запущено набір пов’язаних процесів тільки у власних каталогах. Отже, віртуальні сервери позбавлені доступу до своїх файлів, що підвищує базовий захист від атак, як на самому сервері, так і на його процесах.

1.2.2 Віртуалізація на основі віртуальних машин

Перш ніж говорити про цей тип віртуалізації, важливо дізнатися, що таке віртуальна машина. Так віртуально машина - це тип апаратного середовища або ізольована програма, призначена для виконання певних комп’ютерних процесів. Віртуальні машини найчастіше використовуються для наступних дій:

- захист інформації
- створення (емуляція) архітектур процесорів
- оптимальне використання можливостей фізичних обчислювачів
- моделювання клієнт-серверних систем в одному обчислювачі.

Віртуальна машина створюється як програмний контейнер, що імітує роботу комп'ютеру. Цей контейнер має власну операційну систему, пакет програм і системні пристрої (жорсткий диск, процесор, мережевий контролер або оперативна пам’ять).

Віртуальний комп’ютер повністю ізольований від інших віртуальні машини і зовнішнього світу і його робота реалізується виключно через відповідне програмне забезпечення. Для забезпечення роботи віртуальної

операційної системи віртуального комп’ютеру та виконання поставленого користувачем завдання задіяні обчислювальні ресурси фактичної системи, в якій віртуальна машина дійсно працює.

Отже, виходить, що віртуальна машина — це не що інше як набір системних програм і певна область оперативної пам'яті, які перебувають під повним та індивідуальним контролем самої віртуальної машини.

Деякі нюанси щодо характеру програмного забезпечення віртуальної машини потребують уточнення.

- за рахунок використання різних засобів віртуалізації віртуальні машини можуть отримати прямий доступ до деяких пристрої користувача (розділи жорсткого диска, usb-накопичувачі та інший)
- можливість включення віртуальної машини в комп'ютер або віртуальні мережі як повноправні члени.
- останні розробки процесорів зробили це можливим для них забезпечувати спеціальні комп’ютерні команди та структури дані, необхідні для створення віртуальних машин і керування ними.

Однак слід розуміти, що без спілкування із зовнішнім світом робота віртуальної машини обмежена виділеною для неї ділянкою пам'яті, відсутність доступу до фізичного комп’ютера та інших віртуальних даних машини.

Віртуалізацію на основі віртуальних машин можна розділити на декілька видів:

Гостьова операційна система. Кожен з віртуальних машина є єдиним екземпляром операційної системи всередині відповідає за віртуалізацію додатків, що в свою чергу також працює на певній ОС. Як приклади за допомогою гостьової операційної системи ви можете принести Microsoft Віртуальна машина, Parallels Workstation та інші. Найчастіше це операційна система, яка керує діями програмне забезпечення віртуалізації називається хост-системою. Приклад: VMVare Workstation, Parallels Робоча станція, Microsoft Virtual PC.

Паралельна віртуальна машина. За допомогою засобів кластеризації, можна підключити кілька віртуальних або реальних від обчислювачів до єдиної функціональної віртуальної машини складні операції та розрахунки. Наприклад: Parallel Virtual Machine.

Віртуалізація на основі гіпервізора. При використанні такого типу візуалізації в структурі з’являється інший елемент як гіпервізор.

. Гіпервізор – це програма, яка керує фізичними ресурсами комп’ютера та розподіляє ці ресурси між кількома різними операційними системами, дозволяючи їм працювати одночасно.

Іншими словами, гіпервізор створює кілька копій, клонів апаратних ресурсів з одного фізичного комп’ютера, і кожен клон бачить користувач як окремий пристрій. Ви можете встановити гостьову операційну систему на кожній віртуальній машині, яка не прив’язана до апаратного забезпечення хоста.

Гіпервізор ізолює працюючу ОС від усіх інших, тому кожен з них має монополію на об’єкти, які в ній знаходяться. Однак, якщо необхідно, гіпервізор операційних систем віртуальних машин наає можливість взаємодії один з одним. Механізм зв'язку між операційною системою може бути підписаний в певних файлах і обмін даними по локальній мережі.

1.3 Засоби аналізу образів пам'яті

Одним із центральних завдань нашого комплексу є аналіз образів оперативної пам’ті. Існує кілька програмних засобів, які вирішують цю проблему (Belkasoft Evidence Center, MANDIANT Memorize тощо). Але через їхні недоліки – високої вартості, відсутність відкритого виходного коду – використовувати їх видається можливим.

Окремо хотілося б відзначити програмне рішення - Volatility Framework. Воно проводить аналіз образів оперативної пам’яті, і, що найголовніше, це безкоштовне програмне забезпечення з відкритим вихідним кодом. Однак існує ряд причин, чому Volatility підходить лише в якості інструмента навчання, а не базового функціонального блоку у сторонніх проектах. По перше, це ліцензія, за якою була розроблена Volatility - GNU GPL v2 - вона має ряд обмежень у порівнянні з більш ліберальними MIT або BSD 2-clause "Simplified". По-друге, це програмне забезпечення написано мовою програмування Python і не дозволяє використовувати його нативний API в програмах, написаних іншими мовами. Також під час тестування цього інструменту було виявлено, що він не повністю адаптований для роботи з дампами (англ. dump) пам'яті, отриманими за допомогою бібліотеки libvirt.

1.4 Засоби моніторингу віртуальних систем

Друге, більш глобальне завдання, яке певною мірою включає перше, — спостереження (або моніторинг) віртуальних машин. У цій сфері також існує безліч програмних засобів (Veeam One, Naumen та ін.). Але, на жаль, більшість з них мають іншу мету – стежити за продуктивністю системи. Ще одним недоліком цих інструментів є те, що багато з них можуть працювати лише з певними гіпервізорами.

Завданням даної роботи є створення архітектури універсального комплексну, який дозволяє, крім усього іншого, дозволяє спостерігати за станом віртуальних машин з метою виявлення нестандартних активностей в оперативній пам’яті.

Тому в даній роботі увага буде приділена створенню саме такого криміналістичного програмного засобу, інтегрованого з платформами віртуалізації (насамперед QEMU/KVM)

2 ВИЗНАЧЕННЯ ОПЕРАЦІЙНИХ СИСТЕМ СІМЕЙСТВА WINDOWS NT. АНАЛІЗ ДАМПІВ ФІЗИЧНОЇ ПАМ'ЯТІ МАШИН ПІД КЕРУВАННЯМ WINDOWS ОС

Перш ніж безпосередньо проектувати програмний комплекс слід надати загальну інформацію про влаштування операційних систем сімейства Windows NT і про методику аналізу образів їх пам'яті.

2.1 Операційні системи та їх типи

Операційна система — це комплекс взаємопов’язаних програмних компонентів, за допомогою яких користувачі та їхні програми взаємодіють із апаратним забезпеченням. Вона надає інтерфейси для користувачів і запущених програм. Програми користувача, службові програми та різноманітні додатки спілкуються з апаратним забезпеченням комп’ютера лише через доступ до операційної системи (точніше, до ядра операційної системи). Більшість операційних систем складається з багатьох програмних модулів. Існує група «основних» керуючих модулів, разом з деякими структурами системних даних вона утворює ядро операційної системи – центральну та основну частину системи [5].

Залежно від структурних особливостей ядра розрізняють різні типи операційних систем: мікроядерні та макроядерні (монолітні). В мікроядрних операційних системах саме ядро є дуже компактним, а інші модулі викликаються з ядра як сервісні модулі. На відміну від мікроядерної архітектури, в макроядерних операційних системах основна «керуюча» частина містить велику кількість модулів і використовує набагато більше пам'яті [5] Ядро Windows NT поєднує в собі елементи мікроядерної та монолітної архітектури. Однак Windows NT часто називають мікроядерною операційною системою, хоча це не зовсім так. Це ядро занадто велике, щоб його можна було класифікувати як мікроядро, так і запуск компонент ядра в

одному адресному просторі є властивим для систем з монолітним ядром. З іншого боку, розташування і взаємодія ядерних елементів точно такі ж, як і в мікроядерних системах. Цей тип організації ядра називається «гібридним ядром». Саме к цьому третьому типу належить сімейство операційних систем Windows NT [6].

2.2 Фізична та віртуальна пам'ять

У даній роботі будуть використані два терміни - віртуальна пам'ять і фізична пам'ять. Під фізичної пам’ятью мається на увазі оперативна пам’ять, яка являє собою впорядкований набір однобайтових комірок, кожна з котрих має свою унікальною адресу. Сукупність адрес у фізичній пам’яті, яка використовується для ідентифікації даних, що зберігаються в пам’яті, називається фізичним адресним простором [7]. Віртуальна пам’ять — це абстрактне сховище, яке створене самою операційною системою для полегшення управління фізичною пам’яттю. Віртуальна пам'ять дозволяє програмам вважати, що їм надано всю теоретично доступну пам'ять, організовану в безперервний адресний простір.

У Windows ОС віртуальна пам'ять організована за сегментами сторінок. Адресний простір процесу представляється як набір сегментів змінного розміру. При цьому кожен сегмент розбивається на сторінки — блоки фіксованого розміру, а фізична пам’ять — на блоки однакового розміру — фрейми (кадри) [7].

2.3 Режим ядра та режим користувача

Для запобігання ситуації коли користувальницькі додатки отримують доступ або вносять в важливі системні дані в Windows використовує два режими доступу процесора: режим користувача та режим ядра. Програма

користувача працює в режимі користувача, тоді як код операційної системи (наприклад, системні служби та драйвери пристроїв) працює в режимі ядра. Режим ядра – це режим роботи процесора, при якому є необмежений доступ до системної пам’яті та зовнішніх пристроїв [8].

Кожна сторінка у віртуальній пам’яті має позначку, яка вказує, у якому режимі доступу має перебувати процесор, для читання або запису цієї сторінки. Таким чином, віртуальна пам’ять операційної системи поділяється на: простір ядра (зарезервований для роботи ядра, його розширень і деяких драйверів пристроїв) і простір користувача (в ньому працюють усі програми користувача).

Як згадувалося вище, більшість користувацьких програм працює в режимі користувача, але ви можете переключитися з режиму користувача в режим ядра за допомогою системних викликів. Такий перехід здійснюється за допомогою спеціальної інструкції процесора. Перш ніж відновити контроль над програмою користувача, процесор перемикається в попередній режим [8].

2.4 Трансляція віртуальної адреси у фізичну

При аналізі оперативної пам’яті необхідно вміти відновлювати адресний простір процесу, тобто робити перетворення відомих віртуальних адрес у відповідні їм фізичні, отримав тим самим дані з усіх сторінок віртуальної пам’яті з фреймов оперативної пам’яті.

Для операційних систем Windows 7 k86-64 перетворення віртуальної адреси в фізичну виконується за допомогою чотирирівневої табличної структури перетворення. Віртуальна адреса розділена на п’ять частин: селектор четвертого рівня відображення сторінки, селектор вказівника каталогу сторінок, селектор таблиці сторінок, селектор запису таблиці сторінок і зміщення байтів. За допомогою цих індикаторів здійснюється перетворення адрес. Під час процесу перетворення можуть виникати

недостовірні записи в таблицях сторінок, тому необхідно враховувати специфіку станів знайдених сторінок [9].

Аналіз образу оперативної пам’яті передбачає пошук артефактів (сутностей) операційної системи, який здійснюється за допомогою патерн-аналізу (англійською pattern – зразок). Його суть полягає в пошуку певного набору байтових патернів в образі пам'яті. Вибір патерна залежить від сутності, яку необхідно виявити. Їх класичними прикладами є список процесів, дерево реєстру та список мережевих підключень.

2.5 Процеси та їх потоки

Перше, з чого потрібно почати аналіз дампу оперативної пам’яті операційної системи – це пошук списку активних процесів. Кожен процес в оперативній пам’яті представлений структурою _ERPOCESS, що зберігається в адресному просторі ядра. У цієї структури має багато полів і підструктур, першою з яких є підструктура _KPROCESS - блок управління процесом, який у свою чергу починається з субструктури _DISPATCHER_HEADER. Значення деяких полів, які вона містить, є постійними для всіх процесів даної операційної системи. Точніше, такими полями є Type і Size. Таким чином, пошук процесів в оперативної пам’ті зводиться до пошуку відповідного байтового патерна в адресному просторі ядра.

Список результатів буде набагато довшим, ніж фактичний список активних процесів, але після перевірки певних умов на адреси потоків процеса і полів DTB, які відповідають значенню регістра керування CR3, частина «кандидатів» буде виключена.

Як тільки знайдено зміщення _EPROCESS, не складе труднощів знайти інформацію про процеси (унікальний ідентифікатор, ім’я, час створення тощо) та їх потоки (кількість активних потоків, список усіх потоків тощо).

2.6 Реєстр

Реєстр — це ієрархічно організована база даних параметрів операційної системи, яка містить інформацію про апаратне та програмне забезпечення та його налаштування, профілі користувачів тощо. [10].

Реєстр – представляє собою набір файлів у форматі Windows NT Registry File (REGF), які називаються хайвами (англійської hive). Кожен хайв містить дерево реєстра і представлений в пам’яті структурою _CMHIVE. У даній структурі міститься багато метаданих, цінних для аналізу.

Щоб знайти цю структуру, можна скористатися наведеним вище методом пошуку відповідного байтового патерну в адресному просторі ядра. У цьому випадку байтовим патерном буде сигнатура, з якої починається підструктура _HHIVE структури _CMHIVE.

Кожен хайв сконструйований таки чином, що дані всередині нього зберігаються в комірках (англійською – cells) . У пам'яті операційної системи (і фізичної пам'яті, відповідно) реєстр рідко зберігається в лінійно-однорідній формі, найчастіше його структура дуже фрагментована. Тому для належного вилучення реєстру необхідно здійснити «трансляцію» адрес комірок [8].

2.7 Мережеві з'єднання

Інформація про мережеві з’єднання включає в себе ідентифікатор процесу, час створення, встановлені з’єднання, локальні та віддалені адреси, локальний та віддаленій порти. Вилучення такої інформації виконується аналогічно до описаних вище випадків – за допомогою байтового пошуку відповідних структур.

У тому випадку, коли мова йде про мережеві з'єднання, предметом будуть структури TCB і TcpEndpoint. Сигнатура цих структур

використовується в ролі байтового патерну. Шукаючи ці сутності, ми отримаємо необхідну інформацію [11]

У Libvirt ключовим компонентом віртуальної мережі є міст Ethernet, який імітує віртуальний мережевий комутатор, до якого можуть бути підключені різні мережеві інтерфейси. [16] За замовчуванням Libvirt визначає лише одну віртуальну мережу, яка створює міст Ethernet на вузлі віртуалізації з іменем virbr0 з IP-адресою 192.168.122.1. До цього мосту можна підключити мережеві інтерфейси віртуальних машин (робочі як мережевий комутатор), щоб дозволити вхідний і вихідний мережевий трафік між ВМ і з вузлом віртуалізації.[17] Конфігурація віртуальної мережі за замовчуванням — /etc/libvirt/qemu/networks/default.xml.

Libvirt також створює один інтерфейс TAP на вузлі віртуалізації (названий vnet<number>) для кожного мережевого інтерфейсу, доданого до віртуальної машини. [16] Якщо міст Ethernet представляє мережевий комутатор, ці інтерфейси представляють порти перемикач, де підключено інші пристрої.

На рисунку 2.1 показана мережева схема віртуальної мережі з мостом і двома. До нього підключені віртуальні машини. У цьому прикладі вузол віртуалізації має дві фізичні мережі інтерфейси (eth0 і eth1), підключені до різних мереж. Він також має міст (virbr0) основною функцією якого є робота віртуального комутатора з двома віртуальними портами (vnet0 і vnet1), де підключені мережеві інтерфейси віртуальних машин.

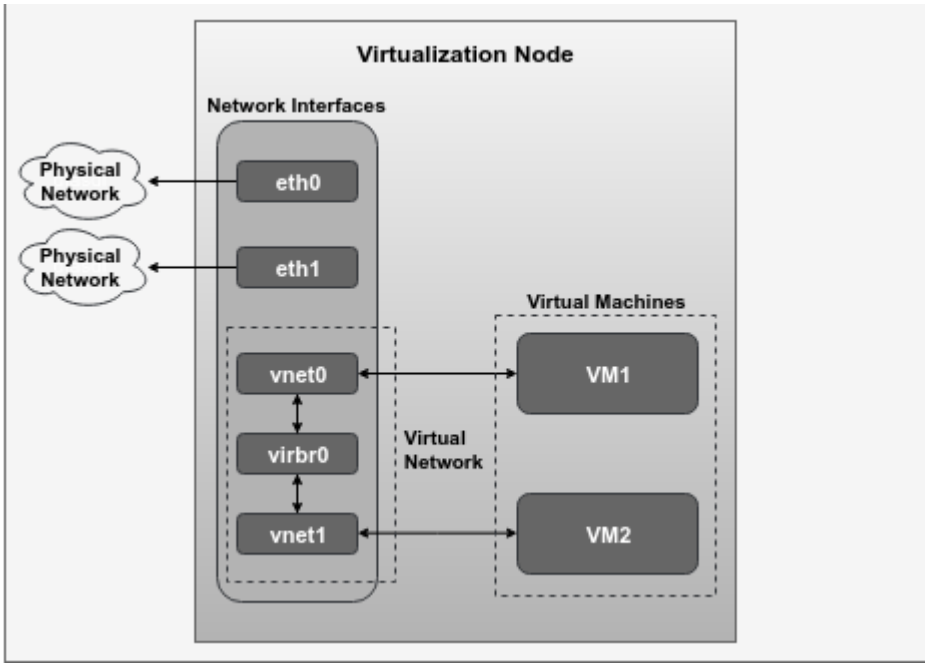


Рисунок 2.1. Схема віртуальної мережі KVM

Після того, як інтерфейси віртуальної мережі кожної віртуальної машини підключені до мосту через інтерфейси TAP їх можна встановити в один із таких режимів:

• ізольований режим: вхідний та вихідний мережевий трафік між віртуальними машинами та віртуальними машинами

вузол віртуалізації дозволений. однак трафік за межами вузла віртуалізації не є дозволено.

• nated mode: вхідний та вихідний мережевий трафік між віртуальними машинами та віртуальними машинами

вузол віртуалізації дозволений. вихідний трафік за межі вузла віртуалізації також дозволено правилами трансляції мережевих адрес (nat).

• режим маршрутизації: вхідний та вихідний мережевий трафік між віртуальними машинами та віртуальними машинами вузол віртуалізації дозволений. вхідний і вихідний трафік за межами

вузол віртуалізації вимагає модифікації таблиць маршрутизації залучених пристроїв.

• мостовий режим: віртуальні машини підключаються до мосту ethernet і фізичної мережі інтерфейс вузла віртуалізації також підключений до мосту. ця конфігурація робить віртуальні машини видимими у фізичній мережі, дозволяючи вхід і вихід трафік між віртуальними машинами та пристроями, підключеними до фізичної мережі без необхідність зміни таблиці маршрутизації залучених пристроїв.

Ізольований, NAT і маршрутизований режими поширені в середовищах тестування, тоді як мостовий режим є поширеним у виробничих середовищах, оскільки він дозволяє отримати доступ до віртуальних машин безпосередньо через призначені їм IP-адреси [16]. Цей режим використовується, наприклад, провайдерами хмари пропонувати VPS з загальнодоступними IP-адресами для розміщення послуг доступні з Інтернету.

Існує ще одне поняття, пов’язане з віртуальними мережами: тип віртуальної мережі інтерфейс, підключений до ВМ. Реалізації, які підтримує Libvirt: rtl3189 (чіпсет Realtek), e1000 (чіпсет Intel) або VirtIO. Так само, як і жорсткі диски, від а З точки зору цифрової криміналістичної експертизи, тип віртуального мережевого інтерфейсу, призначеного для віртуальної машини, не є актуальний, оскільки процес збору даних з нього однаковий. Проте воно варте згадуючи, що VirtIO є паравіртуалізованою реалізацією, тоді як інші є справедливими емуляції. З цієї причини VirtIO пропонує більш високу продуктивність мережі, і це, швидше за все, буде використовуватися у виробничих середовищах.

.

3 АРХІТЕКТУРА ПРОГРАМНОГО КОМПЛЕКСУ АНАЛІЗУ/МОНІТОРИНГУ ОПЕРАТИВНОЇ ПАМ'ЯТІ ВІРТУАЛЬНИХ МАШИН

Концепція архітектури комплексу базується на використанні віртуальних середовищ, тому перед основним викладом розглянемо ключові поняття технології віртуалізації.

3.1 Платформа віртуалізації QEMU/KVM

KVM (Kernel-based Virtual Machine)— це програмне рішення, яке забезпечує підтримку технології віртуалізації в операційній системі на базі Linux. Проект KVM зарекомендував себе як надійне програмне забезпечення і з 2007 року входить в основну гілку ядра Linux. Само собою KVM не виконує емуляції, воно просто дозволяє використовувати апаратну віртуалізацію процесорів (Intel VT-k / VT-d, AMD-V). На рівні ядра операційної системи.

Для безпосередньої емуляції ЕВМ та створення віртуальної машини (англійською - virtual machine) необхідний гіпервізор (англійською – hypervisor). Вибір рішень віртуалізації сьогодні відносно багатий: VMware ESKS / ESKSi, Xen, Parallells OpenVZ, Oracle VirtualBok та ін. У даній роботі технологічним еталоном гіпервізора було обрано програмний емулятор QEMU (скорочено від англ. Quick Emulator), програмний емулятор, гіпервізор другого типу з відкритим вихідним кодом, доступний за ліцензією GPLv2, який підтримує віртуалізацію широкого спектру апаратного забезпечення. Спочатку він був розроблений французьким розробником Фабрісом Белларом(Fabrice Bellard) у 2005 році. Пізніше проект отримав подальший розвиток за підтримки великої міжнародної спільноти в тісному зв’язку з проектом KVM.

Разом дані два продукти забезпечують емуляцію та віртуалізацію, охоплюючи понад 90% усіх типових прикладних завдань системного адміністрування. Надалі в даній роботі «гіпервізор» означає поєднання цих двох програмних рішень і називатиметься QEMU/KVM.

Далі розглянемо докладніше роботу гіпервізора.

Віртуальна машина або таргет платформа(англійською terget – цільова) створюється шляхом запуску процесу QEMU. Теоретично QEMU / KVM дозволяє запускати будь-яку кількість віртуальних машин одночасно, обмежена лише ресурсами хоста (англійською host). Кожна з віртуальних машин із гостьовою операційною системою буде визначена власним процесом QEMU на хост-платформі (див. рисунок 3.1).

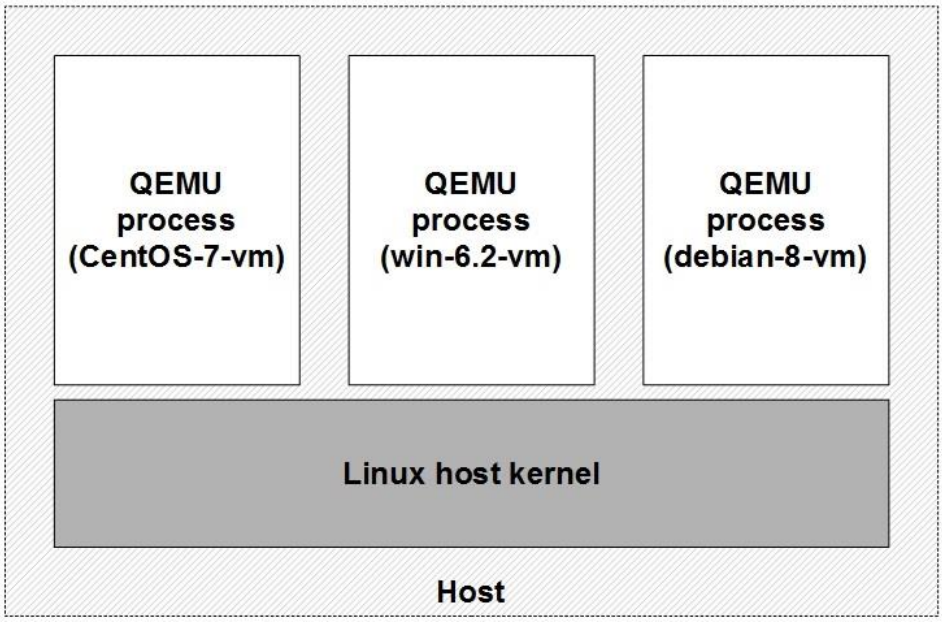


Рисунок. 3.1. Процеси QEMU на хост-платформі.

Коли віртуальна машина вимикається, процес QEMU завершується, але гостьова ОС перезапускається без перезапуску процесу QEMU.

Після запуску QEMU-процесу відбувається виділення та резервація фіксованого блоку віртуальної пам'яті (адресної арени) хост-процесу для оперативної пам’яті віртуальної машини, тобто «фізичної» пам'яті гостьової ОС (див. рисунок 3.2.) [12].

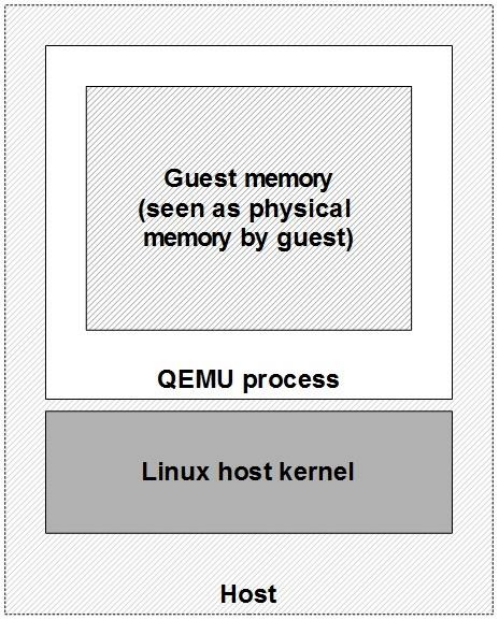


Рисунок 3.2 Фізична пам’ять гостьової операційної системи

Віртуальна пам'ять гостьової ОС, у свою чергу, відображається на вищезгадану «фізичну» штатним для гостьової системи засобом.

Зрозуміло, що для кількох віртуальних машин кожна з них матиме власну «фізичну пам’ять», кожна з яких в кінцевому підсумку буде відображена на загальну фізичну пам’ять хост-платформи (див. рисунок 3.3).

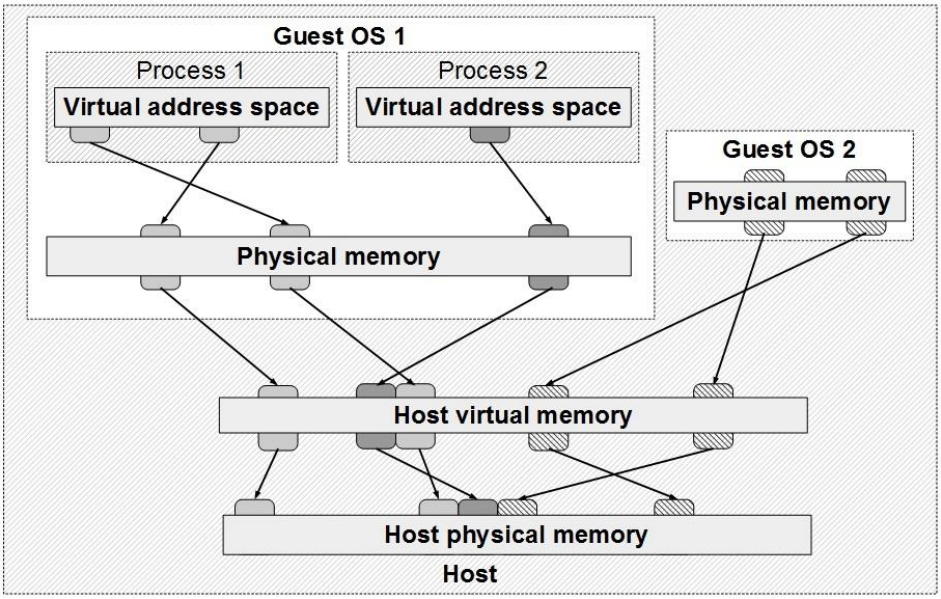


Рисунок 3.3. Відображення пам'яті у разі кількох віртуальних машин

QEMU підтримує два типи запису байтів(порядку байтів) цільових платформ від високого до низького (англійською big-endian або «тупокінцевий») і від низького до високого (англійською little-endian або «гострокінцевий»). Тому при зверненні до пам’яті гостьової ОС необхідно враховувати порядок байтів. Endian-перетворення порядкового ряду виконуються за допомогою допоміжних функцій, а не прямого доступу до оперативної пам’яті віртуальної машини [12].

Як згадувалося вище, KVM є функцією ядра Linux, яка дозволяє програмам, таким як QEMU, безпечно виконувати код гостьових систем безпосередньо на процесорі сервера віртуалізації. Однак це можливо лише в тому випадку, якщо архітектура цільової платформи підтримується основним процесором хоста, що наразі означає x86.

Щоб запустити код гостьової ОС за допомогою KVM, процес QEMU відкриває каталог /dev/kvm і здійснює системний виклик KVM_RUN ioctl (input/output control – системний виклик потоку вводу/виведення). Потім модуль ядра (наприклад, kvm_intel) виконує код гостьової машинний код

безпосередньо на хості. Коли «гість» отримує доступ до периферійних пристроїв або зупиняє гостьовий процесор, очікуючи відключення обладнання, KVM передає управління QEMU. У цей момент процесор QEMU може емулювати бажаний результат операції або продовжувати чекати відключення апаратного переривання [12].

Все вищесказане можна проілюструвати частиною псевдокоду. Основний потік гостьового процесора виглядає так:

```
open("/dev/kvm")
ioctl(KVM_CREATE_VM)
ioctl(KVM_CREATE_VCPU)
for (;;) {
    ioctl(KVM_RUN)
    switch (exit_reason) {
    case KVM_EXIT_IO:  /* ... */
    case KVM_EXIT_HLT: /* ... */
    }
```

Для хост-платформи QEMU є звичайним процесом, таким як sshd або bash, який використовує спільні ресурси хост-системи. Однак при створенні сервера віртуалізації процеси QEMU все ще поділяються на окрему групу:

• пріоритет доступу до ресурсів,

• сусіднє розташування адресних просторів процесу.

Оскільки система емуляції QEMU розроблена для повного та безперервного запуску віртуальної машини (без розділення на функціональні блоки) в адресному просторі процесу QEMU, такі деталі, як запущені процеси в гостьовій ОС, не доступні для читання додаткам хост-платформи. Це робиться з міркувань безпеки та для оптимізації продуктивності віртуальних машин.

Запуск гостьової ОС включає запуск гостьового коду, обробку таймерів, виконання операцій введення-виводу та відповідь на команди системи моніторингу QEMU. Для одночасної обробки всіма цими завданнями потрібна архітектура, яка може реагувати на «сигнали» програмно-апаратних ресурсів, не зупиняючи при цьому роботу гостьової ОС, навіть у разі тривалої обробки (наприклад, операції вводу-виводу з диском або команди моніторингу) [13].].

Існує три популярні архітектури програмного забезпечення, які відповідають нашим вимогам [13]:

• Потокова архітектура: виконує роботу над процесами або потоками, які можуть виконуватися асинхронно або паралельно.

• Подійна архітектура, реагує на події: синхронно відповідає на події в основному циклі, з якого події передаються обробникам подій. Типовим прикладом є системні виклики select (2) або poll (2), для очікування не на одному, а одразу на кількох файлових дескрипторах.

• Гібридна архітектура: поєднання глобального циклу подій і конкретних обробників подій, що працюють в окремих потоках виконання.

Автори QEMU приступили до реалізації гібридної архітектурної моделі (див. рисунок 3.4), оскільки вони мають лише один потік реалізації, неможливо скористатися перевагами такого цінного багатоядерного та/або багатопроцесорного. Крім того, іноді простіше виділити одну конкретну задачу в окремий потік, а не інтегрувати її в глобальний цикл подій. Однак ядро QEMU керується подіями, і більшість коду виконується синхронно.

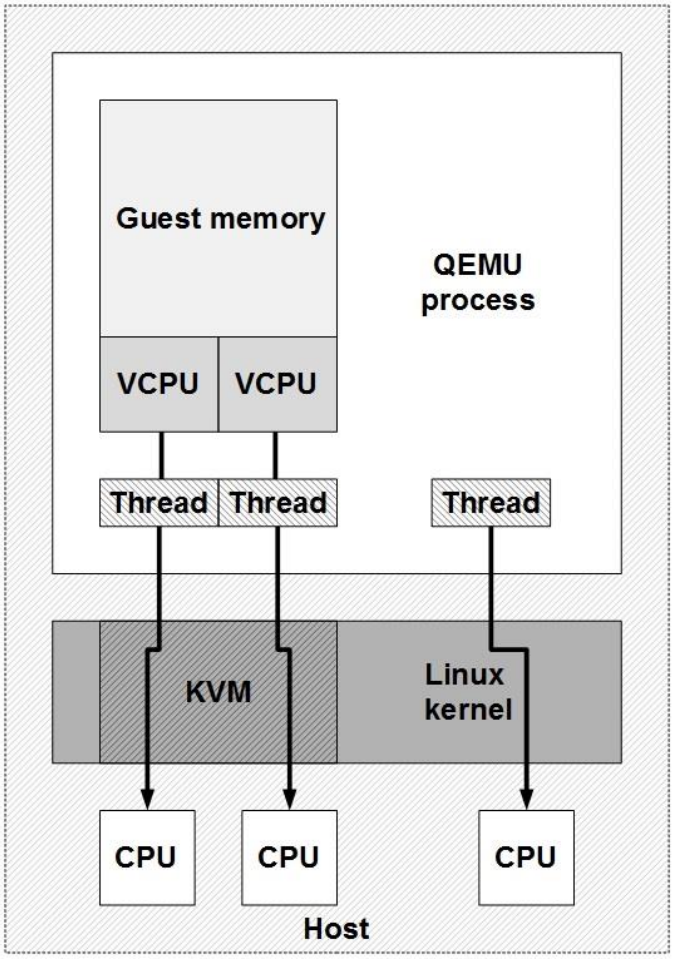


Рисунок 3.4. Особливості архітектури QEMU/KVM.

Основний цикл подій QEMU реалізований в main_loop_vait (), він виконує такі завдання [13]:

• Очікування, поки файлові дескриптори стануть доступними для читання/запису. Знаходження цієї функції в ядрі QEMU пояснюється тим, що вона відіграє центральну роль у доступі до ресурсів, оскільки це звичайні файли, каталоги, сокети, іменовані канали тощо. для будь-якої прикладної програми сутність - файлові дескриптора.

• Запускає експіровані таймери та утиліти відкладеного програмного переривання.

Коли дескриптор файлу стає доступним, закінчується термін дії таймера або запускається обробник переривання, цикл подій викликає (синхронну) функцію функції обробника подій. Кожен такий оператор підпорядковується двом простим правилам:

1. Під час виконання цієї операції заборонено виконувати будь-які інші маніпулятори. Іншими словами, немає необхідності синхронізувати обробку подій: зворотні виклики відбуваються послідовно й атомарно один з одним. У ядрі завжди є лише один потік, який керує QEMU.

2. Відсутність блокування системних викликів або інших довготривалих операцій. Порушення цього правила призводить до «завмирання» гостьової ОС через припинення виконання циклу через затримки в обробці подій.

Діаграма на рисунку 3.5. ілюструє, як виглядає процес QEMU на хост-платформі загалом.

Тепер, коли загальні принципи роботи QEMU/KVM викладені, ми можемо перейти до подальшого опису програмного-комплексу.

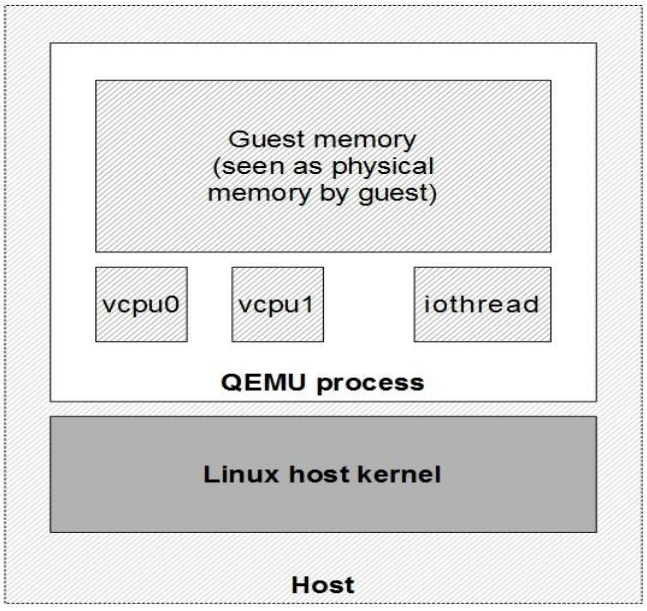


Рисунок 3.5. QEMU процес для хост-платформи у загальному вигляді

3.2 Теоретичний алгоритм роботи

Звернемо увагу, що здається нераціональним керувати віртуальними машинами безпосередньо за допомогою команд QEMU через: (а) можливість використовувати у якості бекенд-гіпервізора не тільки QEMU / KVM; б) автоматизація управління віртуальним середовищем; (в) уніфікація бізнес-логіки аналізу та моніторингу. Тому для управління віртуалізацією було вирішено використовувати універсальну бібліотеку – libvirt, оскільки вона на даний момент підтримує більшість відомих гіпервізорів і стандартизує керування ними через узагальнений API. В результаті програмний пакет більше не залежить від конкретного гіпервізора.

Отже, наш програмний інструмент буде використовувати інтерфейс libvirt для вирішення першого завдання – створення дампу фізичної пам’яті. Бібліотека, у свою чергу, керуватиме віртуальними машинами безпосередньо за допомогою команд гіпервізора.

Алгоритм розроблюваного програмного комплексу можна описати наступним чином:

1. Підготовка хост-платформу до циклу аналізу.

Виділення та резервування області пам'яті, в яку будуть (повторно) введені проаналізовані дампи пам'яті.

2. Зняття дампу пам'яті.

Зняття дампу фізичної пам’яті віртуальної машини за допомогою команди libvirt – create snapshot. Слід підкреслити, що ця дія виконується «на льоту», тобто без призупинення роботи гостьової операційної системи.

3. Запис дампу пам'яті.

Буферізований запис дампа в зарезервовану область пам'яті.

4. Патерн-аналіз дампа пам'яті.

Пошук байтових артефактів операційної системи, у дампі пам’яті використовуючи методи, описані раніше у даній роботі.

5. Аналіз результатів пошуку.

Звичайно, при вступі у нас є ряд правил, яких цей комплекс буде дотримуватися. Тому на етапі аналізу визначаються нестандартні дії та перевіряються умови, щоб отримати подальші вказівки для системи.

Слід зазначити, що всі дії виконуються неінвазивно, оскільки вся взаємодія з гостьовою ОС відбувається через гіпервізори. Тому така система буде правильною з точки зору комп’ютерного криміналістичного аналізу.

Розроблена архітектура програмного комплексу зображено на рис. 3.6.

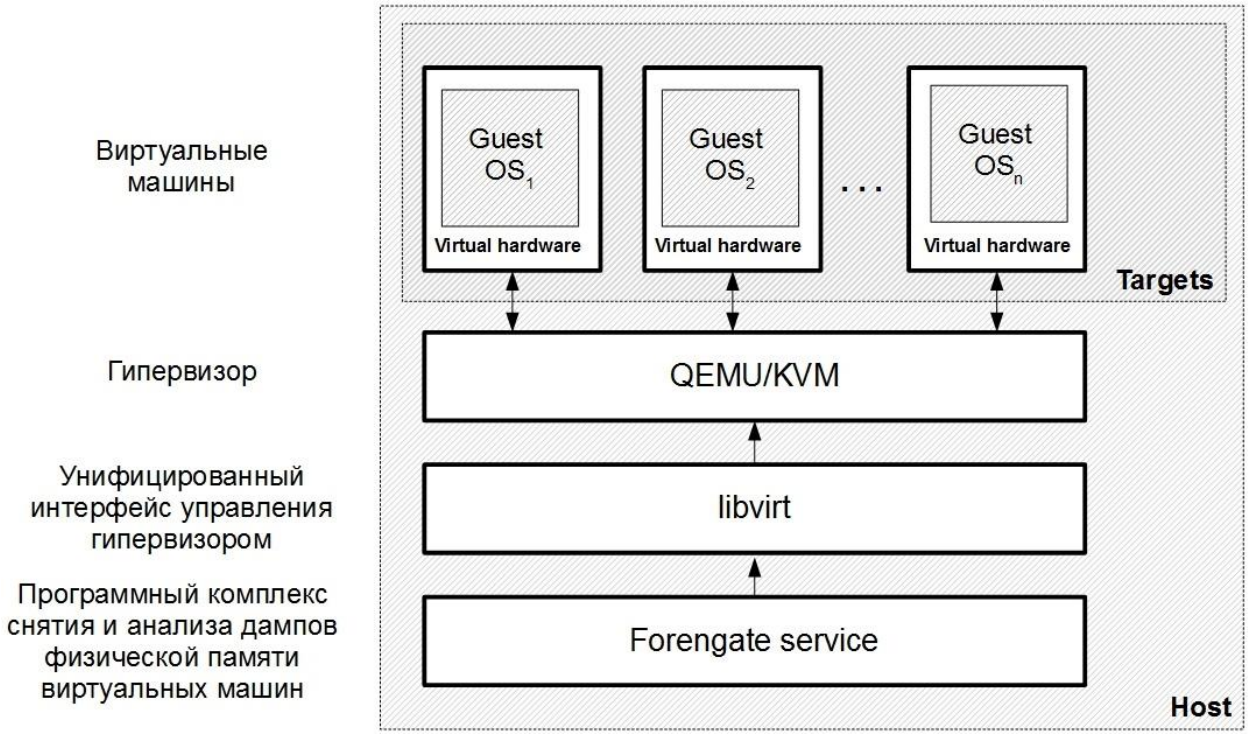


Рисунок. 3.6. Архітектура програмного комплексу першому етапі роботи

На першому етапі розробки архітектури програмного комплексу доступ до фізичної пам’яті гостьових машин QEMU/KVM здійснювався через libvirt. Звичайно, хотілося б отримати доступ до пам’яті безпосередньо з хост-платформи без використання посередників, але поки це здається недосяжним.

Головною проблемою тут, як це не парадоксально, є згадана вище гібридна архітектура QEMU. Справа в тому, що ідеальним рішенням для запису пам'яті віртуальної машини, тобто копіювання віртуальної пам'яті хост-процесу, буде її fork (з англійської fork - системний виклик, який створює новий процес, який є унікальним ідентифікатором повної копії батьківського процесу. процес). Однак, як відомо, коли використовується форк багатопотокового процесу, у процеса-потомка залишається лише один, викликавший форк потік, (що гарантується стандартом POSIX). Це означає, що новостворений процес може бути заблокований назавжди, якщо ініціюючий потік в момент форка чекав на якомусь примітиву синхронізації сигналів з інших потоків.

Однак, незважаючи на описані труднощі, у майбутньому планується більш детальний аналіз пристрою для відображення пам’яті гостьової ОС на пам’ять сервера віртуалізації більш детально, щоб знайти надійні засоби захоплення атомарних знімків пам’яті безпосередньо з хоста-платформи (див. рис. 3.7).

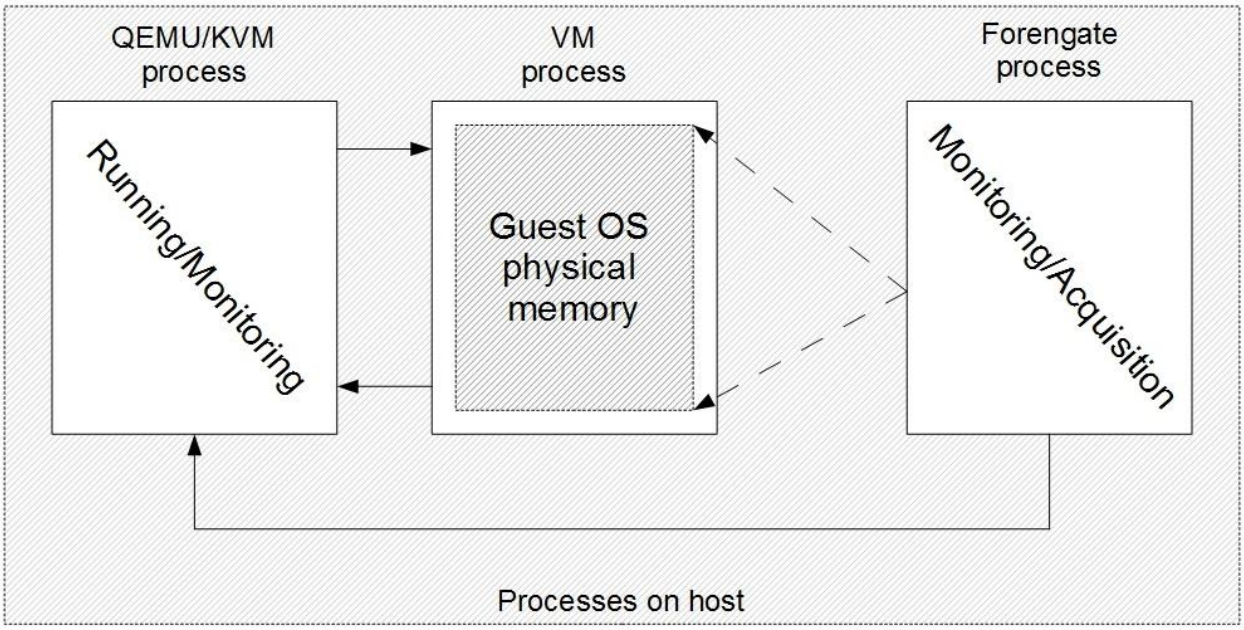


Рисунок. 3.7. Схема роботи комплексу як процесів на хост-платформі. Зв'язки пунктирними лініями планується реалізувати

3.3 Віртуальна оперативна пам’ять.

У середовищі KVM кожна віртуальна машина визначає два ключових параметри, що стосуються RAM використана політика розподілу:

• Максимальне виділення: він представляє максимальну кількість оперативної пам'яті, яку можна виділити ВМ, коли вона виконується.

• Поточний розподіл: він представляє фактичну оперативну пам'ять, виділену віртуальній машині, коли вона є увімкнено. Він може бути меншим або дорівнювати максимальному значенню виділення.

Кожна віртуальна машина працює як звичайний процес GNU/Linux всередині хост-ОС. Коли ВМ увімкнено, її процес виділяє ОЗП, визначену параметром поточного розподілу. Якщо віртуальній машині потрібно більше оперативної пам’яті, виділення збільшується до максимального значення.[16]

Така поведінка досягається за допомогою драйвера з ім’ям balloon це дозволяє кожній віртуальній машині повідомляти гіпервізор, скільки оперативної пам’яті їй потрібно.[17] Гіпервізор потім динамічно призначає доступну оперативну пам’ять між різними віртуальними машинами відповідно до їхньої потреби.[17] Незважаючи на те, що це гнучкий підхід, він може призвести до потенційної проблеми: якщо сума максимальних значень виділення всіх різних віртуальних машин дорівнює більше, ніж фізична оперативна пам’ять хоста та всіх віртуальних машин запитують максимальне виділення в той же час ОС хосту стане нестабільною. Тому рекомендується, щоб поточне значення виділення дорівнює максимальному значенню виділення та сумі цих значень для всіх віртуальних машин нижчі, ніж фізична оперативна пам’ять, доступна на хості.

3.4 Віртуальні образи оперативної пам’яті.

Утиліта virsh також може використовуватися для створення образу вмісту оперативної пам’яті віртуальної машини, але його потрібно призупинити, щоб отримати постійний результат [18].

Команда virsh suspend призупиняє виконання віртуальної машини, але зберігає її оперативну пам’ять розподіл незміненим. Після призупинення роботи віртуальної машини вміст оперативної пам’яті може бути скинутий до а локальний запуск через команду virsh dump. Наприклад, наступна команда створює файл образ вмісту RAM віртуальної машини з назвою VM1: virsh dump VM1 --memory-only/mnt/kvm/VM1.memdump . Отриманий файл – це /mnt/kvm/VM1.memdump і файл параметр --memory-only вказує на збір лише вмісту оперативної пам’яті та загального CPU

значення регістра ВМ. Після створення образу RAM можна відновити виконання ВМ за допомогою команди virsh resume. Важливо відзначити, що команда virsh dump автоматично призупиняє роботу ВМ перед створенням образу RAM і відновлює його після завершення зображення. Якщо віртуальну машину слід призупинити під час дослідження, образ RAM все ще може бути створений, якщо до файлу додано fag –live команда virsh dump. У цьому випадку образ RAM створюється під час роботи віртуальної машини.

Команда virsh dump генерує образ RAM у дампі ядра QEMU ELF формат, який підтримується платформою Volatility Framework (Volatility, 2018). Використовуючи це утиліта, можна досліджувати вміст зображення RAM, шукаючи запущені процеси, мережеві підключення, відкриті файли та інші артефакти, що зберігаються в RAM.

4 АЛГОРИТМИ ПАТТЕРН-АНАЛІЗУ

Одним із підзадач цієї роботи є аналіз шаблону дампу фізичної пам’яті, тобто пошук певного підрядка (байтового патерну) у рядку (дампі фізичної пам’яті). На сьогодні існує великий вибір алгоритмів пошуку підрядків. До них належать такі алгоритми, як алгоритм Бойера-Мура-Хорспула, автоматичний алгоритм Ахо-Корасика, алгоритм Кнутта-Морріса-Пратта тощо.

Алгоритм Бойєра-Мура-Хорспула є спрощенням алгоритму Бойєра-Мура. Процедура роботи алгоритму дуже проста [19]. На першому кроці для кожного символу створюється таблиця зміщення. Потім початок тексту та потрібний шаблон об’єднуються для порівняння. Якщо символи шаблону повністю збігаються з символами рядка, проблема вирішується, і потрібний підрядок знайдено. В іншому випадку шаблон переміщує певну кількість символів вправо. Ця кількість набирається за таким правилом: береться символ рядка над останнім символом шаблону - знак стоп. Перемістіть шаблон так, щоб той самий символ шаблону з’явився під символом зупинки. Якщо хвостовика в шаблоні немає, форма буде переміщена за межі цього хвостовика (див. таблицю 4.1).

	↓ стоп – символ
Текст	111010110
Шаблон	1011
Сдвиг шаблона	1011

Таблиця 4.1.

Це правило реалізується за допомогою таблиці зміщення, в якій кожному символу присвоюється значення, рівне різниці між довжиною шаблону та порядковим номером символу (якщо символ повторюється, входження відображається в крайньому правому куті). Алгоритм продовжується до тих пір, поки шаблон повністю не збігається з підрядком або поки не буде досягнуто кінця рядка. Цей алгоритм простий у реалізації і досить швидкий, щоб знайти один конкретний шаблон у послідовності, але його продуктивність втрачається, якщо одночасно потрібні кілька підрядків.

Алгоритм Кнута-Морріса-Пратта є одним із перших лінійних алгоритмів оцінки найгіршого випадку. Перш ніж розглянути алгоритм, необхідно визначити термін префікс функції з рядка.

Функція префікса рядка - це довжина найбільшого префікса рядка, який одночасно є його суфіксом, іншими словами, це максимальна довжина початку рядка, яка одночасно є його кінцем. Як і в класичному алгоритмі пошуку підрядків, шаблон порівнюється з рядком зліва направо. Однак особливість цього алгоритму полягає в тому, що ви можете уникнути непотрібних змін, використовуючи функцію префікса.

Нехай шаблон p [0,…, m-1] накладається на рядок із шуканими [0,…, n-1]. Розглянемо порівняння рядків у позиції i, тобто. візерунок p [0,…, m-1] з частиною ланцюжка s [i, i + m-1]. Припустимо, що існує перша невідповідність між s [i + j] і p [j], 1 <j <m, а π [j] є значенням префіксної функції з послідовності s [0,…, n-1] для індексу j. Тоді можна продовжити порівняння з місця s [i + j] і p [π [j]], виключаючи непотрібні зсуви [19].

Час роботи такого алгоритму лінійно залежить від кількості вхідних даних.

Алгоритм Ахо-Корасика вирішує задачу ефективного пошуку входження всіх вхідних шаблонів у заданій послідовності [20]. Особливість цього алгоритму полягає в тому, що він використовує структуру даних для

вирішення проблеми – бор, на якому побудований кінцевий детермінований автомат.

Бор — це дерево, кожна вершина якого означає рядок, корінь — нульовий рядок. На краях дерева є один символ, тому, проходячи через усі ребра від кореня до вершини та об'єднуючи символи краю, ми отримуємо рядок, який відповідає цій вершині. Кожній верхівці сосни відповідає лише один ряд. Інша назва грози — префіксальне дерево.

Як зазначалося вище, в цьому алгоритмі необхідно побудувати кінцевий детермінований автомат - автомат, в якому немає дуг, що не містять жодного символу, і з будь-якого стану для будь-якого символу можливий перехід саме в один стан. Стан машини – якась зморшка. Для переходу автомата з одного стану в інший використовуються наступні параметри: поточна вершина v і символ ch. Якщо можна пройти від поточної вершини по ребру з символом ch, то ми йдемо за ним, інакше йдемо за суфіксальним з'єднанням і повторюємо процедуру з нової вершини.

Суфікс вершини v є вказівником на вершину u таким чином, що рядок u є найбільшим фактичним суфіксом рядка v, або, якщо такої вершини немає в сосні, вказівник знаходиться в корені. Щоб отримати посилання з верхнім суфіксом, ви повинні перейти до його батьківського суфікса, перейти за посиланням на батьківський суфікс і розпочати перехід від поточної вершини вздовж символу у батьківському суфіксі. Отримана вершина є суфіксом вершини c.

Цей алгоритм показує хороші результати, коли потрібно шукати кілька постійних шаблонів одночасно.

5 АПРОБАЦІЯ ТА ЕКСПЕРИМЕНТ

5.1 Налаштування робочого оточення

В якості операційної системи використовувався дистрибутив Ubuntu Linux на основі Debian GNU / Linux. Зараз Ubuntu офіційно підтримує лише один гіпервізор QEMU / KVM. Щоб його ефективно використовувати, процесор повинен підтримувати технологію апаратної віртуалізації (Intel VT-k / AMD-V). Основною операційною системою є Ubuntu 14.04.2 Server AMD64 (k86-64). Вибір 64-розрядної платформи пояснюється наявністю важливих для нас функцій:

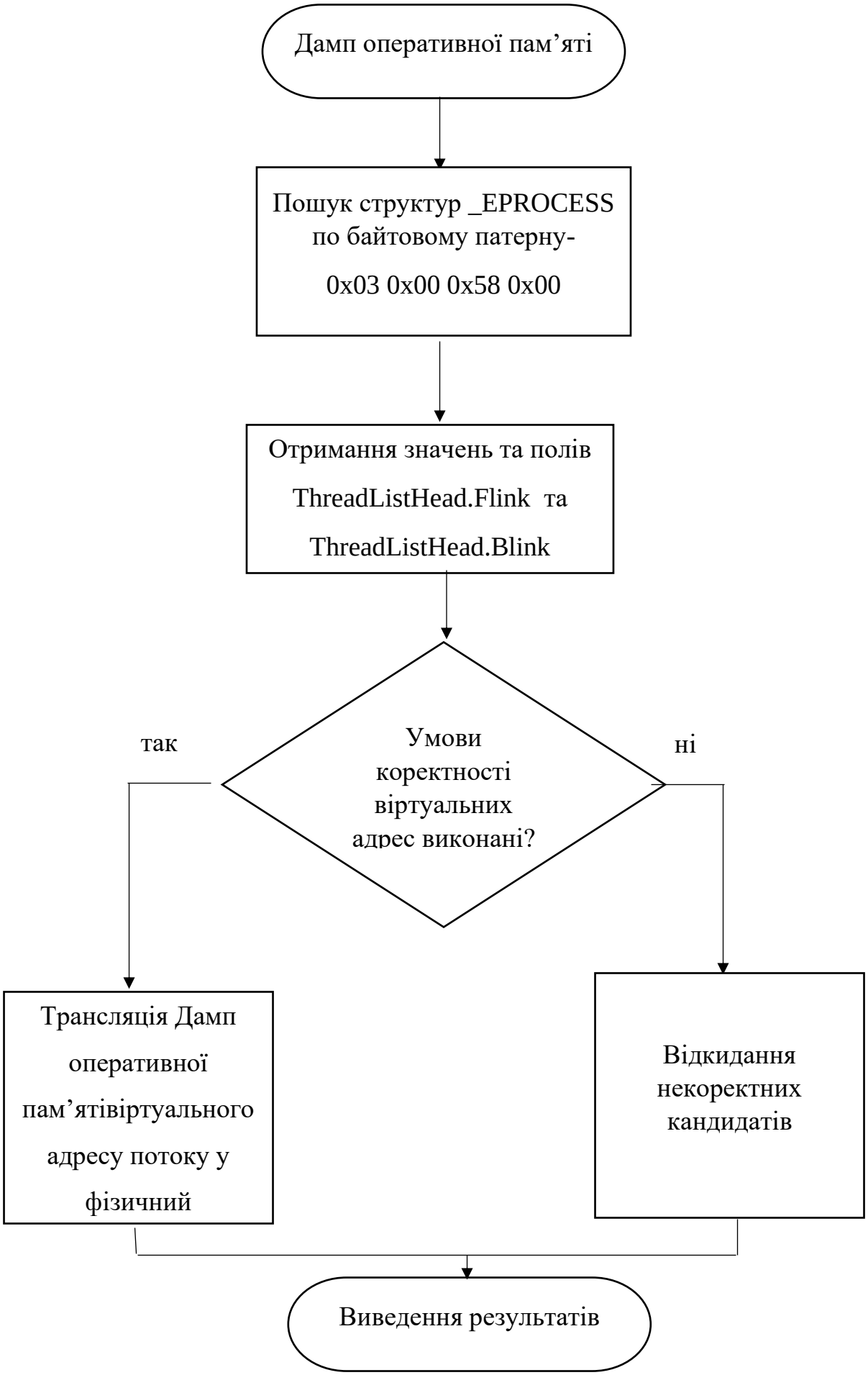
• виділення більш ніж 2 ГБ ОЗП для гостей.

• Технологія віртуалізації, яка дозволяє емулювати різноманітне обладнання, а також керувати ресурсами та ізолювати ресурси між кількома запущеними гостьовими операційними системами.

В якості гостьової системи була обрана Microsoft Windows 7 Professional k86-64, яка на даний момент є однією з найпопулярніших операційних систем.

5.2 Постановка експерименту

У контексті створення пакета судово-медичної експертизи, що забезпечує неінвазивний аналіз динаміки оперативної пам’яті, необхідно автоматизувати наступні алгоритми: алгоритми пошуку даних процесу, регістр та алгоритм трансляції віртуальної адреси. у фізичному. Схематично їх можна зобразити таким чином (див. рис. 5.1).


Дамп оперативної пам’яті
Пошук структур _EPROCESS по байтовому патерну-
0x03 0x00 0x58 0x00
Отримання значень та полів ThreadListHead.Flink та ThreadListHead.Blink
Умови коректності віртуальних адрес виконані?
так
ні
Трансляція Дамп оперативної пам’ятівіртуального адресу потоку у фізичний
Відкидання некоректних кандидатів
Виведення результатів

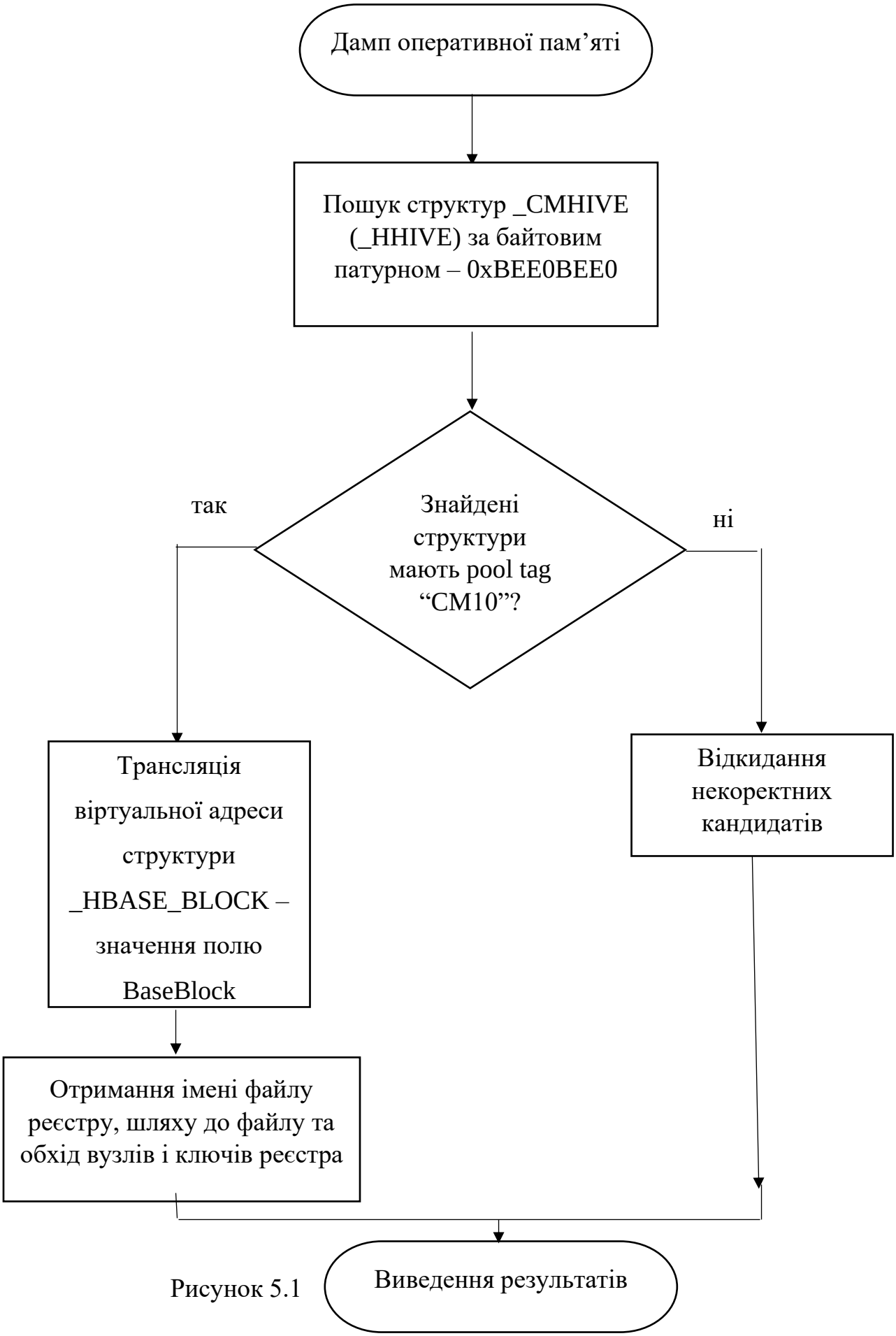

Дамп оперативної пам’яті
Пошук структур _CMHIVE (_HHIVE) за байтовим патурном – 0xBEE0BEE0
Знайдені структури мають pool tag “CM10”?
так
ні
Трансляція віртуальної адреси структури _HBASE_BLOCK – значення полю BaseBlock
Відкидання некоректних кандидатів
Отримання імені файлу реєстру, шляху до файлу та обхід вузлів і ключів реєстра
Виведення результатів

Рисунок 5.1

5.3 Результати експерименту

Для розробки інструменту була обрана мова програмування Go. На першому кроці гіпервізор QEMU/KVM отримав фізичну пам’ять віртуальної машини, яка нас цікавить. Оскільки завдання пошуку шаблону байтів у дампі фізичної пам’яті зводиться до завдання пошуку підрядка в рядку, алгоритм Бойєра-Мура-Хорспула реалізовано як найшвидший із представлених алгоритмів для вирішення цієї проблеми. Використовуючи цей алгоритм, були знайдені всі можливі кандидати на процедурні структури, а потім були відхилені неправильні кандидати шляхом встановлення вищевказаних умов:

```
startKernelAddr := []byte{0xFF, 0xFF, 0x80, 0x0, 0x0, 0x0, 0x0, 0x0}
for i := 0; i < len(thrListHeadF); i++ {
    if bytes.Compare(thrListHeadF[i], startKernelAddr) >= 0 &&
        bytes.Compare(thrListHeadB[i], startKernelAddr) >= 0 &&
        len(byteArrToBinStr(dtb[i])) <= 40 {
        kernEprOffsets = append(kernEprOffsets, eprOffsets[i])
    }
}
```

Для знайдених процесів знайдено значення полів DirectoryTableBase, ThreadListHead.Flink і ThreadListHead.Blink. Отримані віртуальні адреси потоків процесу транслюються у фізичні для відновлення віртуального адресного простору процесу.

На додаток до процесу, фізичні та віртуальні адреси вуликів були також знайдені шляхом пошуку байтів, описаних в алгоритмі, та перевірки умов:

```
func filterHhiveOffsets(hhiveOffsets []int64, fp string) (trueHhiveOffsets []int64)
{
    for i := 0; i < len(hhiveOffsets); i++ {
        poolTag, _ := findStruct(hhiveOffsets[i]-12, fp, 4, false)
        if string(poolTag) == "CM10" {
            trueHhiveOffsets = append(trueHhiveOffsets, hhiveOffsets[i])
        }
    }
    return
}
```

Видаляючи ці адреси, фізичні та віртуальні адреси головного блоку кожного вулика, отримуються імена файлів реєстру.

При побудові індексу комірки було виявлено, що віртуальна адреса, збережена в полі BlockAddress, не вказує на початок структури _HBIN. У зв’язку з тим,

що інформація, пов’язана з реєстром Windows, зберігається в приватному доступі, визначити причину та алгоритм вирішення цієї проблеми неможливо. Проте емпірично було виявлено, що якщо до цієї адреси додати зміщення 0x04, ми зможемо знайти дані, які нас цікавлять, за отриманою адресою.

Під час експерименту було виявлено, що в гостьовій операційній системі, чий дамп фізичної пам’яті було перевірено, запущені такі процеси (див. таблицю 5.1).

Физический адрес процесса	**Имя процесса**	**Физический адрес процесса**	**Имя процесса**
28551C0	Idle	3E6C1B30	winlogon.exe
3E0EDB30	SearchProtocol	3E6C9060	taskhost.exe
3E207430	svchost.exe	3E71EB30	services.exe
3E2435F0	svchost.exe	3E7EB340	svchost.exe
3E287060	svchost.exe	3E7F6B30	lsass.exe
3E2D76B0	mscorsvw.exe	3E7FCB30	lsm.exe
3E2E34B0	dwm.exe	3EEBB650	smss.exe
3E2F7310	explorer.exe	3F7C7060	sppsvc.exe
3E4BD9E0	svchost.exe	3FAB8630	SearchProtocol
3E4FA920	svchost.exe	3FAE2B30	SearchFilterHo
3E5C7B30	svchost.exe	3FC28060	mscorsvw.exe
3E62D060	spoolsv.exe	3FC76B30	wmpnetwk.exe

Физический адрес процесса	Имя процесса	Физический адрес процесса	Имя процесса
3E637920	SearchIndexer.	3FC8D6A0	svchost.exe
3E68DB30	csrss.exe	3FD63B30	calc.exe
3E6A5B30	csrss.exe	3FE36060	WmiPrvSE.exe
3E6ABB30	wininit.exe	3FF05360	svchost.exe
3E6BD350	svchost.exe	3FFF79E0	System

Таблиця 5.1. Перелік знайдених процесів

Отримані дані перевіряються та підтверджуються за допомогою спеціальних засобів налагодження для Windows.

На прикладі потоку системного процесу адреса віртуального потоку транслюється у фізичний. Були отримані наступні результати:

• віртуальна адреса системи процесу: FFFFFA8000CC0460;

• фізична адреса, що відповідає цій віртуальній адресі: 3FF3F460;

• значення за цією фізичною адресою: FFFFFA8000CCB460.

Ви можете використовувати ці дані для аналізу системи на основі того, які програми запущені та яку інформацію вони використовують. Також варто звернути увагу на системні процеси:

• Idle і System – процес ядра ОС Windows, який є одним потоком (або кількома потоками в багатоядерних системах), який виконується, коли процесор не виконує інші потоки. Ці процеси не мають відповідного виконуваного файлу на диску;

• services.exe - керує службами Windows. Цей процес має бути батьківським процесом svchost.exe. Система повинна запустити один екземпляр services.exe з каталогу system32;

• svchost.exe є основним процесом для служб, завантажених із динамічних бібліотек. Деякі комп'ютерні віруси маскуються під svchost.eke, розміщуючи виконуваний файл у несистемному system3232. Цей процес виконується лише як SYSTEM, LOCAL SERVICE або NETWORK SERVICE і не може бути запущений від імені користувача;

• lsass.exe відповідає за дотримання політики безпеки, перевірку паролів та створення маркерів доступу. Шкідливі програми часто ховаються під іменем, подібним до назви процесу, замінюючи першу нижню букву «L» у своїй назві на велику «I», яка виглядає схожою;

• winlogon.exe – це компонент операційної системи Microsoft Windows, який відповідає за вхід в систему. Як і більшість інших системних процесів, його виконуваний файл знаходиться в системному system32;

• explorer.exe - також відомий як Windows Explorer, відповідає за взаємодію з користувачем: графічний інтерфейс, папки, навігацію. Цей процес також має доступ до конфіденційних матеріалів – публічних документів, рахунків-фактур. Процес починається в Провіднику Windows і дозволяє використовувати файлову систему комп'ютера;

• smss.exe - представляє підсистему диспетчера сеансів. Ця підсистема відповідає за запуск сеансу користувача. Цей процес ініціалізує системний потік і відповідає за різні дії, включаючи запуск процесів Winlogon і Win32 (Csrss.eke) і налаштування системних змінних. Файл smss.exe знаходиться в каталозі C:\windows \ System32. Якщо цей файл знаходиться в будь-якому іншому каталозі, слід звернути особливу увагу на цей об'єкт, оскільки є велика ймовірність, що він є шкідливим;

• csrss.exe — це підсистема підсистеми виконання клієнта/сервера, яка відіграє важливу роль у створенні та видаленні процесів і потоків, підтримуючи власний список об’єктів, які можна використовувати для перехресних посилань з іншими джерелами даних. Кілька процесів csrss.exe запускаються, коли кожен сеанс створює окрему копію. Особливу увагу при аналізі слід звернути на назву процесу, оскільки багато шкідливих програм використовують імена легітимних процесів, щоб приховати або трохи змінити їх присутність на комп’ютері, наприклад csrsss.exe.

Також можна було скласти список файлів реєстру за реалізованим алгоритмом (див. таблицю 5.2), тобто ім’я файлу реєстратора, фізичну адресу вулика та можна було перекласти індекс комірки. кожного вулика з подальшою метою обходу вузлів реєстру.

Таблиця 5.2 Перелік знайдених файлів реєстру

Имя файла реестра	**Физический адрес**
Volume Information\Syscache.hve	9C98010
\Microsoft\Windows\UsrClass.dat	13CD9010
??\C:\Users\ * \ntuser.dat	16908010
files\NetworkService\NTUSER.DAT	1A84D410
rofiles\LocalService\NTUSER.DAT	1A9A8010
\SystemRoot\System32\Config\SAM	1C813010
emRoot\System32\Config\SECURITY	1FC5E010
temRoot\System32\Config\DEFAULT	20543010
emRoot\System32\Config\SOFTWARE	21620010

Имя файла реестра	Физический адрес
Device\HarddiskVolume1\Boot\BCD	22018010
NO NAME	274BC010
NO NAME	27534010
System	275BF010

ВИСНОВОК

У даній роботі обговорюються та класифікуються підходи та інструменти аналізу оперативної пам’яті. Також було досліджено внутрішню структуру операційних систем сімейства Windows NT та методи аналізу дампів їх оперативної пам'яті.

Була розроблена архітектура програмного комплексу, інтегрована з платформами віртуалізації, яка дозволяє виробляти криміналістично правильний неінвазивний аналіз оперативної пам’яті гостьових операційних систем.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рудий Т. В., Захарова О. В., Кулешник Я. Ф., Сеник В. В., Бичинюк І. В. Протидія комп'ютерним злочинам: посібник. Львівський державний університет внутрішніх справ. Львів, 2016. 176 с

2. Michael H. L., Andrew C., Jamie L., AAron W. The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory. Indianapolis: John Wiley & Sons, Inc., 2014. 886 p.

3. Digambar P, Bhadran V.K., Forensic Data Carving, Springer Berlin Heidelberg, 2011. pp 137-148.

4. Огляд методів віртуалізації, архітектур та реалізацій. https://www.ibm.com/developerworks/uk/library/l-linuxvirt/index.html

5. Шеховцов В. А. Операційні системи / В. А. Шеховцов – К.: Вид. гр. BHV, 2005. – 576 с.

6. MS Windows NT Kernel-mode User and GDI White Paper. https://technet.microsoft.com/library/cc750820.aspx

7. Антонов О. С. Системне програмне забезпечення

8. Russinovich M., Solomon D., Ionescu A. Windows Internals, Part 1, Sixth Edition. Redmond, Washington: Microsoft Press, 2012. 726 p.

9. Russinovich M., Solomon D., Ionescu A. Windows Internals, Part 2, Sixth Edition. Redmond, Washington: Microsoft Press, 2012. 645 p.

10. Windows registry information for advanced users.

windows-registry-advanced-users.htm

11. Lijuan. X, Lianhai. W, Shuhui. Z, Hengjian. L. A Method to Analyze Memory Images of 64-bit Windows 8, pp 304-312.

12. QEMU Internals: Big picture overview. http://blog.vmsplice.net/2011/03/qemu-internals-big-picture-overview.html

13. QEMU Internals: Overall architecture and threading model. http://blog.vmsplice.net/2011/03/qemu-internals-overall-architecture-and.html

14 Федотов Н. Н. Форензика – компьютерная криминалистика. М.: Юридический мир, 2007. 360 с.

15. Головань С. М., Петров О. С., Хорошко В. О., Чирков Д. В. Нормативне забезпечення інформаційної безпеки: підручник / за ред. проф. В. О. Хорошка. К.: ДУІКТ, 2008. 533 с

16. Chirammal, H. D., Mukhedkar, P., & Vettathu, A. (2016). Mastering KVM virtualization. Packt Publishing.

17. RedHat. (2017). Red hat enterprise linux 7: Virtualization getting started guide. Retrieved from https://access.redhat.com/documentation/.

18. Suneja, S., Isci, C., & de Lara, E. (2015, March). Exploring VM introspection: Techniques and trade-offs. ACM Sigplan Notices, 50(7), 133-146.

19. Вирт Н. Алгоритми та структури даних. Нова версія для Оберона

. М.: ДМК Пресс, 2010. 272 с.

20. Алгоритм Ахо-Корасик. https://xlinux.nist.gov/dads/HTML/ahoCorasick.html

Додаток А

Зауваження нормоконтролера

Таблиця А.1 – Зауваження нормоконтролера

Сторінка	Позначення	Зміст зауваження
ПЗ		Посилання на джерела не у порядку їх використання
ПЗ		Нещільне заповнення сторінок ПЗ
С. 7, 9-11, 21-22, 27-30, 32, 34, 45		Невірний перелік
ПЗ		Назва рисунку та таблиці – риски немає, таблиця 4.1, 5.1 та 5.2 - підпис. Рисунок 5.1
С. 27, 43		Фрагменти коду – рисунок
С. 50-51		Помилки в оформленні списку джерел

Додаток Б

Пакети, встановлені на вузлі віртуалізації

Пакети, встановлені на вузлі віртуалізації, та їх відповідні версії.

Package name	Version
accountsservice	0.6.40-2ubuntu11.3
acl	2.2.52-3
acpid	1:2.0.26-1ubuntu2
adduser	3.113+nmu3ubuntu4
apparmor	2.10.95-0ubuntu2.8
apport	2.20.1-0ubuntu2.15
apport-symptoms	0.20
apt	1.2.25
apt-transport-https	1.2.25
apt-utils	1.2.25
at	3.1.18-2ubuntu1
augeas-lenses	1.4.0-0ubuntu1.1
base-files	9.4ubuntu4.6
base-passwd	3.5.39
bash	4.3-14ubuntu1.2
bash-completion	1:2.1-4.2ubuntu1.1
bcache-tools	1.0.8-2
bind9-host	1:9.10.3.dfsg.P4-8ubuntu1.10
binutils	2.26.1-1ubuntu1~16.04.6
bridge-utils	1.5-9ubuntu1
bsdmainutils	9.0.6ubuntu3
bsdutils	1:2.27.1-6ubuntu3.4
btrfs-tools	4.4-1ubuntu1
busybox-initramfs	1:1.22.0-15ubuntu1
busybox-static	1:1.22.0-15ubuntu1
byobu	5.106-0ubuntu1
bzip2	1.0.6-8
ca-certificates	20170717~16.04.1
cgmanager	0.39-2ubuntu5
cloud-guest-utils	0.27-0ubuntu25
cloud-initramfs-copymods	0.27ubuntu1.5
cloud-initramfs-dyn-netconf	0.27ubuntu1.5
command-not-found	0.3ubuntu16.04.2
command-not-found-data	0.3ubuntu16.04.2
console-setup	1.108ubuntu15.3
console-setup-linux	1.108ubuntu15.3

coreutils	8.25-2ubuntu3~16.04
cpio	2.11+dfsg-5ubuntu1
cpu-checker	0.7-0ubuntu7
crda	3.13-1
cron	3.0pl1-128ubuntu2
cryptsetup	2:1.6.6-5ubuntu2.1
cryptsetup-bin	2:1.6.6-5ubuntu2.1
curl	7.47.0-1ubuntu2.6
dash	0.5.8-2.1ubuntu2
dbus	1.10.6-1ubuntu3.3
debconf	1.5.58ubuntu1
debconf-i18n	1.5.58ubuntu1
debianutils	4.7
dh-python	2.20151103ubuntu1.1
diffutils	1:3.3-3
distro-info-data	0.28ubuntu0.7
dmeventd	2:1.02.110-1ubuntu10
dmidecode	3.0-2ubuntu0.1
dmsetup	2:1.02.110-1ubuntu10
dns-root-data	2015052300+h+1
dnsmasq-base	2.75-1ubuntu0.16.04.4
dnsutils	1:9.10.3.dfsg.P4-8ubuntu1.10
dosfstools	3.0.28-2ubuntu0.1
dpkg	1.18.4ubuntu1.3
e2fslibs:amd64	1.42.13-1ubuntu1
e2fsprogs	1.42.13-1ubuntu1
ebtables	2.0.10.4-3.4ubuntu2
ed	1.10-2
efibootmgr	0.12-4

eject	2.1.5+deb1+cvs20081104-13.1ubuntu0.16.04.1
ethtool	1:4.5-1
exfat-fuse	1.2.3-1
exfat-utils	1.2.3-1
file	1:5.25-2ubuntu1
findutils	4.6.0+git+20160126-2
fonts-ubuntu-font-family-console	1:0.83-0ubuntu2
friendly-recovery	0.2.31ubuntu1
ftp	0.17-33
fuse	2.9.4-1ubuntu3.1
gawk	1:4.1.3+dfsg-0.1
gcc-5-base:amd64	5.4.0-6ubuntu1~16.04.9
gcc-6-base:amd64	6.0.1-0ubuntu1
gdisk	1.0.1-1build1
geoip-database	20160408-1
gettext-base	0.19.7-2ubuntu3

gir1.2-glib-2.0:amd64	1.46.0-3ubuntu1
git	1:2.7.4-0ubuntu1.3
git-man	1:2.7.4-0ubuntu1.3
gnupg	1.4.20-1ubuntu3.1
gpgv	1.4.20-1ubuntu3.1
grep	2.25-1~16.04.1
groff-base	1.22.3-7
grub-common	2.02~beta2-36ubuntu3.17
grub-efi-amd64	2.02~beta2-36ubuntu3.17
grub-efi-amd64-bin	2.02~beta2-36ubuntu3.17
grub-efi-amd64-signed	1.66.17+2.02~beta2-36ubuntu3.17
grub-legacy-ec2	17.2-35-gf576b2a2-0ubuntu1~16.04.2
grub2-common	2.02~beta2-36ubuntu3.17
gzip	1.6-4ubuntu1
hdparm	9.48+ds-1
hostname	3.16ubuntu2
ifenslave	2.7ubuntu1
ifupdown	0.8.10ubuntu1.2
info	6.1.0.dfsg.1-5
init	1.29ubuntu4
init-system-helpers	1.29ubuntu4
initramfs-tools	0.122ubuntu8.10
initramfs-tools-bin	0.122ubuntu8.10
initramfs-tools-core	0.122ubuntu8.10
initscripts	2.88dsf-59.3ubuntu2
insserv	1.14.0-5ubuntu3
install-info	6.1.0.dfsg.1-5
installation-report	2.60ubuntu1
iproute2	4.3.0-1ubuntu3.16.04.3

iptables	1.6.0-2ubuntu3
iputils-ping	3:20121221-5ubuntu2
iputils-tracepath	3:20121221-5ubuntu2
ipxe-qemu	1.0.0+git-20150424.a25a16d-1ubuntu1.2
irqbalance	1.1.0-2ubuntu1
isc-dhcp-client	4.3.3-5ubuntu12.9
isc-dhcp-common	4.3.3-5ubuntu12.9
iso-codes	3.65-1
iw	3.17-1
kbd	1.15.5-1ubuntu5
keyboard-configuration	1.108ubuntu15.3
klibc-utils	2.0.4-8ubuntu1.16.04.4
kmod	22-1ubuntu5
kpartx	0.5.0+git1.656f8865-5ubuntu2.5
krb5-locales	1.13.2+dfsg-5ubuntu2
language-selector-common	0.165.4

laptop-detect	0.13.7ubuntu2
less	481-2.1ubuntu0.2
libaccountsservice0:amd64	0.6.40-2ubuntu11.3
libacl1:amd64	2.2.52-3
libaio1:amd64	0.3.110-2
libapparmor-perl	2.10.95-0ubuntu2.8
libapparmor1:amd64	2.10.95-0ubuntu2.8
libapt-inst2.0:amd64	1.2.25
libapt-pkg5.0:amd64	1.2.25
libasn1-8-heimdal:amd64	1.7~git20150920+dfsg-4ubuntu1.16.04.1
libasound2:amd64	1.1.0-0ubuntu1
libasound2-data	1.1.0-0ubuntu1
libasprintf0v5:amd64	0.19.7-2ubuntu3
libasyncns0:amd64	0.8-5build1
libatm1:amd64	1:2.5.1-1.5
libattr1:amd64	1:2.4.47-2
libaudit-common	1:2.4.5-1ubuntu2.1
libaudit1:amd64	1:2.4.5-1ubuntu2.1
libaugeas0	1.4.0-0ubuntu1.1
libavahi-client3:amd64	0.6.32~rc+dfsg-1ubuntu2
libavahi-common-data:amd64	0.6.32~rc+dfsg-1ubuntu2
libavahi-common3:amd64	0.6.32~rc+dfsg-1ubuntu2
libbind9-140:amd64	1:9.10.3.dfsg.P4-8ubuntu1.10
libblkid1:amd64	2.27.1-6ubuntu3.4
libbluetooth3:amd64	5.37-0ubuntu5.1
libboost-iostreams1.58.0:amd64	1.58.0+dfsg-5ubuntu3.1
libboost-random1.58.0:amd64	1.58.0+dfsg-5ubuntu3.1
libboost-system1.58.0:amd64	1.58.0+dfsg-5ubuntu3.1
libboost-thread1.58.0:amd64	1.58.0+dfsg-5ubuntu3.1

libbrlapi0.6:amd64	5.3.1-2ubuntu2.1
libbsd0:amd64	0.8.2-1
libbz2-1.0:amd64	1.0.6-8
libc-bin	2.23-0ubuntu10
libc6:amd64	2.23-0ubuntu10
libcaca0:amd64	0.99.beta19-2build2~gcc5.2
libcacard0:amd64	1:2.5.0-2
libcap-ng0:amd64	0.7.7-1
libcap2:amd64	1:2.24-12
libcap2-bin	1:2.24-12
libcgmanager0:amd64	0.39-2ubuntu5
libcomerr2:amd64	1.42.13-1ubuntu1
libcryptsetup4:amd64	2:1.6.6-5ubuntu2.1
libcurl3-gnutls:amd64	7.47.0-1ubuntu2.6
libdb5.3:amd64	5.3.28-11ubuntu0.1
libdbus-1-3:amd64	1.10.6-1ubuntu3.3

libdbus-glib-1-2:amd64	0.106-1
libdebconfclient0:amd64	0.198ubuntu1
libdevmapper-event1.02.1:amd64	2:1.02.110-1ubuntu10
libdevmapper1.02.1:amd64	2:1.02.110-1ubuntu10
libdistorm3-3	3.3.0-3
libdns-export162	1:9.10.3.dfsg.P4-8ubuntu1.10
libdns162:amd64	1:9.10.3.dfsg.P4-8ubuntu1.10
libdrm-common	2.4.83-1~16.04.1
libdrm2:amd64	2.4.83-1~16.04.1
libdumbnet1:amd64	1.12-7
libedit2:amd64	3.1-20150325-1ubuntu2
libefivar0:amd64	0.23-2
libelf1:amd64	0.165-3ubuntu1
liberror-perl	0.17-1.2
libestr0	0.1.10-1
libevent-2.0-5:amd64	2.0.21-stable-2ubuntu0.16.04.1
libexpat1:amd64	2.1.0-7ubuntu0.16.04.3
libfdisk1:amd64	2.27.1-6ubuntu3.4
libfdt1:amd64	1.4.0+dfsg-2
libffi6:amd64	3.2.1-4
libflac8:amd64	1.3.1-4
libfreetype6:amd64	2.6.1-0.1ubuntu2.3
libfribidi0:amd64	0.19.7-1
libfuse2:amd64	2.9.4-1ubuntu3.1
libgcc1:amd64	1:6.0.1-0ubuntu1
libgcrypt20:amd64	1.6.5-2ubuntu0.3
libgdbm3:amd64	1.8.3-13.1
libgeoip1:amd64	1.6.9-1
libgirepository-1.0-1:amd64	1.46.0-3ubuntu1

libglib2.0-0:amd64	2.48.2-0ubuntu1
libglib2.0-data	2.48.2-0ubuntu1
libgmp10:amd64	2:6.1.0+dfsg-2
libgnutls-openssl27:amd64	3.4.10-4ubuntu1.4
libgnutls30:amd64	3.4.10-4ubuntu1.4
libgpg-error0:amd64	1.21-2ubuntu1
libgpm2:amd64	1.20.4-6.1
libgssapi-krb5-2:amd64	1.13.2+dfsg-5ubuntu2
libgssapi3-heimdal:amd64	1.7~git20150920+dfsg-4ubuntu1.16.04.1
libhcrypto4-heimdal:amd64	1.7~git20150920+dfsg-4ubuntu1.16.04.1
libheimbase1-heimdal:amd64	1.7~git20150920+dfsg-4ubuntu1.16.04.1
libheimntlm0-heimdal:amd64	1.7~git20150920+dfsg-4ubuntu1.16.04.1
libhogweed4:amd64	3.2-1ubuntu0.16.04.1
libhx509-5-heimdal:amd64	1.7~git20150920+dfsg-4ubuntu1.16.04.1
libicu55:amd64	55.1-7ubuntu0.3
libidn11:amd64	1.32-3ubuntu1.2

libisc-export160	1:9.10.3.dfsg.P4-8ubuntu1.10
libisc160:amd64	1:9.10.3.dfsg.P4-8ubuntu1.10
libiscc140:amd64	1:9.10.3.dfsg.P4-8ubuntu1.10
libisccfg140:amd64	1:9.10.3.dfsg.P4-8ubuntu1.10
libiscsi2:amd64	1.12.0-2
libjpeg-turbo8:amd64	1.4.2-0ubuntu3
libjpeg8:amd64	8c-2ubuntu8
libjson-c2:amd64	0.11-4ubuntu2
libk5crypto3:amd64	1.13.2+dfsg-5ubuntu2
libkeyutils1:amd64	1.5.9-8ubuntu1
libklibc	2.0.4-8ubuntu1.16.04.4
libkmod2:amd64	22-1ubuntu5
libkrb5-26-heimdal:amd64	1.7~git20150920+dfsg-4ubuntu1.16.04.1
libkrb5-3:amd64	1.13.2+dfsg-5ubuntu2
libkrb5support0:amd64	1.13.2+dfsg-5ubuntu2
libldap-2.4-2:amd64	2.4.42+dfsg-2ubuntu3.2
liblocale-gettext-perl	1.07-1build1
liblvm2app2.2:amd64	2.02.133-1ubuntu10
liblvm2cmd2.02:amd64	2.02.133-1ubuntu10
liblwres141:amd64	1:9.10.3.dfsg.P4-8ubuntu1.10
liblxc1	2.0.8-0ubuntu1~16.04.2
liblz4-1:amd64	0.0~r131-2ubuntu2
liblzma5:amd64	5.1.1alpha+20120614-2ubuntu2
liblzo2-2:amd64	2.08-1.2
libmagic1:amd64	1:5.25-2ubuntu1
libmnl0:amd64	1.0.3-5
libmount1:amd64	2.27.1-6ubuntu3.4
libmpdec2:amd64	2.4.2-1
libmpfr4:amd64	3.1.4-1

libmspack0:amd64	0.5-1ubuntu0.16.04.1
libncurses5:amd64	6.0+20160213-1ubuntu1
libncursesw5:amd64	6.0+20160213-1ubuntu1
libnetcf1:amd64	1:0.2.8-1ubuntu1
libnetfilter-conntrack3:amd64	1.0.5-1
libnettle6:amd64	3.2-1ubuntu0.16.04.1
libnewt0.52:amd64	0.52.18-1ubuntu2
libnfnetlink0:amd64	1.0.1-3
libnih-dbus1:amd64	1.0.3-4.3ubuntu1
libnih1:amd64	1.0.3-4.3ubuntu1
libnl-3-200:amd64	3.2.27-1ubuntu0.16.04.1
libnl-genl-3-200:amd64	3.2.27-1ubuntu0.16.04.1
libnl-route-3-200:amd64	3.2.27-1ubuntu0.16.04.1
libnspr4:amd64	2:4.13.1-0ubuntu0.16.04.1
libnss3:amd64	2:3.28.4-0ubuntu0.16.04.3
libnss3-nssdb	2:3.28.4-0ubuntu0.16.04.3

libnuma1:amd64	2.0.11-1ubuntu1.1
libogg0:amd64	1.3.2-1
libopts25:amd64	1:5.18.7-3
libopus0:amd64	1.1.2-1ubuntu1
libp11-kit0:amd64	0.23.2-5~ubuntu16.04.1
libpam-modules:amd64	1.1.8-3.2ubuntu2
libpam-modules-bin	1.1.8-3.2ubuntu2
libpam-runtime	1.1.8-3.2ubuntu2
libpam-systemd:amd64	229-4ubuntu21.1
libpam0g:amd64	1.1.8-3.2ubuntu2
libparted2:amd64	3.2-15ubuntu0.1
libpcap0.8:amd64	1.7.4-2
libpci3:amd64	1:3.3.1-1.1ubuntu1.1
libpciaccess0:amd64	0.13.4-1
libpcre3:amd64	2:8.38-3.1
libperl5.22:amd64	5.22.1-9ubuntu0.2
libpipeline1:amd64	1.4.1-2
libpixman-1-0:amd64	0.33.6-1
libplymouth4:amd64	0.9.2-3ubuntu13.2
libpng12-0:amd64	1.2.54-1ubuntu1
libpolkit-agent-1-0:amd64	0.105-14.1
libpolkit-backend-1-0:amd64	0.105-14.1
libpolkit-gobject-1-0:amd64	0.105-14.1
libpopt0:amd64	1.16-10
libprocps4:amd64	2:3.3.10-4ubuntu2.3
libpulse0:amd64	1:8.0-0ubuntu3.8
libpython-stdlib:amd64	2.7.12-1~16.04
libpython2.7-minimal:amd64	2.7.12-1ubuntu0~16.04.3
libpython2.7-stdlib:amd64	2.7.12-1ubuntu0~16.04.3

libpython3-stdlib:amd64	3.5.1-3
libpython3.5:amd64	3.5.2-2ubuntu0~16.04.4
libpython3.5-minimal:amd64	3.5.2-2ubuntu0~16.04.4
libpython3.5-stdlib:amd64	3.5.2-2ubuntu0~16.04.4
librados2	10.2.9-0ubuntu0.16.04.1
librbd1	10.2.9-0ubuntu0.16.04.1
libreadline5:amd64	5.2+dfsg-3build1
libreadline6:amd64	6.3-8ubuntu2
libroken18-heimdal:amd64	1.7~git20150920+dfsg-4ubuntu1.16.04.1
librtmp1:amd64	2.4+20151223.gitfa8646d-1ubuntu0.1
libsasl2-2:amd64	2.1.26.dfsg1-14build1
libsasl2-modules:amd64	2.1.26.dfsg1-14build1
libsasl2-modules-db:amd64	2.1.26.dfsg1-14build1
libsdl1.2debian:amd64	1.2.15+dfsg1-3
libseccomp2:amd64	2.3.1-2.1ubuntu2~16.04.1
libselinux1:amd64	2.4-3build2

libsemanage-common	2.3-1build3
libsemanage1:amd64	2.3-1build3
libsepol1:amd64	2.4-2
libsigsegv2:amd64	2.10-4
libslang2:amd64	2.3.0-2ubuntu1
libsmartcols1:amd64	2.27.1-6ubuntu3.4
libsndfile1:amd64	1.0.25-10ubuntu0.16.04.1
libspice-server1:amd64	0.12.6-4ubuntu0.3
libsqlite3-0:amd64	3.11.0-1ubuntu1
libss2:amd64	1.42.13-1ubuntu1
libssh2-1:amd64	1.5.0-2ubuntu0.1
libssl1.0.0:amd64	1.0.2g-1ubuntu4.10
libstdc++6:amd64	5.4.0-6ubuntu1~16.04.9
libsystemd0:amd64	229-4ubuntu21.1
libtasn1-6:amd64	4.7-3ubuntu0.16.04.3
libtext-charwidth-perl	0.04-7build5
libtext-iconv-perl	1.7-5build4
libtext-wrapi18n-perl	0.06-7.1
libtinfo5:amd64	6.0+20160213-1ubuntu1
libudev1:amd64	229-4ubuntu21.1
libusb-0.1-4:amd64	2:0.1.12-28
libusb-1.0-0:amd64	2:1.0.20-1
libusbredirparser1:amd64	0.7.1-1
libustr-1.0-1:amd64	1.0.4-5
libutempter0:amd64	1.1.6-3
libuuid1:amd64	2.27.1-6ubuntu3.4
libvirt-bin	1.3.1-1ubuntu10.19
libvirt0:amd64	1.3.1-1ubuntu10.19
libvorbis0a:amd64	1.3.5-3ubuntu0.1

libvorbisenc2:amd64	1.3.5-3ubuntu0.1
libwind0-heimdal:amd64	1.7~git20150920+dfsg-4ubuntu1.16.04.1
libwrap0:amd64	7.6.q-25
libx11-6:amd64	2:1.6.3-1ubuntu2
libx11-data	2:1.6.3-1ubuntu2
libx86-1:amd64	1.1+ds1-10
libxau6:amd64	1:1.0.8-1
libxcb1:amd64	1.11.1-1ubuntu1
libxdmcp6:amd64	1:1.1.2-1.1
libxen-4.6:amd64	4.6.5-0ubuntu1.4
libxenstore3.0:amd64	4.6.5-0ubuntu1.4
libxext6:amd64	2:1.3.3-1
libxml2:amd64	2.9.3+dfsg1-1ubuntu0.5
libxml2-utils	2.9.3+dfsg1-1ubuntu0.5
libxmuu1:amd64	2:1.1.2-2
libxslt1.1:amd64	1.1.28-2.1ubuntu0.1

libxtables11:amd64	1.6.0-2ubuntu3
libyajl2:amd64	2.1.0-2
linux-base	4.0ubuntu1
linux-firmware	1.157.17
linux-headers-4.4.0-112	4.4.0-112.135
linux-headers-4.4.0-112-generic	4.4.0-112.135
linux-headers-4.4.0-116	4.4.0-116.140
linux-headers-4.4.0-116-generic	4.4.0-116.140
linux-headers-4.4.0-62	4.4.0-62.83
linux-headers-4.4.0-62-generic	4.4.0-62.83
linux-headers-generic	4.4.0.116.122
linux-image-4.4.0-112-generic	4.4.0-112.135
linux-image-4.4.0-116-generic	4.4.0-116.140
linux-image-4.4.0-62-generic	4.4.0-62.83
linux-image-extra-4.4.0-112-generic	4.4.0-112.135
linux-image-extra-4.4.0-116-generic	4.4.0-116.140
linux-image-extra-4.4.0-62-generic	4.4.0-62.83
linux-signed-generic	4.4.0.116.122
linux-signed-image-4.4.0-112-generic	4.4.0-112.135
linux-signed-image-4.4.0-116-generic	4.4.0-116.140
linux-signed-image-4.4.0-62-generic	4.4.0-62.83
linux-signed-image-generic	4.4.0.116.122
locales	2.23-0ubuntu10
login	1:4.2-3.1ubuntu5.3
logrotate	3.8.7-2ubuntu2.16.04.2
lsb-base	9.20160110ubuntu0.2
lsb-release	9.20160110ubuntu0.2
lshw	02.17-1.1ubuntu3.4
lsof	4.89+dfsg-0.1
ltrace	0.7.3-5.1ubuntu4
lvm2	2.02.133-1ubuntu10
lxc-common	2.0.8-0ubuntu1~16.04.2
lxcfs	2.0.8-0ubuntu1~16.04.2
lxd	2.0.11-0ubuntu1~16.04.4
lxd-client	2.0.11-0ubuntu1~16.04.4
makedev	2.3.1-93ubuntu2~ubuntu16.04.1

man-db	2.7.5-1
manpages	4.04-2
mawk	1.3.3-17ubuntu2
mc	3:4.8.15-2
mc-data	3:4.8.15-2
mdadm	3.3-2ubuntu7.6
memtest86+	5.01-3ubuntu2
mime-support	3.59ubuntu1
mlocate	0.26-1ubuntu2

mokutil	0.3.0-0ubuntu3
mount	2.27.1-6ubuntu3.4
msr-tools	1.3-2
mtr-tiny	0.86-1ubuntu0.1
multiarch-support	2.23-0ubuntu10
nano	2.5.3-2ubuntu2
ncurses-base	6.0+20160213-1ubuntu1
ncurses-bin	6.0+20160213-1ubuntu1
ncurses-term	6.0+20160213-1ubuntu1
net-tools	1.60-26ubuntu1
netbase	5.3
netcat-openbsd	1.105-7ubuntu1
ntfs-3g	1:2015.3.14AR.1-1ubuntu0.1
ntp	1:4.2.8p4+dfsg-3ubuntu5.8
open-iscsi	2.0.873+git0.3b4b4500-14ubuntu3.4
open-vm-tools	2:10.0.7 3227872-5ubuntu1~16.04.2
openssh-client	1:7.2p2-4ubuntu2.4
openssh-server	1:7.2p2-4ubuntu2.4
openssh-sftp-server	1:7.2p2-4ubuntu2.4
openssl	1.0.2g-1ubuntu4.10
os-prober	1.70ubuntu3.3
overlayroot	0.27ubuntu1.5
parted	3.2-15ubuntu0.1
passwd	1:4.2-3.1ubuntu5.3
pastebinit	1.5-1
patch	2.7.5-1
pciutils	1:3.3.1-1.1ubuntu1.1
perl	5.22.1-9ubuntu0.2
perl-base	5.22.1-9ubuntu0.2
perl-modules-5.22	5.22.1-9ubuntu0.2
plymouth	0.9.2-3ubuntu13.2
plymouth-theme-ubuntu-text	0.9.2-3ubuntu13.2
pm-utils	1.4.1-16
policykit-1	0.105-14.1
popularity-contest	1.64ubuntu2
powermgmt-base	1.31+nmu1
procps	2:3.3.10-4ubuntu2.3
psmisc	22.21-2.1build1
python	2.7.12-1~16.04
python-apt-common	1.1.0~beta1ubuntu0.16.04.1
python-crypto	2.6.1-6ubuntu0.16.04.2
python-distorm3	3.3.0-3
python-minimal	2.7.12-1~16.04
python2.7	2.7.12-1ubuntu0~16.04.3
python2.7-minimal	2.7.12-1ubuntu0~16.04.3

python3	3.5.1-3
python3-apport	2.20.1-0ubuntu2.15
python3-apt	1.1.0~beta1ubuntu0.16.04.1
python3-chardet	2.3.0-2
python3-commandnotfound	0.3ubuntu16.04.2
python3-dbus	1.2.0-3
python3-debian	0.1.27ubuntu2
python3-distupgrade	1:16.04.24
python3-gdbm:amd64	3.5.1-1
python3-gi	3.20.0-0ubuntu1
python3-minimal	3.5.1-3
python3-newt	0.52.18-1ubuntu2
python3-pkg-resources	20.7.0-1
python3-problem-report	2.20.1-0ubuntu2.15
python3-pycurl	7.43.0-1ubuntu1
python3-requests	2.9.1-3
python3-six	1.10.0-3
python3-software-properties	0.96.20.7
python3-systemd	231-2build1
python3-update-manager	1:16.04.12
python3-urllib3	1.13.1-2ubuntu0.16.04.1
python3.5	3.5.2-2ubuntu0~16.04.4
python3.5-minimal	3.5.2-2ubuntu0~16.04.4
qemu-block-extra:amd64	1:2.5+dfsg-5ubuntu10.24
qemu-kvm	1:2.5+dfsg-5ubuntu10.24
qemu-system-common	1:2.5+dfsg-5ubuntu10.24
qemu-system-x86	1:2.5+dfsg-5ubuntu10.24
qemu-utils	1:2.5+dfsg-5ubuntu10.24
readline-common	6.3-8ubuntu2
rename	0.20-4
resolvconf	1.78ubuntu6
rsync	3.1.1-3ubuntu1.2
rsyslog	8.16.0-1ubuntu3
run-one	1.17-0ubuntu1
sbsigntool	0.6-0ubuntu10.1
screen	4.3.1-2build1
seabios	1.8.2-1ubuntu1
secureboot-db	1.1
sed	4.2.2-7
sensible-utils	0.0.9ubuntu0.16.04.1
sgml-base	1.26+nmu4ubuntu1
shared-mime-info	1.5-2ubuntu0.1
sharutils	1:4.15.2-1
shim	13-0ubuntu2
shim-signed	1.33.1~16.04.1+13-0ubuntu2

snap-confine	2.29.4.2
snapd	2.29.4.2
software-properties-common	0.96.20.7
sosreport	3.5-1~ubuntu16.04.2
squashfs-tools	1:4.3-3ubuntu2.16.04.1
ssh-import-id	5.5-0ubuntu1
strace	4.11-1ubuntu3
sudo	1.8.16-0ubuntu1.5
systemd	229-4ubuntu21.1
systemd-sysv	229-4ubuntu21.1
sysv-rc	2.88dsf-59.3ubuntu2
sysvinit-utils	2.88dsf-59.3ubuntu2
tar	1.28-2.1ubuntu0.1
tasksel	3.34ubuntu3
tasksel-data	3.34ubuntu3
tcpd	7.6.q-25
tcpdump	4.9.2-0ubuntu0.16.04.1
telnet	0.17-40
time	1.7-25.1
tmux	2.1-3build1
tzdata	2017c-0ubuntu0.16.04
ubuntu-cloudimage-keyring	2013.11.11
ubuntu-core-launcher	2.29.4.2
ubuntu-keyring	2012.05.19
ubuntu-minimal	1.361.1
ubuntu-release-upgrader-core	1:16.04.24
ubuntu-server	1.361.1
ubuntu-standard	1.361.1
ucf	3.0036

udev	229-4ubuntu21.1
ufw	0.35-0ubuntu2
uidmap	1:4.2-3.1ubuntu5.3
unattended-upgrades	0.90ubuntu0.9
unzip	6.0-20ubuntu1
update-manager-core	1:16.04.12
update-notifier-common	3.168.7
ureadahead	0.100.0-19
usbutils	1:007-4
util-linux	2.27.1-6ubuntu3.4
uuid-runtime	2.27.1-6ubuntu3.4
vbetool	1.1-3
vim	2:7.4.1689-3ubuntu1.2
vim-common	2:7.4.1689-3ubuntu1.2
vim-runtime	2:7.4.1689-3ubuntu1.2
vim-tiny	2:7.4.1689-3ubuntu1.2

vlan	1.9-3.2ubuntu1.16.04.4
wget	1.17.1-1ubuntu1.3
whiptail	0.52.18-1ubuntu2
wireless-regdb	2015.07.20-1ubuntu1
xauth	1:1.0.9-1ubuntu2
xdg-user-dirs	0.15-2ubuntu6
xfsprogs	4.3.0+nmu1ubuntu1.1
xkb-data	2.16-1ubuntu1
xml-core	0.13+nmu2
xz-utils	5.1.1alpha+20120614-2ubuntu2
zerofree	1.0.3-1
zlib1g:amd64	1:1.2.8.dfsg-2ubuntu4.1

Додаток В

Конфігураційні файли вузла віртуалізації

Вміст файлу /etc/fstab

# <file system>	<mount point>	<type>	<options>	<dump>	<pass>
/dev/sda2	/	ext4	errors=remount-ro	0	1
/dev/sda1	/boot/efi	vfat	umask=0077	0	1
/dev/sda3	none	swap	sw	0	0
/dev/sdb4	/mnt/forensic-data	ext4	errors=remount-ro	0	2

Вміст файлу /etc/network/interfaces

```
# Цей файл описує мережеві інтерфейси, доступні у вашій системі
# та спосіб їх активації. Для отримання додаткової інформації дивіться інтерфейси (5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto enp7s0f0
iface enp7s0f0 inet manual

auto br0
    iface br0 inet static
    address 192.168.10.1
    broadcast 192.168.10.255
    bridge_ports enp7s0f0
    bridge_stp off
    bridge_fd 0
    bridge_maxwait 0
```

Вміст файлу /etc/libvirt/storage/lvm-group.xml

```
<!--
WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY TO BE
OVERWRITTEN AND LOST. Changes to this xml configuration should be made
using: virsh pool-edit lvm-group or other application using the libvirt
API.
-->

<pool type='logical'>
  <name>lvm-group</name>
  <uuid>fb1197d6-d126-4ce9-a2ef-daf818092031</uuid>
  <capacity unit='bytes'>0</capacity>
  <allocation unit='bytes'>0</allocation>
  <available unit='bytes'>0</available>
  <source>
    <name>lvm-group</name>
    <format type='lvm2'/>
  </source>
  <target>
    <path>/dev/lvm-group</path>
  </target>
</pool>
```

ДОДАТОК Г. ФАЙЛИ КОНФІГУРАЦІЇ ВІРТУАЛЬНИХ МАШИН

Вміст файлу /mnt/kvm/xmls/VM1.xml.

```
<domain type='kvm'>
  <name>VM1_Ubuntu_17.10</name>
  <uuid>be0c72a2-c9d7-4dd5-83e3-b2084387be81</uuid>
  <memory unit='KiB'>4194304</memory>
  <currentMemory unit='KiB'>4194304</currentMemory>
  <vcpu placement='static'>8</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-i440fx-xenial'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
  </features>
  <cpu mode='custom' match='exact'>
    <model fallback='allow'>Penryn</model>
  </cpu>
  <clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='hpet' present='no'/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <pm>
    <suspend-to-mem enabled='no'/>
    <suspend-to-disk enabled='no'/>
  </pm>
  <devices>
    <emulator>/usr/bin/kvm-spice</emulator>
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source dev='/dev/lvm-group/VM1_ubuntu'/>
      <backingStore/>
      <target dev='hda' bus='ide'/>
      <alias name='ide0-0-0'/>
```

```
  <address type='drive' controller='0' bus='0' target='0' unit='0'/>
</disk>
<controller type='usb' index='0' model='ich9-ehci1'>
  <alias name='usb'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x06' \
  function='0x7'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci1'>
  <alias name='usb'/>
  <master startport='0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x06' \
  function='0x0' multifunction='on'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci2'>
  <alias name='usb'/>
  <master startport='2'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x06' \
  function='0x1'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci3'>
  <alias name='usb'/>
  <master startport='4'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x06' \
  function='0x2'/>
</controller>
<controller type='pci' index='0' model='pci-root'>
  <alias name='pci.0'/>
</controller>
<controller type='ide' index='0'>
  <alias name='ide'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x01' \
  function='0x1'/>
</controller>
<controller type='virtio-serial' index='0'>
  <alias name='virtio-serial0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x0'/>
</controller>
<interface type='network'>
  <mac address='52:54:00:9e:d3:6c'/>
  <source bridge='br0'/>
  <target dev='vnet0'/>
  <model type='rtl8139'/>
  <alias name='net0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' \
```

```
      function='0x0'/>
    </interface>
    <serial type='pty'>
      <source  path='/dev/pts/1'/>
      <target port='0'/>
      <alias name='serial0'/>
    </serial>
    <console type='pty'  tty='/dev/pts/1'>
      <source  path='/dev/pts/1'/>
      <target  type='serial'  port='0'/>
      <alias name='serial0'/>
    </console>
    <channel  type='spicevmc'>
      <target type='virtio'  name='com.redhat.spice.0'  state='disconnected'/>
      <alias name='channel0'/>
      <address  type='virtio-serial'  controller='0'  bus='0'  port='1'/>
    </channel>
    <input  type='mouse'  bus='ps2'/>
    <input type='keyboard'  bus='ps2'/>
    <graphics type='spice' port='5900' autoport='yes' listen='127.0.0.1'>
      <listen type='address' address='127.0.0.1'/>
    </graphics>
    <video>
      <model type='qxl'  ram='65536'  vram='65536'  vgamem='16384'  heads='1'/>
      <alias name='video0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' \
      function='0x0'/>
    </video>
    <redirdev bus='usb' type='spicevmc'>
      <alias  name='redir0'/>
    </redirdev>
    <redirdev bus='usb' type='spicevmc'>
      <alias  name='redir1'/>
    </redirdev>
    <memballoon model='virtio'>
      <alias name='balloon0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x07' \
      function='0x0'/>
    </memballoon>
  </devices>
  <seclabel type='dynamic'  model='apparmor'  relabel='yes'>
    <label>libvirt-be0c72a2-c9d7-4dd5-83e3-b2084387be81</label>
    <imagelabel>libvirt-be0c72a2-c9d7-4dd5-83e3-b2084387be81</imagelabel>
  </seclabel>
</domain>
```

Вміст файлу /mnt/kvm/xmls/VM2.xml.

```
<domain type='kvm'>
```

```
<name>VM2_CentOS_7</name>
<uuid>999170f4-75c1-4d45-b875-fb77cc2a00ea</uuid>
<memory unit='KiB'>4194304</memory>
<currentMemory unit='KiB'>4194304</currentMemory>
<vcpu placement='static'>8</vcpu>
<resource>
  <partition>/machine</partition>
</resource>
<os>
  <type arch='x86_64' machine='pc-i440fx-xenial'>hvm</type>
  <boot dev='hd'/>
</os>
<features>
  <acpi/>
  <apic/>
</features>
<cpu mode='custom' match='exact'>
  <model fallback='allow'>Penryn</model>
</cpu>
<clock offset='utc'>
  <timer name='rtc' tickpolicy='catchup'/>
  <timer name='pit' tickpolicy='delay'/>
  <timer name='hpet' present='no'/>
</clock>
<on_poweroff>destroy</on_poweroff>
<on_reboot>restart</on_reboot>
<on_crash>restart</on_crash>
<pm>
  <suspend-to-mem enabled='no'/>
  <suspend-to-disk enabled='no'/>
</pm>
<devices>
  <emulator>/usr/bin/kvm-spice</emulator>
  <disk type='block' device='disk'>
    <driver name='qemu' type='raw' cache='none' io='native'/>
    <source dev='/dev/lvm-group/VM2_centos'/>
    <backingStore/>
    <target dev='hda' bus='ide'/>
    <alias name='ide0-0-0'/>
    <address type='drive' controller='0' bus='0' target='0' unit='0'/>
  </disk>
  <controller type='usb' index='0' model='ich9-ehci1'>
```

```
  <alias name='usb'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x7'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci1'>
  <alias name='usb'/>
  <master startport='0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x0' multifunction='on'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci2'>
  <alias name='usb'/>
  <master startport='2'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x1'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci3'>
  <alias name='usb'/>
  <master startport='4'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x2'/>
</controller>
<controller type='pci' index='0' model='pci-root'>
  <alias name='pci.0'/>
</controller>
<controller type='ide' index='0'>
  <alias name='ide'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x01' \
  function='0x1'/>
</controller>
<controller type='virtio-serial' index='0'>
  <alias name='virtio-serial0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x04' \
  function='0x0'/>
</controller>
<interface type='bridge'>
  <mac   address='52:54:00:f6:eb:05'/>
  <source bridge='br0'/>
  <target dev='vnet1'/>
  <model type='rtl8139'/>
  <alias name='net0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' \
  function='0x0'/>
</interface>
<serial type='pty'>
```

```
      <source  path='/dev/pts/5'/>
      <target port='0'/>
      <alias name='serial0'/>
    </serial>
    <console type='pty'  tty='/dev/pts/5'>
      <source  path='/dev/pts/5'/>
      <target  type='serial'  port='0'/>
      <alias name='serial0'/>
    </console>
    <channel  type='spicevmc'>
      <target type='virtio'  name='com.redhat.spice.0'  state='disconnected'/>
      <alias name='channel0'/>
      <address  type='virtio-serial'  controller='0'  bus='0'  port='1'/>
    </channel>
    <input  type='mouse'  bus='ps2'/>
    <input type='keyboard'  bus='ps2'/>
    <graphics type='spice'  port='5901'  autoport='yes'  listen='127.0.0.1'>
      <listen type='address'  address='127.0.0.1'/>
    </graphics>
    <video>
      <model type='qxl'  ram='65536'  vram='65536'  vgamem='16384'  heads='1'/>
      <alias name='video0'/>
      <address type='pci'  domain='0x0000'  bus='0x00'  slot='0x02'  \
      function='0x0'/>
    </video>
    <redirdev bus='usb'  type='spicevmc'>
      <alias  name='redir0'/>
    </redirdev>
    <redirdev bus='usb'  type='spicevmc'>
      <alias  name='redir1'/>
    </redirdev>
    <memballoon model='virtio'>
      <alias name='balloon0'/>
      <address type='pci'  domain='0x0000'  bus='0x00'  slot='0x06'  \
      function='0x0'/>
    </memballoon>
  </devices>
  <seclabel type='dynamic'  model='apparmor'  relabel='yes'>
    <label>libvirt-999170f4-75c1-4d45-b875-fb77cc2a00ea</label>
    <imagelabel>libvirt-999170f4-75c1-4d45-b875-fb77cc2a00ea</imagelabel>
  </seclabel>
</domain>
```

Вміст файлу /mnt/kvm/xmls/VM3.xml.

```
<domain type='kvm'>
  <name>VM3_Windows_2008</name>
  <uuid>9f44405d-2df8-483d-92f8-2050756235fd</uuid>
  <memory  unit='KiB'>4194304</memory>
```

```
<currentMemory  unit='KiB'>4194304</currentMemory>
<vcpu  placement='static'>8</vcpu>
<resource>
  <partition>/machine</partition>
</resource>
<os>
  <type arch='x86_64'  machine='pc-i440fx-xenial'>hvm</type>
  <boot dev='hd'/>
</os>
<features>
  <acpi/>
  <apic/>
  <hyperv>
    <relaxed state='on'/>
    <vapic state='on'/>
    <spinlocks  state='on'  retries='8191'/>
  </hyperv>
</features>
<cpu  mode='custom'  match='exact'>
  <model fallback='allow'>Penryn</model>
</cpu>
<clock offset='localtime'>
  <timer name='rtc'  tickpolicy='catchup'/>
  <timer  name='pit'  tickpolicy='delay'/>
  <timer  name='hpet'  present='no'/>
  <timer name='hypervclock'  present='yes'/>
</clock>
<on_poweroff>destroy</on_poweroff>
<on_reboot>restart</on_reboot>
<on_crash>restart</on_crash>
<pm>
  <suspend-to-mem enabled='no'/>
  <suspend-to-disk  enabled='no'/>
</pm>
<devices>
  <emulator>/usr/bin/kvm-spice</emulator>
  <disk  type='block'  device='disk'>
    <driver name='qemu'  type='raw'  cache='none'  io='native'/>
    <source dev='/dev/lvm-group/VM3_win2008'/>
    <backingStore/>
    <target  dev='hda'  bus='ide'/>
```

```
  <alias name='ide0-0-0'/>
  <address type='drive' controller='0' bus='0' target='0' unit='0'/>
</disk>
<controller type='usb' index='0' model='ich9-ehci1'>
  <alias name='usb'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x7'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci1'>
  <alias name='usb'/>
  <master startport='0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x0' multifunction='on'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci2'>
  <alias name='usb'/>
  <master startport='2'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x1'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci3'>
  <alias name='usb'/>
  <master startport='4'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x2'/>
</controller>
<controller type='pci' index='0' model='pci-root'>
  <alias name='pci.0'/>
</controller>
<controller type='ide' index='0'>
  <alias name='ide'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x01' \
  function='0x1'/>
</controller>
<controller type='virtio-serial' index='0'>
  <alias name='virtio-serial0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x04' \
  function='0x0'/>
</controller>
<interface type='network'>
  <mac   address='52:54:00:70:30:7f'/>
  <source bridge='br0'/>
  <target dev='vnet2'/>
  <model type='rtl8139'/>
  <alias name='net0'/>
```

```
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' \
      function='0x0'/>
    </interface>
    <serial type='pty'>
      <source  path='/dev/pts/6'/>
      <target port='0'/>
      <alias name='serial0'/>
    </serial>
    <console type='pty'  tty='/dev/pts/6'>
      <source  path='/dev/pts/6'/>
      <target  type='serial'  port='0'/>
      <alias name='serial0'/>
    </console>
    <channel type='spicevmc'>
      <target type='virtio'  name='com.redhat.spice.0'  state='disconnected'/>
      <alias name='channel0'/>
      <address  type='virtio-serial'  controller='0'  bus='0'  port='1'/>
    </channel>
    <input type='tablet'  bus='usb'>
      <alias name='input0'/>
    </input>
    <input type='mouse'  bus='ps2'/>
    <input type='keyboard'  bus='ps2'/>
    <graphics type='spice' port='5902' autoport='yes' listen='127.0.0.1'>
      <listen type='address' address='127.0.0.1'/>
    </graphics>
    <video>
      <model type='qxl'  ram='65536'  vram='65536'  vgamem='16384'  heads='1'/>
      <alias name='video0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' \
      function='0x0'/>
    </video>
    <redirdev bus='usb' type='spicevmc'>
      <alias  name='redir0'/>
    </redirdev>
    <redirdev bus='usb' type='spicevmc'>
      <alias  name='redir1'/>
    </redirdev>
    <memballoon model='virtio'>
      <alias name='balloon0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x06' \
      function='0x0'/>
    </memballoon>
  </devices>
  <seclabel type='dynamic'  model='apparmor'  relabel='yes'>
```

```
    <label>libvirt-9f44405d-2df8-483d-92f8-2050756235fd</label>
    <imagelabel>libvirt-9f44405d-2df8-483d-92f8-2050756235fd</imagelabel>
  </seclabel>
</domain>
```

Вміст файлу /mnt/kvm/xmls/VM4.xml.

```
<domain type='kvm'>
  <name>VM4_Windows_2016</name>
  <uuid>5463353b-7bf4-49f0-b48f-1ff20d9c499a</uuid>
  <memory unit='KiB'>4194304</memory>
  <currentMemory unit='KiB'>4194304</currentMemory>
  <vcpu placement='static'>8</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-i440fx-xenial'>hvm</type>
  </os>
  <features>
    <acpi/>
    <apic/>
  </features>
  <cpu mode='custom' match='exact'>
    <model fallback='allow'>Penryn</model>
  </cpu>
  <clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='hpet' present='no'/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <pm>
    <suspend-to-mem enabled='no'/>
    <suspend-to-disk enabled='no'/>
  </pm>
  <devices>
    <emulator>/usr/bin/kvm-spice</emulator>
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source dev='/dev/lvm-group/VM4_win2016'/>
      <backingStore/>
      <target dev='hda' bus='ide'/>
```

```
  <boot order='2'/>
  <alias name='ide0-0-0'/>
  <address type='drive' controller='0' bus='0' target='0' unit='0'/>
</disk>
<controller type='usb' index='0' model='ich9-ehci1'>
  <alias name='usb'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x7'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci1'>
  <alias name='usb'/>
  <master startport='0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x0' multifunction='on'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci2'>
  <alias name='usb'/>
  <master startport='2'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x1'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci3'>
  <alias name='usb'/>
  <master startport='4'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' \
  function='0x2'/>
</controller>
<controller type='pci' index='0' model='pci-root'>
  <alias name='pci.0'/>
</controller>
<controller type='ide' index='0'>
  <alias name='ide'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x01' \
  function='0x1'/>
</controller>
<controller type='virtio-serial' index='0'>
  <alias name='virtio-serial0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x04' \
  function='0x0'/>
</controller>
<interface type='network'>
  <mac   address='52:54:00:24:13:f5'/>
  <source bridge='br0'/>
  <target dev='vnet3'/>
  <model type='rtl8139'/>
```

```
      <alias name='net0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' \
      function='0x0'/>
    </interface>
    <serial type='pty'>
      <source  path='/dev/pts/2'/>
      <target port='0'/>
      <alias name='serial0'/>
    </serial>
    <console type='pty'  tty='/dev/pts/2'>
      <source  path='/dev/pts/2'/>
      <target  type='serial'  port='0'/>
      <alias name='serial0'/>
    </console>
    <channel type='spicevmc'>
      <target type='virtio'  name='com.redhat.spice.0'  state='disconnected'/>
      <alias name='channel0'/>
      <address  type='virtio-serial'  controller='0'  bus='0'  port='1'/>
    </channel>
    <input type='mouse'  bus='ps2'/>
    <input type='keyboard'  bus='ps2'/>
    <graphics type='spice' port='5903' autoport='yes' listen='127.0.0.1'>
      <listen type='address' address='127.0.0.1'/>
    </graphics>
    <video>
      <model type='qxl'  ram='65536'  vram='65536'  vgamem='16384'  heads='1'/>
      <alias name='video0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' \
      function='0x0'/>
    </video>
    <redirdev bus='usb' type='spicevmc'>
      <alias  name='redir0'/>
    </redirdev>
    <redirdev bus='usb' type='spicevmc'>
      <alias  name='redir1'/>
    </redirdev>
    <memballoon model='virtio'>
      <alias name='balloon0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x06' \
      function='0x0'/>
    </memballoon>
  </devices>
  <seclabel type='dynamic'  model='apparmor'  relabel='yes'>
    <label>libvirt-5463353b-7bf4-49f0-b48f-1ff20d9c499a</label>
    <imagelabel>libvirt-5463353b-7bf4-49f0-b48f-1ff20d9c499a</imagelabel>
```

```
    </seclabel>
</domain>
```