МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

МАНДРИК ОЛЕКСАНДР МИКОЛАЙОВИЧ

Допускається до захисту:
Завідувач кафедри
інформаційних технологій,
к.т.н., доцент,
_____Нескородєва Т. В.
«__»_____20__р.

ВИЯВЛЕННЯ СЛІДІВ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ДАМПІ ОПЕРАТИВНОЇ ПАМ'ЯТІ.

Спеціальність 125 Кібербезпека
Кваліфікаційна (бакалаврська) робота

Науковий керівник:
Барибін О. І., доцент кафедри
інформаційних технологій
к.т.н.

_____
(підпис)

Оцінка :____/____ /_____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця 2021

АНОТАЦІЯ

Мандрик О. М Виявлення слідів шкідливого програмного забезпечення в дампі оперативної памяті. Спеціальність 125 (Кібербезпека) Донецький національний університет імені Василя Стуса, Вінниця, 2021.

У кваліфікаційній роботі запропонована методологія для ведення рослідування з участю шкідливого програмного забезпечення, запропоновані варіанти дій в залежності від типу системи, наведені приклади використання програмного забезпечення.

Ключові слова: ЦИФРОВА КРИМІНАЛІСТИКА, ДАМПИ ОПЕРАТИВНОЇ ПАМЯТІ, АНАЛІЗ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

Рис: 9; Бібліограф: 10;

Mandryk O.M Detection of malware traces in the RAM dump. Specialty 125 (Cybersecurity)

Vasyl Stus Donetsk National University, Vinnytsia, 2021.

In the qualification work the methodology for conducting investigation with the participation of malicious software is offered, options of actions depending on type of system are offered, examples of use of the software are resulted.

Key words: DIGITAL FORENSICS, RAM DUMPS, MALWARE ANALYSIS.

Fig: 9; Bibliography: 10;

ЗМІСТ

ВСТУП .

Необхідність виявляти шкідливе ПЗ до того, як воно завдасть шкоди комп'ютерам, мобільним телефонам і іншим електронним пристроям, вже багато років привертає увагу дослідників і фахівців в галузі боротьби з шкідливими програмами. Щоб захистити користувачів від атак шкідливих програм, на комп'ютер завантажуються антивірусні програмні продукти. Антивірус в основному використовує сигнатурні методи для виявлення шкідливих програм. Однак цей метод не може виявити шкідливе програмне забезпечення, що використовує методи ізоляції або шифрування. Він також не може виявити невидимі (нові види шкідлвого програмного забезпечення).

Програми для криміналістичної експертизи вміють витягати багато цінної інформації з оперативної пам'яті, в тому числі фрагменти відкритих раніше файлів, фрагменти шкідливого коду, цінні об'єкти, як активні, так і завершенні: процеси, паролі PGP, інформацію про різної активності на комп'ютері . Правоохоронні органи в процесі розслідування покладаються на ці відомості як на реальні докази. Останнім часом аналіз RAM вважається майже таким же ефективним і надійним інструментом криміналістичної експертизи, як аналіз вмісту HDD[1].

Мета виконяння даної роботи – надання методичних рекомендацій для рослідуваннь з участю активних данних як прямих доказів.

Відповідно до сформульованої мети роботи **завданнями** роботи є:

- розглянути основні поняття цифрової криміналістики.
- розглянути найпопулярніші види програмного забезпечення, навести приклад їх використання та налаштування.
- обгрунтувати вибір програмного забезпечення в залежності від ситуації.
- Запропонувати схему вибору програмного забезпечення в залежності від ситуації.

Об’єктом бакалаврської роботи є процес впровадження методології для розслідуваннь в яких інформація з знімків стану системи є основним доказом.

Предметом дослідження є створення дампів оперативної памяті та подальша їх обробка для передачі спеціалістам з аналізу шкідливого програмного забезпечення.

Парший розділ описує теоретичні та технічні визначення з якими повинен бути знайомий спеціаліст який веде розслідування.

Другий розділ містить опис найпопулярнішого і найефективнішого програмного забезпечення, пропонує інструкцію по налаштуванню цього програмного забезпечення та надає рекомендації по вибору програмного забезпечення в залежності від ситуації.

Виконання роботи дозволяє ефективно отримувати дампи оперативної памяті для подальшого аналізу та передачі, також ефективно вибирати програмне забезпечення в різних ситуаціях.

РОЗДІЛ №1 - ВСТУП ДО КРИМІНАЛІСТИКИ ПАМ’ЯТІ

У цьому розділі наведено загальний огляд апаратних компонентів та структур операційної системи, які впливають на аналіз пам'яті Розділ починається з висвітлення важливих аспектів апаратної архітектури і завершується наданням огляду загальних примітивів операційної системи. Поняття та термінологія, які обговорюються в цьому розділі, будуть часто згадуватись в наступних розділах.

1.1 Огляд апаратної системи.

Фізична організація ПК складається з друкованих плат, які з'єднують різні компоненти та забезпечують роз'єми для периферійних пристроїв. Основна плата в системі цього типу, материнська плата, забезпечує з'єднання, що дозволяють компонентам системи взаємодіяти. Ці канали зв'язку зазвичай називають комп'ютерними шинами. У цьому розділі висвітлено компоненти та шини, з якими слід знати слідчого. Рис 1.1 ілюструє, як типово організовані різні компоненти.

Центральний процесор та блок управіння памятю. Два найважливіших компоненти на материнській платі - це процесор, який виконує програми, і основна пам’ять, яка тимчасово зберігає виконувані програми та пов’язані з ними дані. Процесор зазвичай називають центральним процесором (ЦП). Процесор отримує доступ до основної пам'яті для отримання її інструкцій, а потім виконує ці інструкції для обробки даних[2].

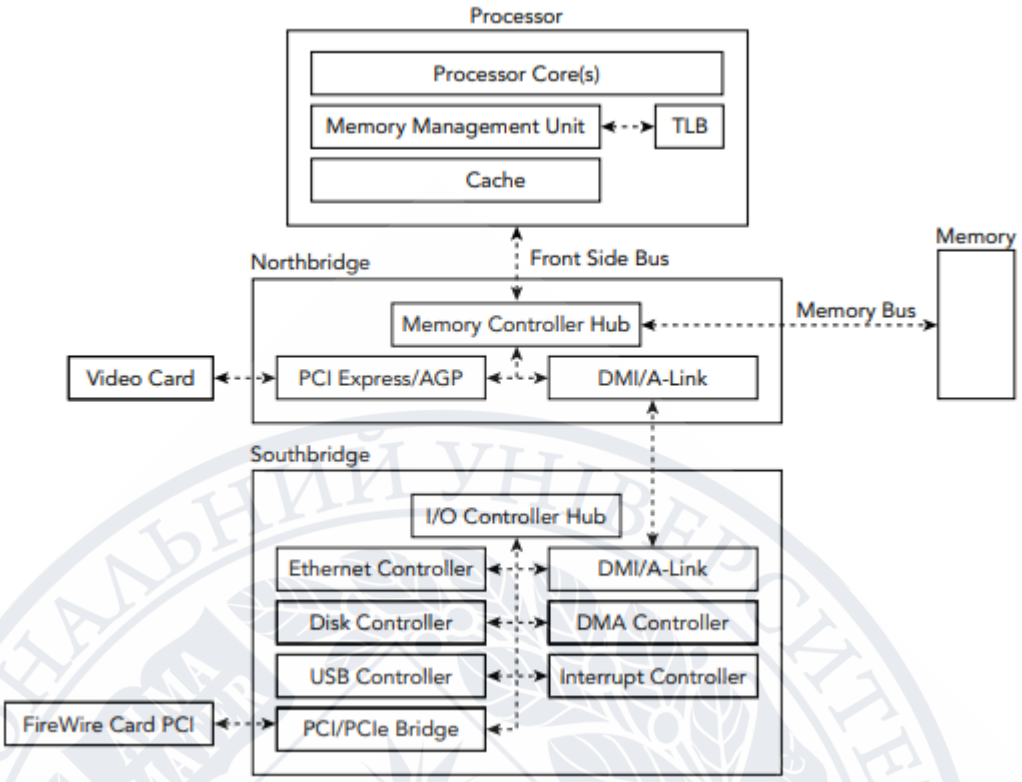

Рис 1-1: Фізична організація сучасної системи

Читання з основної пам'яті часто значно повільніше, ніж читання з власної пам'яті центрального процесора. Як результат, сучасні системи використовують декілька шарів швидкої пам'яті, які називаються кешами, щоб компенсувати цю диспропорцію. Кожен рівень кеш-пам'яті (L1, L2 тощо) є відносно повільнішим ніж його попередник. У більшості систем ці кеші вбудовані в процесор та кожне з його ядер. Якщо дані не знайдені в певному кеші, дані потрібно отримати з кешу наступного рівня або основної пам'яті.

Процесор покладається на свій блок управління пам’яттю (БУП), щоб допомогти знайти, де зберігаються дані. БУП - це апаратний блок, який переводить адресу, яку запитує процесор, на відповідну адресу в основній пам'яті. Як буде описано далі в цьому розділі, структури даних для управління перекладом адрес також зберігаються в основній пам'яті[2].

Оскільки для даного перекладу може знадобитися кілька операцій зчитування з пам'яті, процесор використовує спеціальний кеш, відомий як переклад зовнішнього буфера (ПЗБ), для таблиці перекладу БУП. Перед

кожним доступом до пам'яті проводиться опитування з ПЗБ, перш ніж просити БУП виконати дорогу операцію перекладу адреси.

Прямий доступ до пам'яті. Для підвищення загальної продуктивності більшість сучасних систем надають пристроям вводу-виводу можливість безпосередньої передачі даних, що зберігаються в системній пам'яті, без втручання процесора. Ця можливість називається прямим доступом до пам'яті (ПДП). До того, як був представлений ПДП, процесор використовувався в повному обсязі під час передачі вводу-виводу та часто виступав посередником. У сучасних архітектурах центральний процесор може ініціювати передачу даних і дозволити контролеру ПДП керувати передачею даних, або пристрій вводу-виводу може ініціювати передачу, незалежно від центрального процесора. Окрім очевидного впливу на продуктивність системи, ПДП також має важливі наслідки для криміналістики пам'яті. Він забезпечує механізм прямого доступу до вмісту фізичної пам'яті з периферійного пристрою без залучення ненадійного програмного забезпечення, що працює на машині[2]. Наприклад, шина PCI підтримує пристрої, які виконують функції ведучих шин, що означає, що вони можуть вимагати управління шиною для ініціації транзакцій. Як результат, пристрій PCI з функцією ведучої шини та підтримкою ПДП може отримати доступ до пам'яті системи, не залучаючи центральний процесор. Інший приклад - інтерфейс IEEE 1394, який зазвичай називають Firewire. Чіп контролера IEEE 1394 забезпечує однорангову послідовну карту розширення, призначену для підключення високошвидкісних периферійних пристроїв до ПК. Незважаючи на те, що інтерфейс IEEE 1394 зазвичай зустрічається лише у дорогих системах, цей інтерфейс можете бути доданий як до настільних ПК, так і до ноутбуків за допомогою карт розширення[3].

Енергонезалежна пам’ять (ОЗУ). Основна пам’ять ПК реалізована з оперативною пам’яттю (ОЗУ), яка зберігає код і дані, до яких процесор активно звертається і зберігає. На відміну від послідовного зберігання доступу, зазвичай асоційованого з дисками, довільний доступ відноситься до

характеристик постійного часу доступу незалежно від того, де дані зберігаються на носії. Основною пам’яттю більшості ПК є динамічна оперативна пам’ять (дОЗУ). Вона динамічна, оскільки використовує різницю між зарядженим і розрядженим станом конденсатора для зберігання біта даних. Щоб конденсатор підтримував цей стан, він повинен періодично переосмислювати завдання, яке зазвичай виконує контролер пам'яті. ОЗУ вважається енергозалежною пам'яттю, оскільки їй потрібно живлення, щоб дані залишались доступними. Таким чином, за винятком випадків атак холодного завантаження, після вимкнення ПК втрачається пам’ять[1]. Це основна причина, чому потібна методика вибору дій та вибору програмного забезпечення, якщо ви плануєте зберегти докази щодо поточного стану системи[3].

Архітектура центрального процесора.

Як зазначалося раніше, центральний процесор є одним з найважливіших компонентів комп’ютерної системи. Щоб ефективно витягти структуру з фізичної пам’яті та зрозуміти, наскільки шкідливий код може порушити безпеку системи, потрібно чітко розуміти модель програмування, яку забезпечує ЦП для доступу до пам’яті.

Адресні простори. Щоб ЦП виконував інструкції та отримував доступ до даних, що зберігаються в основній пам’яті, він повинен вказати унікальну адресу для цих даних. Процесори, використовують байтову адресацію, а доступ до пам'яті здійснюється послідовно. Адресний простір стосується діапазону дійсних адрес, що використовуються для ідентифікації даних, що зберігаються в межах кінцевого розподілу пам'яті[2]. Ця схема адресації зазвичай починається з байта 0 і закінчується на зміщенні кінцевого байта пам'яті при розподілі. Єдиний безперервний адресний простір, який піддається дії запущеної програми, називається лінійним адресним простором. На основі моделей пам'яті, та їх використання, використовуються лінійні адреси та віртуальні адреси як взаємозамінні. Ці адреси отримують шляхом перекладу лінійних адрес у фізичні, використовуючи одну або кілька таблиць сторінок.

Архітектура Intel IA-32. Архітектура IA-32 зазвичай відноситься до сімейства архітектур x86, які підтримують 32-розрядні обчислення. Зокрема вона визначає набір команд та середовище програмування для 32-розрядних процесорів Intel. IA-32 - це маленька ендіанна машина, яка використовує байтову адресацію. Програмне забезпечення, що працює на процесорі IA-32, може мати лінійний адресний простір та фізичний адресний простір до 4 Гб. Можна збільшити обсяг фізичної пам'яті до 64 ГБ за допомогою функції IA-32 Physical Address Extension (PAE)[2]. Цей підрозділ зосереджений на захищеному режимі роботи архітектури IA-32, який є операційним режимом, що забезпечує підтримку таких функцій, як віртуальна пам’ять, підкачка, рівні привілеїв та сегментація. Це основний стан процесора, а також режим, в якому працює більшість сучасних операційних систем.

Регістри. Архітектура IA-32 визначає невелику кількість надзвичайно швидкої пам'яті, яка називається регістрами, яку центральний процесор використовує для тимчасового зберігання під час обробки. Кожне ядро процесора містить вісім 32-розрядних регістрів загального призначення для виконання логічних та арифметичних операцій, а також кілька інших регістрів, які контролюють поведінку процесора. У цьому розділі висвітлено декілька регістрів управління, що мають відношення до аналізу пам'яті. Регістр EIP, який також називають лічильником програми, містить лінійну адресу наступної команди, яка виконується. Архітектура IA-32 також має п'ять регістрів управління, які визначають конфігурацію процесора та характеристики виконуваного завдання. CR0 містить прапори, які керують режимом роботи процесора, включаючи прапор, що дозволяє підкачати сторінку. CR1 зарезервований і не повинен бути доступним. CR2 містить лінійну адресу, яка спричинила помилку сторінки. CR3 містить фізичну адресу початкової структури, що використовується для перекладу адрес. Він оновлюється під час перемикання контексту, коли планується нове завдання. CR4 використовується для розширення архітектурних розширень, включаючи PAE[2].

Сегментація. Процесори IA-32 реалізують два механізми управління пам’яттю: сегментацію та пейджинг. Сегментація ділить 32-розрядний лінійний адресний простір на кілька сегментів змінної довжини. Усі посилання на пам'ять IA-32 адресуються за допомогою 16-бітового селектора сегментів, який ідентифікує конкретний дескриптор сегмента, та 32-бітового зміщення у зазначений сегмент. Дескриптор сегмента - це структура даних, що зберігається в пам'яті, яка визначає місце розташування, розмір, тип та дозволи для даного сегмента. Кожне ядро процесора містить два спеціальні регістри, GDTR і LDTR, які вказують на таблиці дескрипторів сегментів, які називаються глобальною таблицею дескрипторів (GDT) та локальною таблицею дескрипторів, відповідно. Реєстрації сегментації CS (для коду), SS (для стека) та DS, ES, FS та GS (кожен для даних) повинні завжди містять дійсні селектори[2]. Хоча сегментація є обов’язковою, операційні системи приховують сегментовану адресацію, визначаючи набір сегментів, що перекриваються, нульовою базовою адресою, створюючи тим самим видимість єдиного безперервного “плоского” лінійного адресного простору.

Однак захист сегментації все ще застосовується для кожного сегмента, і для посилань на код та дані потрібно використовувати окремі дескриптори сегментів.

Підкачка. Підкачка забезпечує можливість віртуалізації лінійного адресного простору. Він створює середовище виконання, в якому моделюється великий лінійний адресний простір із невеликим обсягом фізичної пам'яті та дискового сховища. Кожен 32-розрядний лінійний адресний простір розбивається на розділи фіксованої довжини, що називаються сторінками, і які можуть бути відображені у фізичні пам’яті у довільному порядку. Коли програма намагається отримати доступ до лінійної адреси, це відображення використовує каталоги сторінок та таблиці сторінок, що перебувають у пам'яті, щоб перевести лінійну адресу у фізичну[2]. У типовому сценарії сторінки розміром 4 КБ, як показано на рисунку(1.2), 32-розрядна віртуальна

адреса розбивається на три розділи, кожен з яких використовується як індекс в ієрархії структури підкачки або пов'язаної фізичної сторінки.

Архітектура IA-32 також підтримує сторінки розміром 4 МБ, для перекладу яких потрібен лише каталог сторінок. Використовуючи різні структури підкачки для різних процесів, операційна система може забезпечити кожному процесу вигляд однопрограмованого середовища через віртуалізований лінійний адресний простір. На рисунку(1.3) показано більш детальну розбивку бітів, які перетворюють віртуальну адресу на зміщення у фізичній пам'яті[2].

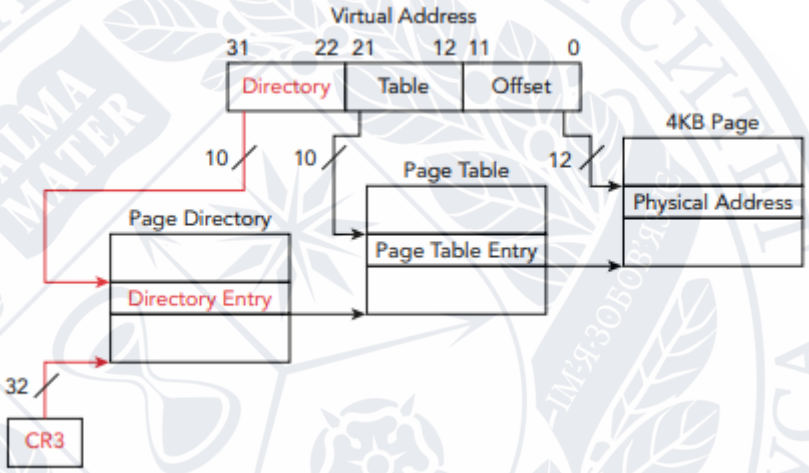

Рис 1.2: Переклад адреси на сторінку розміром 4 КБ із використанням 32-розрядної сторінки.

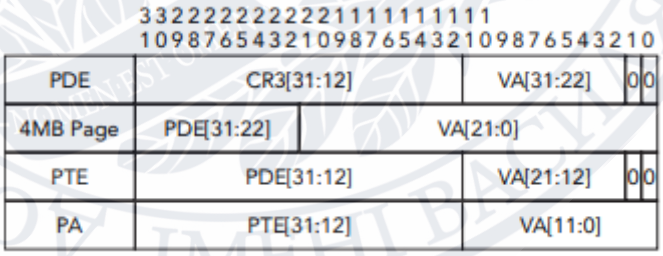

Рис 1.3: Формати адрес адрес структур підкачки, що використовуються у 32-розрядному підкачці.

Для обчислення адреси входу в каталог сторінки (PDE), поєднуються біти 31:12 з реєстру CR3 з бітами 31:22 з віртуальної адреси. Потім знаходиться запис таблиці сторінок (PTE), комбінуючи біти 31:12 з PDE з бітами 21:12 віртуальної адреси. Далі можна отримати фізичну адресу (PA), поєднавши біти 31:12 PTE з бітами 11: 0 віртуальної адреси[2].

Архітектура Intel 64. Середовище виконання архітектури Intel 64 схоже на IA-32, але є кілька відмінностей. Реєстри, виділені в архітектурі IA-32, все ще існують в межах Intel 64, але були розширені до 64 біт. Найважливіша зміна полягає в тому, що Intel 64 тепер може підтримувати 64-розрядні лінійні адреси. Як результат, архітектура Intel 64 підтримує лінійний адресний простір до 264 байт. Найновіші реалізації архітектури на даний момент не підтримують цілі 64 біти, лише 48-бітові лінійні адреси. Як результат, віртуальні адреси в цих системах мають канонічний формат. Це означає, що для бітів 63:48 встановлені або всі 1, або всі 0, залежно від стану біта 47. Наприклад, адреса 0xfffffa800ccc0b30 має біти 63:48, оскільки встановлений біт 47. Також важливо зосередитись на змінах в управлінні пам'яттю, оскільки вони мають прямий вплив на криміналістику пам’яті. Найголовніша відмінність полягає в тому, що архітектура Intel 64 тепер підтримує додатковий рівень структур підкачки, який називається сторінкою рівня 4 (PML4). Усі записи в ієрархії структур підкачки складають 64 біти, і вони можуть відображати віртуальні адреси на сторінки розміром 4 КБ, 2 МБ або 1 ГБ. На рисунку (1.4) наведено приклад перекладу адреси на сторінку розміром 4 Кбайт із використанням 64-бітної / IA-32e підкачки.

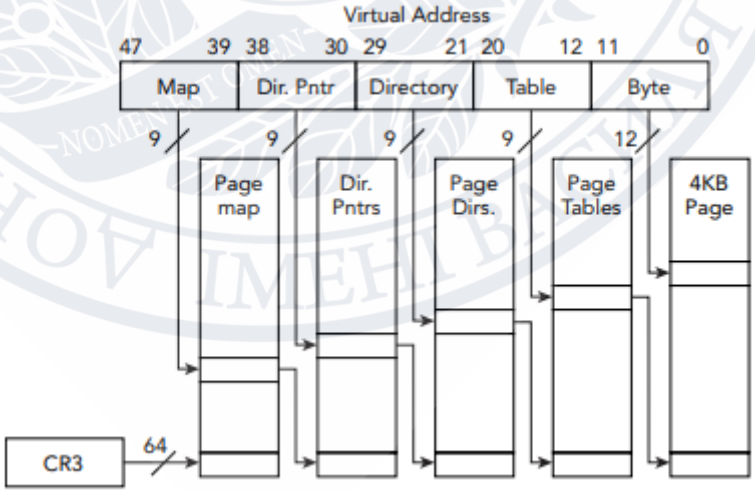

Рис 1.4: Переклад адреси на сторінку розміром 4 КБ із використанням 64-бітної / IA-32e підкачки.

На рисунку 1.8 показані формати адрес адрес структури структури підкачки, що використовуються в 64-бітній / IA-32e підкачці. Кожна структура підкачки складається з 512 записів (29) і індексується значеннями, вилученими з наступних діапазонів, взятих з 48-бітової віртуальної адреси:

- Біти 47:39 (зсув PML4E)
- Біти 38-30 (зсув PDPTE)
- Біти 29:21 (зсув PDE)
- Біти 20:12 (зсув PTE)

Якщо прапорець PS встановлений у PDPTE, запис відображає сторінку розміром 1 ГБ, якщо вона підтримується. Подібним чином, якщо прапор PS встановлено в PDE, PDE відображає сторінку розміром 2 МБ. Припускаючи, що присутні всі проміжні записи щоденника, останні 12 бітів визначають зміщення байтів у межах фізичної сторінки[2].

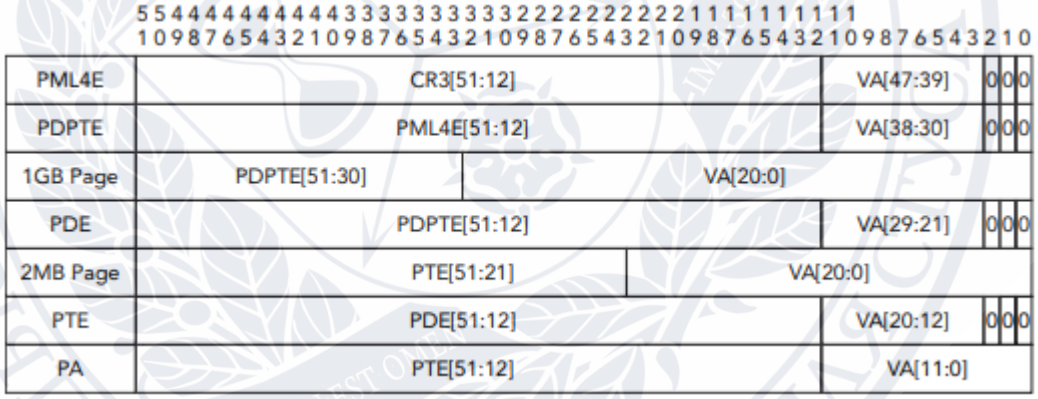

Рис 1.5: Формати адрес структур структури підкачки, що використовуються в 64-бітній / IA-32e підкачці.

Таблиця дескрипторів переривань.Архітектури ПК часто забезпечують механізм для переривання виконання процесу та передачі управління програмному режиму привілейованого режиму. Для архітектур IA-32 та Intel 64 підпрограми зберігаються в таблиці дескрипторів переривань (IDT). Кожен процесор має свій власний IDT, що складається з 256 8-байтових або 16-байтових записів, в яких перші 32 записи зарезервовані для визначених процесором винятків і переривань. Кожен запис містить адресу процедури

переривання (ISR), яка може обробляти певне переривання або виняток. У разі переривання або винятку зазначений номер переривання служить індексом IDT (який опосередковано посилається на сегмент у GDT), і ЦП викличе відповідний обробник[2]. Після більшості переривань операційна система відновить роботу там, де її було спочатку перервано. Наприклад, якщо потік намагається отримати доступ до сторінки пам'яті, яка є недійсною, це генерує помилку сторінки. Номер винятку 0xE обробляє помилки сторінок у архітектурах x86 та Intel 64. Таким чином, запис IDT для 0xE містить покажчик функції для обробника помилок сторінки операційної системи. Після запуску обробника помилок сторінки елемент керування може повернутися до потоку, який намагався отримати доступ до сторінки пам'яті. Операційні системи також використовують IDT для зберігання обробників для багатьох інших подій, включаючи системні виклики, точки зупинки налагоджувача та інші несправності.

1.2 Огляд операційної системи.

У цьому розділі наведено загальний огляд аспектів сучасних операційних систем, які впливають на криміналістику пам’яті. Зокрема, він зосереджується на важливих особливостях, спільних для трьох операційних систем, про які йдеться в цій книзі: Microsoft Windows, Linux та Mac OS X.

Поділ привілеїв. Щоб запобігти доступу потенційно несправних або шкідливих користувацьких програм до критичних компонентів операційної системи або маніпулювання ними, більшість сучасних операційних систем реалізують певну форму ізоляції привілеїв режиму користувача та ядра. Ця ізоляція намагається запобігти впливу програм на стабільність операційної системи або інших процесів. Код, пов'язаний з користувацькими програмами (ненадійними), виконується в користувацькому режимі, а код, пов'язаний з операційною системою (надійним), виконується в режимі ядра. як захисні кільця. У більшості операційних систем режим ядра реалізований у кільці 0 (найбільш привілейований), а режим користувача у кільці 3 (найменш привілейований). Коли процесор виконується в режимі ядра, код має

необмежений доступ до базового обладнання, включаючи привілейовані інструкції, та до областей пам'яті ядра та процесів (крім нових систем із SMEP, що запобігає виконанню сторінок користувача кільцевим 0). Щоб користувацька програма мала доступ до критичних компонентів операційної системи, програма перемикається з режиму користувача на режим ядра за допомогою чітко визначеного набору системних викликів. Розуміння рівня доступу, який отримав зловмисний код, може допомогти надати цінну інформацію про тип змін, які він може внести в систему.

Системні дзвінки. Операційні системи призначені для надання послуг користувацьким програмам. Застосування користувача запитує послугу з ядра операційної системи за допомогою системного дзвінка. Наприклад, коли додатку потрібно взаємодіяти з файлом, спілкуватися по мережі або породжувати інший процес, потрібні системні дзвінки. Як результат, системні дзвінки визначають API низького рівня між користувацькими програмами та ядром операційної системи. Зверніть увагу, що більшість програм не реалізуються безпосередньо з точки зору системних дзвінків. Натомість більшість операційних систем визначають набір стабільних API, які відображають один або кілька системних викликів (наприклад, API, надані ntdll.dll та kernel32.dll в Windows). Перш ніж користувацька програма зробить системний виклик, вона повинна налаштувати середовище виконання для передачі аргументів ядру через заздалегідь визначену конвенцію (наприклад, на стеку або в конкретних регістрах). Щоб викликати системний виклик, програма виконує програмне переривання або інструкцію, специфічну для архітектури, яка зберігає контекст реєстру режиму користувача, змінює режим виконання на ядро, ініціалізує стек ядра та викликає обробник системних викликів. Після обслуговування запиту виконання повертається в режим користувача і відновлюється непривілейований контекст реєстру. Потім керування повертається до інструкції після системного дзвінка. Оскільки це настільки важливий міст між користувальницькими програмами та операційною системою, код, що використовується для обслуговування

переривань системних викликів, зазвичай перехоплюється продуктами безпеки і націлюється шкідливим програмним забезпеченням.

Управління процесами. Процес - це екземпляр програми, що виконується в пам'яті. Операційна система відповідає за управління процесом створення, призупинення та припинення. У більшості сучасних операційних систем є функція, яка називається мультипрограмування, що дозволяє виконувати багато процесів одночасно. Коли програма виконується, створюється новий процес, пов’язаний із власним набором атрибутів, включаючи унікальний ідентифікатор процесу та адресний простір. Адресний простір процесу стає контейнером для коду програми, спільних бібліотек, динамічних даних та стеку середовища виконання. Процес також має принаймні єдиний потік виконання. Процес забезпечує середовище виконання, ресурси та контекст для запуску потоків. Важливим аспектом аналізу пам'яті є перелік процесів, які виконувались в системі, та аналіз даних, що зберігаються в їх адресних просторах, включаючи паролі, URL-адреси, ключі шифрування, електронну пошту та журнали.

Потоки. Потік - це основна одиниця використання та виконання процесора. Потік часто характеризується ідентифікатором потоку, набором регістрів ЦП та стеком (-ами) виконання, які допомагають визначити контекст виконання потоку. Незважаючи на свій унікальний контекст виконання, потоки процесу мають однаковий код, дані, адресний простір та ресурси операційної системи. Процес із кількома потоками може одночасно виконувати кілька завдань. Наприклад, один потік може спілкуватися по мережі, тоді як інший потік відображає дані на екрані. З точки зору криміналістики пам'яті, структури потокових даних корисні, оскільки вони часто містять мітки часу та початкові адреси. Ця інформація може допомогти вам визначити, який код у процесі був виконаний і коли він розпочався.

Планування процесора. Здатність операційної системи розподіляти час виконання процесора між кількома потоками називається плануванням процесора. Однією з цілей планування є оптимізація використання процесора,

коли потоки перемикаються вперед-назад між очікуванням операцій вводу-виводу та виконанням інтенсивних обчислень процесора. Планувальник операційної системи реалізує політики, які регулюють, які потоки виконуються та як довго вони виконуються. Переключення виконання одного потоку на інший називається перемиканням контексту. Контекст виконання включає значення регістрів ЦП, включаючи поточний вказівник інструкцій. Під час перемикання контексту операційна система призупиняє виконання потоку і зберігає його контекст виконання в основній пам'яті. Потім операційна система отримує контекст виконання іншого потоку з пам'яті, оновлює стан регістрів ЦП і відновлює виконання там, де раніше було призупинено. Збережений контекст виконання, пов'язаний із підвішеними потоками, може надати цінну інформацію під час аналізу пам'яті. Наприклад, він може надати детальну інформацію про те, які розділи коду виконувались або які параметри передавались системним викликам.

Системні ресурси

Ще однією важливою функцією, яку надає операційна система, є допомога в управлінні ресурсами «процесу». Як вже згадувалося раніше, процес діє як контейнер для системних ресурсів, доступних для його потоків. Більшість сучасних операційних систем підтримують структури даних для управління ресурсами, до яких активно здійснюється доступ, які процеси можуть отримати до них доступ та спосіб їх доступу. Приклади ресурсів операційної системи, які зазвичай відстежуються, включають процеси, потоки, файли, мережеві сокети, об'єкти синхронізації та регіони спільної пам'яті. Тип керованих ресурсів та структури даних, що використовуються для їх відстеження, часто відрізняються між операційними системами. Наприклад, Windows використовує об'єктний менеджер для нагляду за використанням системних ресурсів і згодом зберігає цю інформацію в таблиці дескрипторів. Дескриптор надає процесу унікальний ідентифікатор для доступу та керування системними ресурсами. Він також використовується для забезпечення контролю доступу до цих ресурсів та відстеження їх використання. Linux і Mac

використовують дескриптори файлів подібним чином. Далі буде описано, як витягти цю інформацію з таблиць дескрипторів дескрипторів або файлів і як використовувати її, щоб отримати уявлення про діяльність цього процесу.

Управління пам'яттю. Управління пам’яттю відноситься до алгоритмів операційної системи для управління розподілом, вивільненням та організацією фізичної пам’яті. Ці алгоритми часто залежать від обговорюваної раніше технічної підтримки.

Віртуальна пам’ять. Операційні системи забезпечують кожен процес своїм приватним віртуальним адресним простором. Ця абстракція створює поділ між логічною пам’яттю, яку бачить процес, і фактичною фізичною пам’яттю, встановленою на машині. Як результат, можна писати програми так, ніби вони мають доступ до всього адресного простору і в яких усі діапазони є резидентами пам'яті. Насправді деякі сторінки адресного простору можуть не бути резидентними. За лаштунками менеджер пам’яті відповідає за передачу областей пам’яті у друге сховище, щоб звільнити місце у фізичній пам’яті. Під час виконання менеджер пам'яті та БУП спільно працюють над перетворенням віртуальної адреси у фізичну. Якщо потік отримує доступ до віртуальної адреси, яка була переміщена до вторинного сховища, ці дані потім повертаються у фізичну пам’ять (як правило, через помилку сторінки). Ця взаємодія представлена на рисунку 1.6. Фактичний розмір віртуального адресного простору часто залежить від характеристик обладнання та операційної системи. Операційні системи часто розділяють діапазон доступних адрес на ті адреси, пов'язані з операційною системою, і ті, які є приватними для процесу. Діапазон адрес, зарезервованих для операційної системи, як правило, узгоджується з усіма процесами, тоді як приватні діапазони залежать від процесу, який виконується. За допомогою апаратного забезпечення менеджер пам'яті може розділити дані, щоб запобігти зловмисному або неправильному процесу від читання або запису пам'яті, що належить до пам'яті ядра або інших процесів.

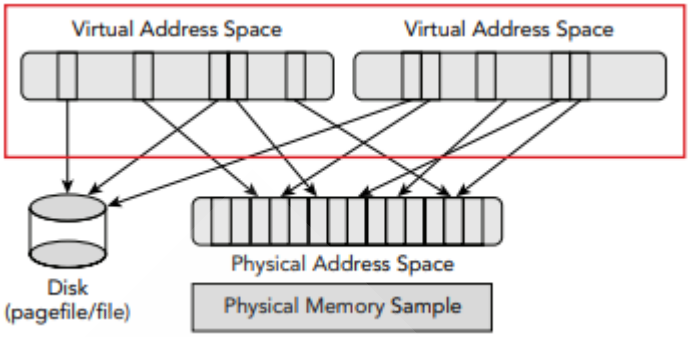

Рис 1.6: Ілюстрація декількох віртуальних адресних просторів, що використовують спільну пам'ять та додаткове сховище.

Підкачка по запиту. Механізмом, який зазвичай використовується для реалізації віртуальної пам’яті, є підкачування сторінок попиту, яка є політикою управління пам’яттю для визначення того, які регіони є основними в основній пам’яті, а які переміщуються до більш повільного вторинного сховища, коли виникає потреба. Найпоширенішим вторинним сховищем є файл або розділ на внутрішньому диску, який іменується відповідно файлом сторінки або свопом. Впровадження підкачи намагається завантажити в пам’ять лише ті сторінки, які насправді потрібні, на відміну від цілих процесів.

Підкачка сторінок по запиту спирається на характеристику використання пам'яті, відому як місцевість посилання, яка базується на спостереженні, що місця пам'яті, ймовірно, будуть часто доступні за короткий проміжок часу, як і їхні сусіди. В ідеалі, сторінка підкачки зменшує час, необхідний для завантаження процесу в пам’ять, і збільшує кількість процесів, які постійно перебувають у пам'яті. Для поліпшення продуктивності та стабільності диспетчер пам'яті операційної системи часто має механізм, який визначає, які регіони пам’яті перебувають на сторінках, а не ті, які повинні залишатися постійними. Менеджер пам’яті зазвичай відстежує, які сторінки є резидентними, а яких немає в попередньо обговорених даних підкачки конструкцій. Якщо потік намагається отримати доступ до сторінки, яка не є резидентною, апаратне забезпечення генерує помилку сторінки. Поки апаратне забезпечення генерує помилку сторінки, операційна система використовує інформацію про стан, закодовану в структурах підкачки, щоб

визначити, як обробляти несправність. Наприклад, сторінка може бути пов’язана з областю файлу, яка не була завантажена в пам’ять, або сторінка може бути переміщена до файлу сторінки. Підкачка сторінок попиту забезпечує суттєві переваги для операційної системи та прозорість для запущених програм.

За певних обставин можна поєднати дані, що не перебувають у пам'яті, знайдені на диску, з даними, що зберігаються в пам'яті, щоб забезпечити більш повний вигляд віртуальної пам'яті.

РОЗДІЛ №2 ОГЛЯД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЙОГО НАЛАШТУВАННЯ ТА МЕТОДИНІ РЕКОМЕДАЦІЇ ЩО ДО ВЕДЕННЯ РОСЛІДУВАННЯ ТА ВИБОРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

2.1 Отримання знімків пам’яті.

Збір пам’яті передбачає копіювання вмісту енергонезалежної пам'яті в енергонезалежне сховище. Це, один з найважливіших і найнестійкіших кроків у процесі криміналістики пам’яті. На жаль, багато аналітиків сліпо довіряють інструментам збору дампів оперативної памяті, не зупиняючись, щоб розглянути, як ці інструменти працюють, або типи проблем, з якими вони можуть зіткнутися. В результаті вони отримують пошкоджені зображення пам'яті, знищені докази та обмежені можливості аналізу, якщо такі є. Незважаючи на те, що цей розділ присвячений придбанню пам'яті Windows, багато концепцій стосуються інших операційних систем[4].

Збереження цифрового середовища. Хоча основна увага в цьому розділі приділяється аналізу даних, що зберігаються в енергонезалежній пам’яті, успіх цього аналізу часто з самого початку залежить від етапу отримання розслідування. На цьому етапі слідчий повинен прийняти важливі рішення щодо того, які дані збирати та найкращий метод збору цих даних. По суті, збір пам'яті - це процедура копіювання вмісту фізичної пам'яті на інший пристрій зберігання для збереження. У цьому розділі висвітлюються важливі проблеми, пов'язані з доступом до даних, що зберігаються у фізичній пам'яті, та міркування, пов'язані із записом даних до місця призначення. Конкретні методи та засоби, якими користуватись, часто залежать від цілей розслідування та особливостей системи, яку потрібно дослідити. Цифровий дослідник повинен прагнути зберегти стан цифрового середовища таким чином, щоб дозволити слідчому дійти до надійних висновків шляхом аналізу. Дані, що зберігаються на диску та в оперативній пам’яті, забезпечують два найважливіші компоненти цього середовища. Традиційні погляди цифрового розслідування фокусувались на припущенні, що надійність висновків цілком залежала від отримання доказів без зміни їх стану. Наприклад, загальновизнані

процедури обробки доказів передбачали вимкнення системи та створення дубліката копії (зображення) даних на дисковому накопичувачі для аналізу в режимі офлайн. Ці процеси та процедури збору були спрямовані на мінімізацію спотворень даних файлової системи за рахунок знищення інших джерел даних (наприклад, оперативної пам'яті, пам'яті пристрою), які також допомагають зафіксувати цифрове середовище[4].

У міру розвитку сфери цифрового розслідування стало очевидним, що вибіркове збереження деяких доказів за рахунок інших не менш важливих доказів може також вплинути на надійність отриманих висновків. Це особливо важливо, коли зловмисні актори намагаються використати обмеження традиційних цифрових криміналістичних методів збору доказів. Порівнюючи дані з кількох джерел (диск, мережа, пам’ять тощо) у цифровому середовищі, ви часто можете краще зрозуміти, що сталося в системі, ніж обмежена перспектива, яку надає вміст дискового сховища окремо. Для того, щоб включити ці альтернативні джерела, ви повинні прийняти, що всі методи збору, включаючи традиційні процедури збору дисків, призведуть до певного спотворення цифрового середовища. Слідчі повинні знати, як ці спотворення можуть вплинути на їх аналіз, і порядок збирання даних для зменшення цього впливу. Курс дій часто визначається пріоритетним, виходячи з порядку зменшення волатильності (тобто докази того, що зміни відбуваються швидше, отримуються до доказів, які є більш стабільними). З практичної точки зору це означає, що спочатку слід отримати свідчення про нестабільну пам’ять. Наприклад, у більшості ситуацій ви не можете створити загальноприйняте уявлення про "образ" фізичної пам'яті, якщо стан роботи машини змінюється під час придбання пам'яті. Більш відповідний опис "пам'яті зображень" - це вибірка стану фізичної пам'яті в даний момент часу. Теоретично, можливо, можна назвати набуття дискретних одиниць пам'яті (сторінок) як "зображення" цих сторінок, але стан фізичної пам'яті в цілому не може бути безпосередньо виміряний, і його слід виводити із стану цих окремих зразків . Хоча процес отримання зразка фізичної пам'яті може додати на етапі збору,

додаткову інформацію, яку він дає, може призвести до більшої впевненості в аналізі слідчого та менших спотворень фактів розслідування[4].

Якщо цільова система - «голий метал», наприклад, ноутбук, настільний комп'ютер або сервер, вам слід визначити, чи працює вона в даний час (потрібно памятати що не завжди є прямий доступ до цольової системи). Якщо він переходить у сплячий режим або вимикається, поточний стан пам’яті мінливий. Але, у багатьох випадках, останні мінливі дані могли бути записані на більш постійні пристрої зберігання, такі як жорсткий диск. Ці альтернативні джерела даних включають файли сплячого режиму, файли сторінок та дампи аварійного завершення роботи. Отримання пам'яті з енергонезалежних джерел передбачає завантаження цільової системи (систем) живим CD / DVD / USB для доступу до її диска або створення криміналістичного копіювання образу диска та встановлення його (лише для читання) з робочої станції аналізу.

Працююча система дає можливість дізнатися про стан енергозалежної пам'яті, але вам будуть потрібні права адміністратора. Якщо підозрюваний або жертва вже увійшли в систему з обліковим записом адміністратора або співпрацюють з вашим розслідуванням (наприклад, надають належні облікові дані), вам пощастило. У вас також може бути доступ адміністратора через ваший статус слідчого (облікові дані були надані вам підтримуючою корпорацією). У цих сценаріях можна використовувати службову програму, яка буде описаниа. В інших випадках вибір стратегії не такі однозначні. Можливо, отримати доступ адміністратора можна за допомогою експлойта для підвищення привілеїв (наступальна тактика, яка може призвести до недійсності криміналістичної достовірності ваших доказів) або шляхом підбору пароля методом грубої сили[4].

Інший варіант - збір даних з апаратною підтримкою. В цьому випадку не потрібні облікові дані для цільової системи (систем) - досить фізичного доступу. В цьому випадку потрібно покладатися на прямий доступ до пам'яті (DMA) з використанням таких технологій, як Firewire, Thunderbolt, ExpressCard або PCI. Зворотною стороною цього методу є те, що якщо цільова

машина не оснащена пристроєм для зняття даних, вам необхідно вимкнути цільову систему, щоб встановити необхідні апаратні адаптери (і це призведе до знищення важливих данних). До інших недоліків можна віднести той факт, що Firewire дозволяє отримати тільки перші 4 ГБ ОЗУ, що може серйозно обмежити успіх для систем з великим обємом пам'ятті. Крім того, пристрої PCI для придбання пам'яті вкрай рідкісні, тому вони досить дорогі. Фактично, в даний час існує тільки один продукт, який є у продажу і коштує близько 8000 доларів США (WindowsSCOPE)[5].

Коли фактори приведуть до придбання на основі програмного забезпечення, все одно доведеться прийняти безліч рішень. Потрібно звернути увагу на наступні моменти:

- Віддалений доступ проти локального достопу: чи маєте ми фізичний доступ до цільових систем? Справа може швидко ускладнитися, якщо комп'ютер знаходиться в іншій державі чи країні. Крім того, ви можете зіткнутися з ситуаціями коли метою є сервер без підключеної клавіатури або монітора, що знаходиться в центрі безпеки або мережевих операцій. .У цих ситуаціях віддалене придбання (через мережу) може бути вашим єдиним варіантом.
- Вартість: Чи є бюджетні обмеження на програмне забезпечення для придбання, яке можна придбати? Очевидно, обмеження вплине на те, які інструменти вам доступні.
- Формат: Чи потрібні дампи пам’яті у певному форматі файлу? Однак завжди можна конвертувати файли між форматами, якщо спочатку захоплений дамп пам'яті несумісний із нашими вимогами або засобами для аналізу.
- Інтерфейс командногї строки чи графічний інтерфейс: чи віддається перевага інструментам без графічного інтерфейсу чи програмам з наявним графічним інтерфейсом користувача (GUI)? Існує поширена думка, що інструменти з графічним інтерфейсом користувача залишають більший слід на цілі і мають велику поверхню атаки, тому вони погано підходять для криміналістичних досліджень. Однак добре написаний графічний інтерфейс

виробляє набагато менше шуму, ніж погано написаний інструмент командного рядка. Крім того, у вас може не бути доступу до консолі або служб віртуальних мережевих обчислень (VNC) / протоколу віддаленого робочого столу (RDP), на яких можна запускати додатки з графічним діінтерфейсом.

- Збір даних в порівнянні з опитуванням під час виконання: чи потрібен вам повний дамп фізичної пам'яті або просто можливість визначати запущені процеси, мережеві з'єднання і т. Д? Вам потрібно постійно опитувати систему на предмет змін? У корпоративних середовищах з сотнями або тисячами систем недоцільно отримувати повні дампи пам'яті тільки для перевірки наявності певного індикатора. Замість цього ви можете виконати швидке очищення всього підприємства, перевіривши кілька конкретних областей пам'яті на кожній машині[6].

Часте питання - який інструмент я повинен використовувати для отримання зліпків пам’яті? Насправді немає однозначної відповіді. Правильний інструмент для конкретної роботи залежить від особливостей справи. Визначившись із особливостями справи, ви можете зосередитись на виборі інструменту, який найкраще підтримує ваші цілі. Далі в цьому розділі ми представляємо деякі конкретні запобіжні заходи, які слід дотримуватися під час здійснення придбань[6].

2.3 Ризики які виникають під час збору знімків памяті.

Перш ніж отримати знімок фізичної пам'яті з підозрілої системи, потрібно завжди враховувати пов'язаний з цим ризик. Оскільки більшість ОС не надають підтримуваний власний механізм для отримання фізичної пам'яті, ви будете використовувати систему таким чином, щоб вона могла виявитися в непередбаченому стані. Крім того, система з погано написаним шкідливим ПЗ може бути нестабільною і вести себе непередбачувано. Рішення про отримання знімків фізичної пам'яті вимагає, щоб ви врівноважували вигоду отримання даних з властивим (-ими) ризиком (ами), пов'язаних з процедурою отримання. Наприклад, якщо метою є критично важлива система, яку можна вимкнути або перезавантажити тільки в екстремальних обставинах, ви повинні

бути готові обґрунтувати, чому отримання пам'яті життєво важливо для вашого дослідження. Можуть бути навіть обставини, при яких наслідки (наприклад, смерть, шкода навколишньому середовищу) дестабілізації системи ніколи не стоять ризику[6].

Далі описані деякі з основних причин, по яким захоплення пам'яті може призвести до нестабільності системи і пошкодження важливих даних. Важливо відзначити, в наступних прикладах використовується Microsoft Windows, проблеми не є специфічними для будь-якої ОС - вони є функціями процесора і архітектури обладнання.

Пам'ять пристрою. Фізична пам’ять - це логічна схема адресації, яка дозволяє рівномірний доступ (або “адресу”) до різних ресурсів материнської плати. На комп'ютерах на базі x86 / x64 мікропрограма (BIOS) надає ОС фізичну карту пам'яті з різними регіонами, позначеними як зарезервовані для використання прошивкою, шинами ISA або PCI або різними пристроями материнської плати. Ці регіони називаються областями пам'яті пристрою. На малюнку 2.1 показано спрощену схему розміщення фізичної пам’яті, якщо врахувати ці „діри”, що існують завдяки областям пам’яті пристрою. У 32-розрядних системах з обсягом пам'яті менше 4 ГБ «дірки» спричиняють загальний обсяг пам'яті, доступний для ОС, дещо менший, ніж рекламована ємність чіпів оперативної пам'яті[7].

Ненавмисне читання з одного з цих зарезервованих регіонів може бути небезпечним. Залежно від виду пристрою, до якого ви отримуєте доступ, читання з фізичної адреси в регіоні може отримати дані, що зберігаються в цьому місці, або змінити стан пристрою, до якого ви отримуєте доступ. Наприклад, існують фізичні адреси, які відображаються в регістрах пристроїв, які змінюють стан пристрою кожного разу, коли зчитується фізичне місцезнаходження. Ця зміна може заплутати драйвери пристроїв або мікропрограму, які залежать від значень у цих регістрах, що в кінцевому підсумку призведе до зависання системи. Це зависання виникає особливо часто, коли зчитуються адреси, зайняті відеочипом, високоточним таймером

подій (HPET) або неясними, застарілими пристроями PCI. Додатковою проблемою є те, що більшість з цих пристроїв не призначені для одночасного доступу більше ніж одного процесора одночасно.

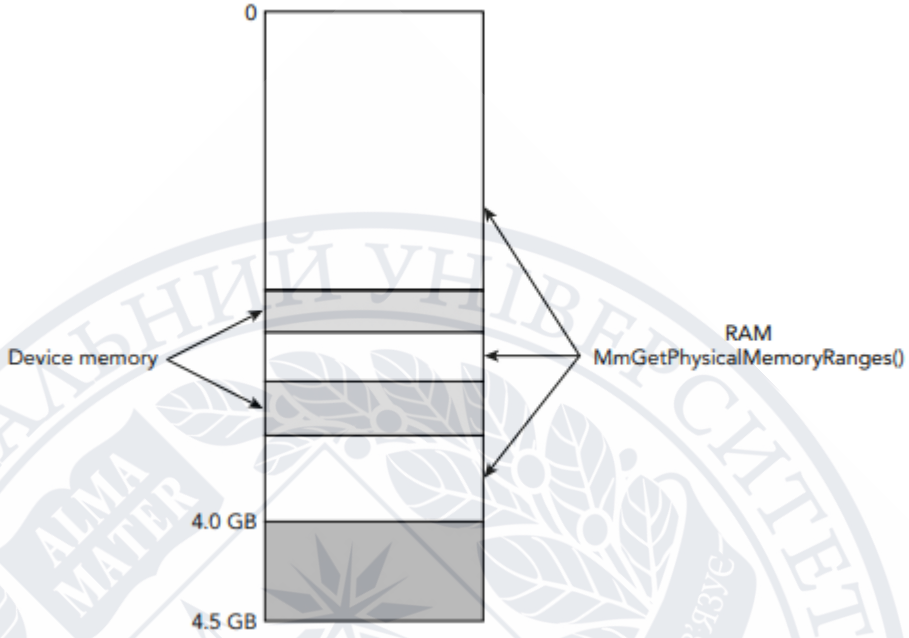

Рисунок 2.1: Макет фізичної пам'яті для сумісної системи x86 / x64 показує різні діри через області пам'яті пристрою.

Незважаючи на ризик, пов'язаний із збором областей пам'яті пристроїв, це може дати докази з високою судово-медичною цінністю. Наприклад, дані в цих регіонах можуть містити векторну таблицю переривань у реальному режимі (IVT) з артефактами, залишеними руткітами на основі прошивки.Також можна знайти докази руткітів BIOS, які вводять код у верхній частині пам'яті реального режиму. Крім того, ви можете використовувати область CMOS для зміни порядку завантаження та регістрів непрямого доступу IOAPIC для перенаправлення переривань. З програмних засобів збору пам'яті, про які буде йти мова далі, лише KnTDD від GMG Systems може зібрати ці регіони з хорошим рівнем надійності та точності[8].

Когерентність кешу. Сучасні процесори розроблені з одним або декількома кешами внутрішньої пам'яті для поліпшення продуктивності. Запис таблиці сторінок може бути запрограмований з різними атрибутами кеш-пам'яті (некешованими, кешованими, комбінованими), які визначають

спосіб доступу процесора до сторінки фізичної пам'яті. Ці процесори мають задокументовані конструктивні обмеження:

Вони не призначені для одночасного відображення однієї і тієї ж фізичної адреси з кількома атрибутами кешу. Це може призвести до невизначеної поведінки процесора, включаючи, але не обмежуючись, пошкодження буфера перекладу (TLB) та пошкодження даних за вказаною адресою пам'яті. Таким чином, погано написаний інструмент придбання може легко зробити недійсною саму пам'ять, що набувається[8].

2.3 Коли створювати знімок пам’яті?

Вибір належного часу для збору доказів фізичної пам’яті залежить від ряду факторів. Наступні пункти - це лише пропозиції, а не правила. Наприклад, якщо збираються докази з комп’ютера підозрюваного, можливо спланувати створення дампу таким чином, щоб підозрюваний в цей час знаходився в Інтернеті (або, принаймні, ввійшов у систему), що може надати доступ до сеансу входу підозрюваного, інформації про хмарні послуги або віддалене сховище, до якого він може мати доступ, і будь-які зашифровані документи, які підозрюваний міг переглядати. З іншого боку, якщо збір доказів з комп’ютера жертви, можливо краже призначити час збору знімків пам'яті тоді коли підозрюваний не активний, щоб уникнути небажаної підозри. Збір даних мережевих сеансів з комп’ютера жертви та з нього за деякий час до придбання може допомогти зробити це.

Потрібно пам'ятати, що в більшості випадків збір доказів відбувається із запущеної комп’ютерної системи. Збільшення кількості змін, що відбуваються під час збору знімків пам’яті, може збільшити кількість аномалій, з якими спеціаліст з аналіку шкідливого програмного забезпеченя зіткнеться.

Якщо це можливо, потрібно уникайти збору доказів пам’яті під час періодів кардинальних змін, наприклад, під час запуску, вимкнення системи або під час виконання завдань з обслуговування системи (тобто дефрагментація диска, повне сканування вірусів, резервне копіювання

системи). Також рекомендується обмежити взаємодію з машиною до завершення збору дампів.

2.4 Як проводити збір знімків пам’яті.

Після того, як сприятливий час для збору дампів пам’яті був визначений, потрібно розглянути низку важливих запобіжних заходів. Цей розділ зосереджений на найпоширеніших випадках використання та пов’язаних із ними застереженнях, які можуть швидко перерости у серйозні наслідки, якщо не бути достатньо обережним[9].

Локальний збір дампів на зовнішні носії. У цьому випадку дампи пам’яті переносяться на зовнішній диск USB, ESATA або Firewire, підключений до цільової системи. Ніколи не рекомендується зберігати дампи пам’яті на локальні диски цільової системи, такі як розділ C:, оскільки це перезапише значну кількість доказів, які можуть мати значення для вашої. Через розмір оперативної пам’яті на сучасних комп’ютерах потрібно переконатись, що цільовий диск відформатований за допомогою NTFS або іншої високопродуктивної файлової системи. Також пам’ятайте, що зловмисне програмне забезпечення часто поширюється шляхом зараження зовнішніх носіїв. Наступні пункти містять додаткові поради щодо локального збору дампів пам'яті.

- Ніколи не приєднуйте один і той же знімний носій до кількох можливо заражених комп'ютерів, щоб уникнути поширення інфекції під час збору доказів.
- Не підключайте можливий інфікований знімний носій безпосередньо до вашої робочої станції. Підключіть зовнішній носій пам’яті до проміжної “жертовної” системи та огляньте її. Потім скопіюйте докази на основну робочу станцію через невелику ізольовану мережу (наприклад, через концентратор або некерований комутатор).
- Завжди «стерилізуйте» (надійно стирайте) знімний носій, перш ніж використовувати (або повторно використовувати) його для збору доказів.

Віддалений збір дампів оперативної пам'яті. У примітивному сценарії віддаленого збору зазвичай надсилаються інструменти через мережу до цільової машини через PsExec або скопіювавши їх у спільний доступ C$ або ADMIN$ за допомогою блоку повідомлень сервера (SMB). Потім можа запланувати завдання або встановити службу в цільовій системі, яка запускає інструмент (інструменти) і надсилає вміст фізичної пам’яті через прослуховувач netcat або інший протокол зворотного зв'язку. Основними проблемами цього методу є розкриття облікових даних адміністратора та вміст оперативної пам’яті цільової системи, що надсилається звичайним текстом через мережу. Основна пам’ять комп’ютера містить багато конфіденційної інформації, яка може бути розкрита при отриманні у вигляді простого тексту через відкриту мережу[9].

У середовищі домену або підприємства, облікові дані адміністратора домену забезпечують зручний спосіб доступу до цільової системи; однак, якщо цільовий комп'ютер уже скомпрометований, зловмисники можуть відновити згенеровані маркери автентифікації з пам'яті для використання. Кращим рішенням є створення тимчасового облікового запису адміністратора, який дозволяє лише доступ до цільової системи. Після чого тимчасовий обліковий запис адміністратора після завершення збору дампів можна видалити. Також є можливість блокування з'єднань із цільової машини в брандмауері або маршрутизаторі (за винятком систем, що беруть участь у віддаленому зборі дампів). Це заважає будь-якому зловмисному програмному забезпеченню або зловмисникам використовувати вкрадені облікові дані для подальшого проникнення в мережу. Збіл мережевого ресурсу слід використовувати лише в крайньому випадку, але в деяких обмежених випадках це може бути необхідним. Починаючи з SMB 3.0 (Windows Server 2012), підтримується наскрізне шифрування. Крім того, деякі інструменти (наприклад, CryptCat, KnTDD, F-Response Enterprise)[9] підтримують отримання доказів по мережі за допомогою SSL / TLS. Таакож можна розглянути можливість стиснення доказів перед передачею їх по мережі, щоб зменшити необхідний час для їх

отримання. Не забувайте обчислювати хеші цілісності до і після передачі,щоб ваші докази не змінилися під час передачі.

Збір данних під час виконання.

Збір данних під час виконання дозволяє швидко пройтись по всьому підприємству та перевірити наявність певних показників у фізичній пам'яті (замість того, щоб фіксувати повний дамп пам'яті з кожної системи). Зазвичай цей тип аналізу виконується в автоматизованому режимі.

Різні комерційні набори забезпечують можливості збору дампів фізичної пам'яті на різних підприємствах, такі як F-Response, AccessData Enterprise та EnCase Enterprise.

Збір дампів пам’яті за допомогоюапаратних засобів.

Через обмеження, згаданих раніше, я не розглядаю докладно апаратні придбання. Однак варто знати, що Volatility підтримує збір та опитування пам'яті через Firewire. Вам знадобиться бібліотека libforensic1394, стек JuJu Firewire і спеціальний виклик Volatility. Зверніть увагу на параметр -l замість -f:

$ python vol.py –l firewire://forensic1394/<devnumber> plugin [options]

<Devnumber> - це номер пристрою (зазвичай 0, якщо підключеня тільки до одного пристрою Firewire). Використання плагіну imagecopy для отримання дампу пам'яті або будь-який інший плагін аналізу для опитування працюючої системи, але потрыбно пам'ятати про обмеження 4 ГБ, про який говорилося раніше.

Інший варіант використання апаратного аналізу пам'яті включає розблокування робочих станцій. Наприклад, інструмент Inception від Carsten Maartmann-Moe знаходить і виправляє інструкції, які дозволяють входити в захищені паролем комп'ютери з Windows, Linux і Mac OS X навіть без облікових даних. Однак, як зазначено на веб-сайті інструменту, якщо необхідні інструкції не знайдені в нижніх 4 ГБ пам'яті, він може працювати ненадійно.

2.5 Програмні засоби.

Усі програмні засоби збору даних дотримуються подібного протоколу для отримання пам'яті. Зокрема, ці інструменти працюють, завантажуючи модуль ядра, який відображає потрібні фізичні адреси у віртуальний адресний простір завдання, що виконується в системі. На даний момент вони можуть отримати доступ до даних із віртуального адресного простору та записати їх у запитуване енергонезалежне сховище(ОЗП). Програмне забезпечення для придбання має два способи зробити це відображення віртуальної фізичної адреси:

- Підхід, який використовують більшість, якщо не всі комерційно доступні інструменти, включає використання API операційної системи для створення запису в таблиці сторінок. До типових функцій відносяться: ZwMapViewOfSection (on \ Device \ PhysicalMemory), MmMapIoSpace, MmMapLockedPagesSpecifyCache і MmMapMemoryDumpMdl.
- Другий можливий підхід використовує інші API ОС для виділення порожнього запису таблиці сторінок і ручного кодування бажаної фізичної сторінки в запис таблиці сторінок.

2.6 Оцінка інструменту.

На відміну від інструментів для створення образів дисків, які не були розроблені формальні специфікації і не проводилися оцінки інструментів для отримання пам'яті. Фактично, це все ще область відкритих досліджень і тема для суперечок. Одна з основних проблем при оцінці засобів отримання знімків пам'яті полягає в тому, що вони можуть працювати по-різному в залежності від версії ОС, конфігурації операційної системи і встановленого обладнання. Також важливо підкреслити, що платформи віртуалізації часто набагато більш передбачувані і однорідні, ніж реальне обладнання, а це означає, що вони можуть не дати хорошого індикатора того, як інструмент буде працювати в неоднорідному середовищі реального обладнання.

З операційної точки зору, основні атрибути надійного інструменту криміналістичної експертизи полягають в тому, що він повинен отримувати докази точним, повним, документованим способом і з надійною реєстрацією помилок. Основна проблема багатьох сучасних інструментів збору даних полягає в тому, що, коли вони виходять з ладу (або повністю, або при читанні однієї або декількох сторінок), вони часто не виводять ніяких помилок. Ця проблема не дозволяє дослідникам зрозуміти, що проблема існує, поки вони не досягнуть фази аналізу , після чого часто буває занадто пізно повернутися і отримати інший образ пам'яті з цільової системи. Жоден метод збору доказів не застрахованй від помилок. Однак, якщо інструмент збору даних надійно реєструє помилки, тоді аналітик може вирішити, як вчинити з помилкою. З оперативної точки зору або з точки зору доказів найгірший сценарій - не знати, що у вас є, після того, як ви завершили збір доказів.

Також потрібно мати на увазі, що те, що інструмент не призвів до збою або зависання цільової машини, не означає, що він створив точний і повний "образ" пам'яті. У системах Microsoft Windows ви можете використовувати засіб перевірки драйверів Microsoft, щоб визначити, проявив постачальник інструменту розумну обережність при розробці інструменту. Крім цього, найкраща лінія захисту - це тестування інструменту на системах, які схожі (наскільки це можливо) на системи, на яких ви будете їх використовувати в польових умовах. На жаль, це не завжди можливо через різні комбінації існуючого обладнання та програмного забезпечення.

2.7 Вибір інструмента.

У наступному списку в показані часто використовувані інструменти створення знімків пам'яті також запропоновано блок схуму для вибору програмного забезпечення в залежності від можливостей, ситуації, виду та стану системи.

- GMG Systems, Inc., KnTTools: основні функції цього інструменту включають модулі віддаленого розгортання, перевірки криптографічної

цілісності, збір доказів через SSL, стиснене виведення, додаткове регулювання смуги пропускання, автоматичний збір даних про стан користувача в реальному часі для перехресних посилань, захоплення файлів підкачки , надійна реєстрація помилок, а також ретельне тестування і документація. Він також може отримувати ROM / EEPROM / NVRAM з BIOS і пам'яті периферійних пристроїв (PCI, відеокарта, мережевий адаптер).

- F-Response: набір продуктів від F-Response представив революційну нову можливість в криміналістиці пам'яті - можливість опитувати живі системи з віддаленого місця через з'єднання iSCSI тільки для читання. F-Response представляє собою незалежний від виробника і ОС вид фізичної пам'яті і жорстких дисків цільової системи, що означає, що ви можете отримати до них доступ з аналітичних станцій Windows, Mac OS X або Linux і обробити їх за допомогою будь-якого інструменту.
- Mandiant Memoryze: інструмент, який можна легко запустити зі знімного носія і який підтримує завантаження з найпопулярніших версій Microsoft Windows. Ви можете імпортувати XML-висновок Memoryze в Mandiant Redline для графічного аналізу даних у фізичній пам'яті.
- HBGary FastDump: інструмент, який займає мінімально можливий розмір надає можливість збирати файли підкачки і фізичну пам'ять в один вихідний файл (HPAK), а також здатність перевіряти пам'ять процесу (потенційно інвазивна операція, яка змушує замінені сторінки зчитувати назад в ОЗУ перед збиранням даних).
- MoonSols Windows Memory Toolkit: Сімейство MWMT включає win32dd, win64dd і саму останню версію DumpIt - утиліту, яка об'єднує 32- і 64-розрядні інструменти отримання дампа пам'яті в виконуваний файл, для роботи якого потрібен один клік. Ніякої подальшої взаємодії не потрібно. Однак, якщо вам потрібні більш просунуті параметри, такі як вибір між типами вихідних форматів, включення шифрування RC4 або створення сценаріїв для виконання на кількох машинах, ви також можете це зробити.

- AccessData FTK Imager: цей інструмент підтримує збір багатьох типів даних, включаючи RAM. AccessData також продає попередньо сконфігурованих набір USB-інструментів для збору даних в реальному часі, який крім журналів чатів, здійснювання підключення до мережі і т.д. Отримує знімок оперативної пам’яті.
- EnCase / WinEn: інструмент збору даних від Guidance Software може вивантажувати пам'ять в стислому форматі і записувати метадані в заголовки (наприклад, назва справи, аналітика і т. Д.). Версія EnCase для підприємств використовує аналогічний код в своєму агентові, який дозволяє віддалено опитувати працюючі системи.
- Belkasoft Live RAM Capturer: Утиліта, яка має можливість дампити пам'ять навіть при наявності агресивних анти-налагоджувальних і антидампових механізмів. Він підтримує всі основні 32- і 64-розрядні версії Windows і може запускатися з USB-накопичувача.
- ATC-NY Windows Memory Reader: цей інструмент може зберігати пам'ять в форматі необробленого або аварійного дампа і включає в себе безліч опцій хеширования цілісності. При використанні з UNIX-подібної середовища, такий як MinGW або Cygwin, ви можете легко відправити висновок віддаленого приймача netcat або через зашифрований тунель SSH.
- Winpmem: єдиний інструмент для збору пам'яті з відкритим вихідним кодом для Windows. Він включає в себе можливість виведення файлів у форматі сирого або аварійного дампа, вибір між різними методами збору даних (включаючи експериментальний метод перепризначення PTE) і можливість збору дампів фізичної пам'яті через пристрій для аналізу локальної системи в реальному часі.

Як вибрати інструмен якщо вище зазначені інструменти не підходять або не має можливості використовувати їх?

Запропонована схема рис 2.3 допоможе вибрати програмне чи апаратне забезпечення в залежності від ряду факторів:

- Вид системи

- Стан системи
- Рівень доступу до системи
- Фінансові ресурси
- Наявність або відсутність прямого доступу.

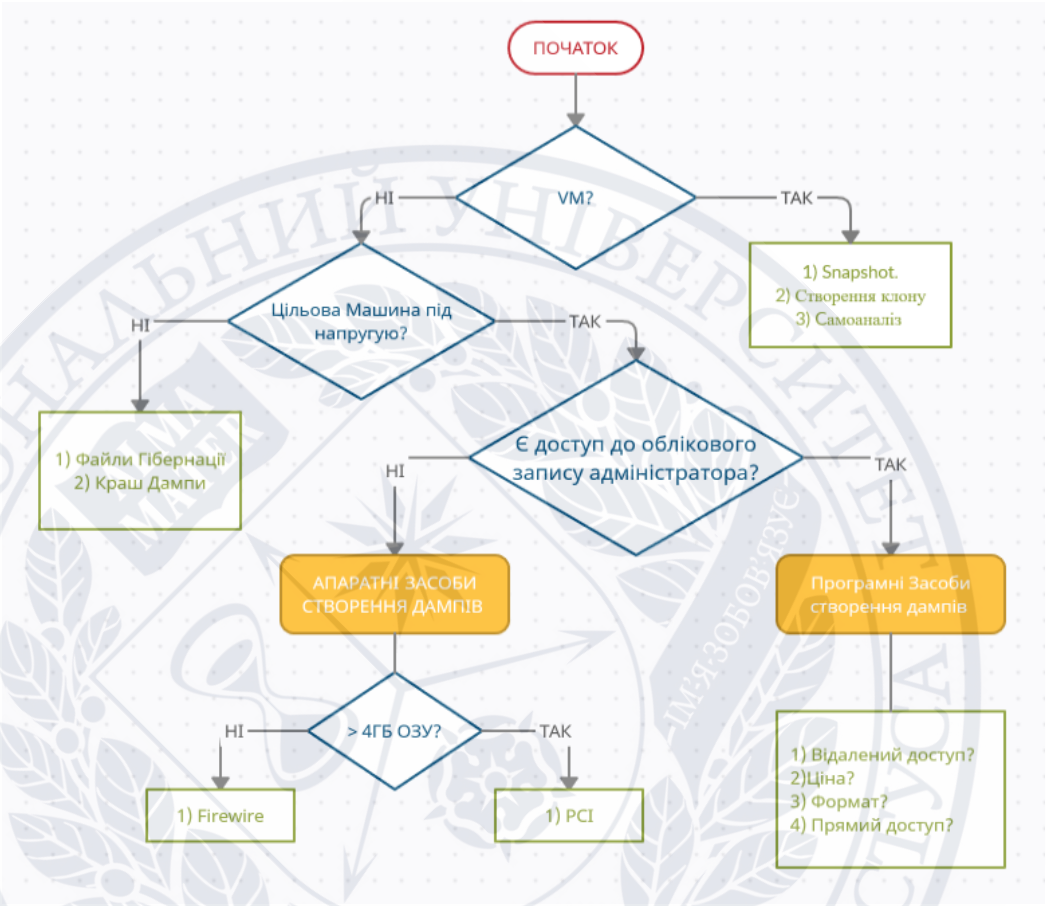

Рис 2.3 Діаграма початкових факторів, яку потрібно врахувати перед збором дампів оперативної пам’яті

2.8 Приклад з KnTDD в дії.

Наступні кроки показують, як розгорнути KnTTools з USB носія. Зокрема, я продемонструю отримання даму фізичної пам'яті і файлу-ів підкачки в зашифрованому стислому форматі.

1. Створення цифрового сертифікату X509 / PKCS #7. Для створення можна використовувати makecert.exe, інструмент, який поширюється з Microsoft Visual Studio та іншими пакетами розробки програмного забезпечення Microsoft, включаючи безкоштовну Visual Studio Express. Наступна команда створює самопыдписний сертифікат з ім'ям *mandryk.o.cer:*

```
C:\Mandryk>makecert.exe -r -pe -n "CN=mandryk.o@useonce.mnin.org"
    -sky exchange -ss my -a sha1 -len 2048
    -b 05/05/2021 -e 05/05/2021 mandryk.o.cer
Succeeded
```

2. Запуск KnTDD з USB носія (G: в даному випадку), як тільки він буде підключений до цільової системі. Параметри запитують хеші SHA512, шифрування за допомогою самопідписного сертифіката, стиснення LZNT1 і збір файлів підкачки. Kntdd.exe був перейменований в covert.exe. Перейменування запобігає простим атакам анти-криміналістичної експертизи, засновані на назвах поширених інструментів збору інформації. Крім того, рандомізують імена пристроїв і символічних посилань.

```
G:\deploy\covert.amd64>covert.exe -v -o memory.bin --log
   --cryptsum sha_512
   --pagefiles --force_pagefiles --4gplus
   --comp lznt1 --cert michael.ligh.cer --case case001
KnTTools, 2.3.0.2898
kntdd physical memory forensic acquisition utility, 2.3.0.2898
Copyright (C) 2006-2013 GMG Systems, Inc.
Current User: WIN-948O8I1DO91\Mandryk
Fully-qualified DNS Name: WIN-948O8I1DO91
NetBIOS Name: WIN-948O8I1DO91
Fully-qualified physical DNS Name: WIN-948O8I1DO91
Uptime: 0 days 7:39:4.987
Physical memory modules installed on the system: 0x80000000
Physical memory visible to the operating system: 0x7ff7e000
Highest physical page plus 4096 bytes: 0x80000000
PAE is enabled!
[snip]
Installing driver covertdrv from image path
       G:\deploy\covert.amd64\amd64\covertdrv.sys
```

```
Starting driver as \Device\LNk2OwGwN
Driver symbolic link name: \DosDevices\XPBM8vLS
Driver started.
Binding to symbolic link \\.\Vpa1CSE6zN
Symbolic link is bound.
Total bytes copied: 0x80000000
The loaded OS kernel name is C:\Windows\system32\ntoskrnl.exe
[snip]
Reading standard Cmos data...Ok!
Reading standard IOAPIC data...Ok!
Reading disk boot sector...
ArcPath: multi(0)disk(0)rdisk(0)partition(1)->\\.\PhysicalDrive0OK!
Reading volume boot records...
ArcPath: multi(0)disk(0)rdisk(0)partition(1)->\\.\PhysicalDrive0
OK:  1 volume boot records read.
[snip]
Copying pagefiles...
C:\pagefile.sys
    Pid: 0x4
    Handle: 0xffffffff800001e0
    Object: 0xfffffa8002ca7390
    LogicalFileSize: 0x72d33000
    Directory: No
    Compressed: No
    Encrypted: No
    PagingFile: Yes
    Start: 0x0 ExtentSize: 0x72d33000 LogicalOffset: 0x251508000
Copying pagefiles completed successfully.
[snip]
```

3. Далі з міркувать безпеки потрібно перенести отримані докази на робочу станцію криміналістичної експертизи, використовуючи передові методи, описані в розділі «Віддалене створення дампів оперативної памяті.». Перерахування каталогу доказів показує стислий і зашифрований (розширення lznt1.kpg) дамп фізичної пам'яті, файл(и) підкачки, журнал збору даних, стан користувальницької системи і хеші. Важливі файли ОС (ntoskrnl.exe, tcpip.sys, ndis.sys і т.д) також збираються для подальшого аналізу - вони з'являться в підкаталозі WINDOWS.

```
C:\Mandryk>dir "{DC04DB43-AC21-4060-8954-17D0AD3166DC}"
 Directory of C:\Analyst\{DC04DB43-AC21-4060-8954-
17D0AD3166DC}05/06/2014 09:14 PM <DIR> .
 5/05/2021 09:14 PM <DIR> ..
 05/05/2021 09:14 PM 1,524 memory.bin.dumpheader.lznt1.kpg
 05/05/2021 09:14 PM 933,585,748 memory.bin.lznt1.kpg
 05/05/2021 09:14 PM 4,116 memory.log.lznt1.kpg
 05/05/2021 09:14 PM 51,067,636 memory.Pagedump_C!!
pagefile!sys.DMP.lznt1.kpg
 05/05/2021 09:14 PM 293,348 memory.user_system_state.xml.lznt1.kpg
 05/05/2021 09:14 PM 4,068 memory.xml.lznt1.kpg
 05/05/2021 09:14 PM <DIR> WINDOWS
 6 File(s) 984,956,440 bytes
 3 Dir(s) 901,435,392 bytes free
```

4. Використання службової програми kntencrypt.exe для розшифровки файлів за допомого сертифіката. Ця команда створює нові файли з розширенням *lznt1*, *щоб вказати, що вони все ще стиснуті.*

```
C:\Mandryk\KnTTools.amd64>kntencrypt.exe --cert mandryk.o.cer
 -d -v "{DC04DB43-AC21-4060-8954-17D0AD3166DC}/*"
KnTTools, 2.3.0.2898
KnTTools encryption utility., 2.3.0.2898
Copyright (C) 2006-2013 GMG Systems, Inc.
```

```
07/05/2014  02:31:41 (UTC)
06/05/2014  21:31:41 (local time)
Current User: WIN-948O8I1DO91\Mandryk
C:\Analyst\KnTTools.amd64\{DC04DB43-AC21-4060-8954-
17D0AD3166DC}\memory.bin.dumpheader.lznt1.kpg--
>C:\Analyst\KnTTools.amd64\{DC04DB43-AC21-4060-8954-
17D0AD3166DC}\memory.bin.dumpheader.lznt1...
OK.percent complet
C:\Analyst\KnTTools.amd64\{DC04DB43-AC21-4060-8954-
17D0AD3166DC}\memory.bin.lznt1.kpg--
>C:\Analyst\KnTTools.amd64\{DC04DB43-AC21-4060-8954-
17D0AD3166DC}\memory.bin.lznt1...
OK.percent complete
[snip]
```

5. Використовуйте спеціальну версію dd.exe, що надається разом з KnTTools, для розпакування файлів доказів у вихідний каталог. Ми також використовуємо опцію --sparse, щоб файли рекомпресували за допомогою стиснення файлів NTFS (що призводить до того, що файли доказів займають набагато менше місця на диску, і вони, як правило, швидше читаються).

```
C:\Mandryk\KnTTools.amd64>dd.exe -v
    if="{DC04DB43-AC21-4060-8954-17D0AD3166DC}/*.lznt1"
    of=decompressed\ --decomp lznt1 --sparse –localwrt
Forensic Acquisition Utilities, 1.4.0.2464
dd, 5.4.0.2464
Copyright (C) 2007-2013 GMG Systems, Inc.
C:\Mandryk\KnTTools.amd64\decompressed\memory.user_system_state.xml
327680/1008180 bytes (compressed/uncompressed)
0+1 records in
0+1 records out
1008180 bytes written
```

[snip]

На рис 3.8.1 показаний приклад каталогу доказів після розшифрування та декомпресії. Файл memory.bin є основним зразком пам'яті, а стан користувача збирається в memory.user_system_state.xml. Як показано на малюнку, цей XML-файл містить детальну інформацію про поточний стан машини з точки зору операційної системи (тобто її API). Ви можете проаналізувати XML, щоб порівняти дані з тим, що ви знайдете у зразку фізичної пам’яті (це робить KnTList).

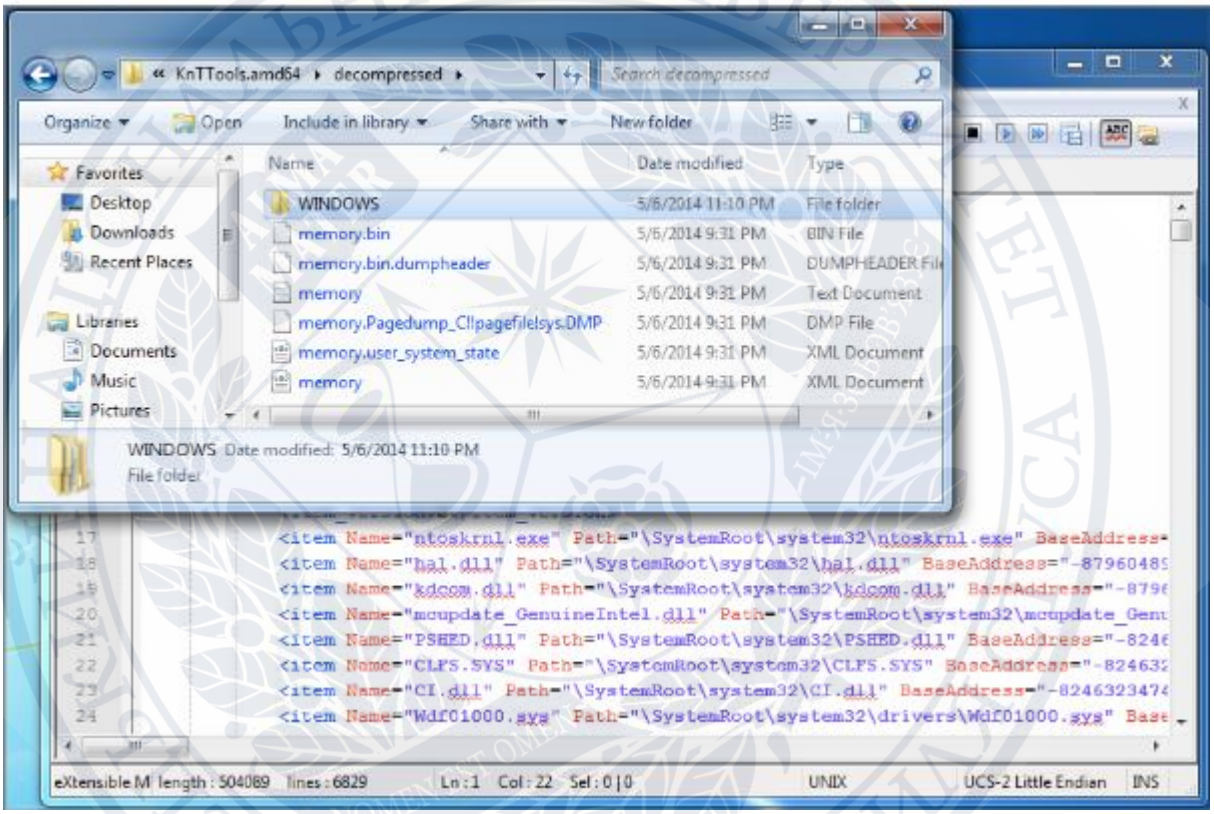

Рис: 2.2 Приклад доказів, зібраних KnTDD, включаючи фізичну пам’ять, файл сторінки, стан користувача XML та файли ядра ОС .

Віддалене отримання дампів оперативної пам'яті KnTDD.

Тут демонструється один з варіантів віддаленого збору даних, доступних з KnTDD. На стороні сервера (машина, яку ви будете використовувати для отримання даних) ми виконали наступну команду. Параметр -L прослуховує кілька підключень, оскільки зразок пам'яті передається через порт, відмінний від порту журналу, стану користувача XML і т. Д. Наш IP-адреса сервера 192.168.228.143 (збір через локальну підмережу).

```
F:\KnTTools.amd64> nc --verbose -L --port 8000
    --source 192.168.228.143
    -O memory.bin --localwrt
Forensic Acquisition Utilities, 3.4.0.2464
Netcat network data redirector., 1.11.2.2464
Copyright (C) 2002-2013 GMG Systems, Inc.
Windows 10 Pro 6.2.9200 Multiprocessor Free(, 9200.win8_rtm.120725-1247)
05/05/2021 19:21:52 PM (UTC)
05/05/2021 12:21:52 PM (local time)
Current User: ugly\Test
Current Locale: English_United States.437
User Default Locale Language: 0x0409
The VistaFirewall Firewall is active with exceptions.
Listening on TCP: ugly.localdomain: 8000.
```

У цільовій системі (тій, з якої ми отримуємо пам’ять) була використана наступна команда:

```
F:\KnTTools.amd64> kntdd.exe --verbose --out 192.168.228.143
    --iport 8000 --4gplus
    --cryptsum sha_512 --pagefiles --force_pagefiles
    --log --case "Case002"
    --comp lznt1
    --cert mandryk.o.cer
```

2.9Приклад з Linux Memory Extractor (LiME) в дії

В цьому розділі буде розглянуто ситуацію коли цільовою системою для розлідування є система під упарвлінням операційної системи LINUX.

Linux Memory Extractor, також відомий як LiME є новітнім інструментом для отримання пам'яті Linux. Він усуває проблемb пов'язані з раніше

обговореними інструментами і методами. LiME працює шляхом завантаження драйвера ядра, але замість створення символьного пристрою, до якого користувач може отримати доступ, всі збори даних виконуються всередині ядра. Це значно підвищує точність результуючого зразка, оскільки відсутня перемикання контексту між користувальницької середовищем і ядром, необхідне для передачі даних. LiME також покращує fmem, автоматично визначаючи діапазони адрес, які містять основну пам'ять. Щоб перерахувати діапазони, LiME переглядає пов'язаний список ядра iomem_resource. Цей список містить дескриптори для кожного сегмента фізичної пам'яті. Якщо ім'я сегмента збігається з ім'ям RAM (System RAM), відповідні початковий і кінцевий члени визначають, де у фізичній пам'яті знаходиться сегмент.

При захопленні пам'яті за допомогою LiME ви можете вибирати між кількома форматами збору даних. При використанні рекомендованого формату лайма створюється структурований файл, який усуває необхідність створення файлу з нулями. Цей структурований формат містить метадані, що описують фізичне усунення кожного розділу, що дозволяє інструменту аналізу пам'яті динамічно відновлювати вихідну структуру пам'яті. Кожен сегмент отриманої пам'яті починається з lime_header. Наступний висновок показує цю структуру метаданих:

```
>>> dt("lime_header")
'lime_header' (32 bytes)
0x0   : magic                  ['unsigned int']
0x4   : version                ['unsigned int']
0x8   : start                  ['unsigned long long']
0x10  : end                    ['unsigned long long']
0x18  : reversed               [‘unsigned long long']
```

Компіляція LiME

Щоб використовувати LiME, ви повинні спочатку скомпілювати модуль ядра для тієї версії ядра, яку ви хочете проаналізувати. Це докладно описано в

документації проекту. Після того, як ви скомпілювали модуль, у вас буде файл з ім'ям lime- <версія ядра> .ko, який ви можете завантажити в цільову систему.

Завантаження LiME і дамп пам'яті LiME має можливість вивантажувати пам'ять на локальний диск або по мережі, і це можна зробити в наступних форматах:

- raw: усі діапазони пам’яті об’єднані разом
- padded: Подібно до raw формату, за винятком того, що проміжки між діапазонами пам'яті заповнюються нулями.
- lime: Запис у згаданий формат LiME (рекомендується)

Ви можете контролювати місце призначення дампа пам'яті, задавши параметр шляху для модуля ядра. Щоб вивантажити вміст пам'яті в файл на пристрої зберігання, просто вкажіть path = / path / to / memdmp.lime. У цьому випадку пам'ять купується і записується в бажане місце відразу після завантаження модуля. Приклад отримання пам'яті на диск у форматі Lime показаний в наступній команді:

```
$ sudoinsmod lime.ko "path=/mnt/externaldrive/memdmp.lime format=lime"
```

Зверніть увагу, що в цій команді ми вказуємо шлях як / mnt / external, щоб вивантажити пам'ять на зовнішній диск. Ніколи, крім випадків крайньої необхідності, не слід зберігати свідоцтва на локальному диску, так як це призведе до перезапису нерозподіленого дискового простору, що може бути корисно при експертизі диска.

Для мережевого збору даних шлях задається як path = tcp: 4444 (фактичний порт можна вибрати любій з незарезервованих). Потім в цільовій системі і на зазначеному порту створюється прослуховує сокет. Ви можете підключитися до сокету за допомогою такого інструменту, як netcat, і збір даних почнеться, як тільки з'єднання буде встановлено. Нижче наводиться приклад отримання пам'яті по мережі за допомогою netcat. Перша команда запускається на машині, з якої ви хочете отримати пам'ять:

```
$ sudo insmod lime.ko "path=tcp:4444 format=lime"
```

Потім на своїй станції криміналістично експертизи можна використовувати netcat для отримання дампу (за умови, що 192.168.1.40 - це IP-адреса цільового комп'ютера, на якому встановлений LiME):

```
$ nc 192.168.1.40 4444 > memdmp.lime
```

2.10Приклад з Mac Memory Reader (MMR) в дії

В цьому розділі буде розглянуто спосіб отримання дампів оперативної памяті у випазду коли цьільовою системою є система під управлінням операційної системи MacOS.

«Mac Memory Reader» (MMR) був першим інструментом, доступним для отримання фізичної пам'яті з систем Mac. Його можна використовувати безкоштовно, але з закритим вихідним кодом.

MMR включає компонент користувацького простору і розширення ядра. Після завантаження розширення ядра створює два файли пристроїв:

- /dev/mem: цей пристрій експортує вміст фізичної пам'яті, щоб зробити його доступним для компонента робочих просторів.
- /dev/pmap: цей пристрій експортує список діапазонів фізичної пам'яті.

Інструмент для користувача простору спочатку запитує файл / dev / pmap, щоб отримати зміщення і розміри діапазонів фізичної пам'яті. Потім він запитує відповідні дані, читаючи / dev / mem. За замовчуванням MMR зберігає вибірку пам'яті в форматі файлу Mach-O, який описується далі в цьому розділі. У метаданих файлу зберігається асоціація зсувів у файлі дампа пам'яті зі зсувами в фізичної пам'яті. Адресний простір Mach-O Volatility (volatility / plugins / addrspaces / macho.py) прозоро транслює зміщення під час аналізу пам'яті.

Нижче показано отримання з MMR з використанням формату Mach-O за замовчуванням:

```
$ sudo ./MacMemoryReader mem.dmp
No kernel file specified, using '/mach_kernel'
```

Dumping memory regions:

available 0000000000000000 (568.00 KB) [WRITTEN]

available 0000000000090000 (64.00 KB) [WRITTEN]

available 0000000000100000 (511.00 MB) [WRITTEN]

available 0000000020200000 (199.00 MB) [WRITTEN]

LoaderData 000000002c900000 (76.00 KB) [WRITTEN]

available 000000002c913000 (948.00 KB) [WRITTEN]

LoaderData 000000002ca00000 (5.26 MB) [WRITTEN]

available 000000002cf42000 (760.00 KB) [WRITTEN]

LoaderData 000000002d000000 (35.21 MB) [WRITTEN]

RT_data 000000002f336000 (336.00 KB) [WRITTEN]

RT_code 000000002f38a000 (196.00 KB) [WRITTEN]

LoaderData 000000002f3bb000 (232.00 KB) [WRITTEN]

available 000000002f3f5000 (268.06 MB) [WRITTEN]

available 0000000040005000 (1.15 GB) [WRITTEN]

BS_data 0000000089d0f000 (84.00 KB) [WRITTEN]

available 0000000089d24000 (4.12 MB) [WRITTEN]

[snip]

Reported physical memory: 8589934592 bytes (8.00 GB)

Statistics for each physical memory segment type:

reserved: 6 segments, 46727168 bytes (44.56 MB)--assigned to unreadable device

LoaderCode: 2 segments, 516096 bytes (504.00 KB) -- WRITTEN

LoaderData: 35 segments, 42881024 bytes (40.89 MB) -- WRITTEN

BS_code: 83 segments, 2093056 bytes (2.00 MB) -- WRITTEN

BS_data: 109 segments, 43204608 bytes (41.20 MB) -- WRITTEN

RT_code: 1 segment, 200704 bytes (196.00 KB) -- WRITTEN

RT_data: 1 segment, 344064 bytes (336.00 KB) -- WRITTEN

available: 20 segments, 8436510720 bytes (7.86 GB) -- WRITTEN

ACPI_recl: 1 segment, 155648 bytes (152.00 KB) -- WRITTEN

ACPI_NVS: 1 segment, 262144 bytes (256.00 KB) -- WRITTEN

MemMapIO: 3 segments, 217088 bytes (212.00 KB) -- assigned to unreadable device

Total memory written: 8526168064 bytes (7.94 GB)

*Total memory assigned to unreadable devices *

(not written): 46944256 bytes (44.77 MB)

Reported memory not in the physical memory map: 16822272 bytes (16.04 MB)

У попередньому висновку ми виділили кілька цікавих рядків. Перший рядок повідомляє, що машина стверджує, що у неї 8 ГБ оперативної пам'яті. В останніх 3 рядках виведення зазначено, що в захоплення записано 7,94 ГБ ОЗУ; 44,77 МБ ОЗУ було призначено нечитабельним пристроїв; і 16,04 МБ, як повідомлялося, існують, але не були в фактичної мапі фізичної пам'яті. Якщо скласти останні 3 рядки, вийде 7,9999 ГБ (по суті, 8 ГБ). Якби ці числа істотно відрізнялися від 8 ГБ, ви б перевірили діапазони, в яких повідомляє ядро, на узгодженість і можливе пряме маніпулювання об'єктами ядра (DKOM) від шкідливих програм. Точно так же, якщо ви знали, що обсяг ОЗУ, встановленого в системі, що не дорівнює 8 ГБ, вам слід перевірити відповідні структури даних. Після завершення зразок пам'яті може зразу застосовуватись для аналізу.

ВИСНОВКИ

На основі всього вищезазначеного можна зробити наступні висновки:

1. 1.На основі загальних відомостей про архітектури апаратного та прогамного забезпечення запропоновано методичні рекомендації для ведення слідства в яких доказовими матерілами є знімки оперативної памяті зараженої або підозлювашої техніки.
2. Враховуючи актуальний стан ринку програмного забезпечення запроновано ряд програмного забезпечення для зняття дампів оперативної памяті також приведено прикла по їх налаштуванню та роботі з ними.
3. Запропонована схема дій в ситуаціїх коли описане в цій роботі програмне забезпечення не підходить для аналізу або зняття знімків оперативної памяті.
4. Описано теоритичну та технічну базу з якою повинен бути знайомий слідчий який бере учать у розслідуванні.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.

1. J. Alex, Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. - Lest We Remember: Cold Boot Attacks on Encryption Keys., 01 May 2009, 50 -56
2. Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, 4 April 02, 2021 розділ 20 PROCESSOR IDENTIFICATION AND FEATURE DETERMINATION.
3. Michael Johas Teener - Technical Introduction to IEEE 1394 06.Mar.2018.
4. Abhiram Kumar - Basics of Memory Forensics 18 Aug 2020.
5. Windows SCOPE Cyber Forensics – Ultimate Quick Start Guide 07 July 2016, 1-10.
6. Fabio Pagani. - ADVANCES IN MEMORY FORENSICS 09 September 2019, 1-106.
7. Andrew Case - Memory forensics: The path forward March 2017, 23-33.
8. Jacob Taylor, Benjamin Turnbull, Gideon Creech - Volatile Memory Forensics Acquisition Efficacy: A Comparative Study Towards Analysing Firmware-Based Rootkits August 2018, 1-11.
9. Rami Sihwail - Malware Detection Approach Based on Artifacts in Memory Image and Dynamic Analysis, September 2019, 1-12.
10. Sanjiv K, BhatiaShailesh, TiwariSu, RuidanMunesh, Chandra TrivediK, K. Mishra - Advances in Computer Communication and Computational Sciences: Proceedings. 12 October 2019 250-329.