ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА СИСТЕМ ТА ТЕХНОЛОГІЙ КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Рекомендації щодо захисту кінцевих точок з використанням Osquery на платформі Mac M1»

зі спеціальності *125 Кібербезпека*
(код, найменування спеціальності)

освітньо-професійної програми *Інформаційна та кібернетична безпека*
(назва програми)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Анна ТАЛАН
(підпис)

Виконав: здобувач вищої освіти групи БСД-44
ТАЛАН Анна
(прізвище, ім'я)

Керівник д.ф., МАРЧЕНКО Віталій
(науковий ступінь, вчене звання, прізвище, ім'я)

Рецензент _____
(науковий ступінь, вчене звання, прізвище, ім'я)

Київ 2025

ВСТУП

Актуальність дослідження. У сучасному світі інформаційних технологій та глобалізації цифрових процесів, що швидко розвиваються, безпека кінцевих точок стала однією з найважливіших сфер, в якій організації захищають свої інформаційні системи. Кінцеві точки - це пристрої, які безпосередньо взаємодіють з користувачами та мережевою інфраструктурою і часто використовуються для запуску кібератак, які можуть призвести до витоку конфіденційних даних, порушення бізнес-процесів та значних фінансових збитків. З огляду на постійне ускладнення середовища загроз, включаючи все більш витончені атаки, такі як фішинг, програми-вимагачі, безфайлові атаки і цільові APT, потреба в надійному захисті кінцевих точок стає ще більш критичною.

Сучасні рішення для захисту кінцевих точок, такі як платформи EDR, не тільки виявляють і блокують відомі загрози, а й виконують детальний аналіз поведінки пристроїв в режимі реального часу, підвищуючи ефективність реагування на інциденти. Одним з перспективних інструментів у цій галузі є Osquery – рішення з відкритим вихідним кодом, яке дозволяє здійснювати гнучкий і масштабований моніторинг кінцевих точок за допомогою SQL-запитів для отримання детальної інформації про стан системи, процеси, мережеві з'єднання та інші важливі параметри.

Враховуючи зростаючу складність кіберзагроз та необхідність інтеграції різних рішень в єдину систему безпеки, важливо дослідити роль Osquery в сучасних стратегіях кібербезпеки та розробити рекомендації щодо його використання на сучасних платформах, включаючи Mac M1. Це дозволить підвищити рівень захисту кінцевих точок, забезпечити своєчасне виявлення інцидентів та знизити ризики для Організацій.

Предмет дослідження – методи та способи захисту кінцевих точок організації з використанням Osquery на платформі Mac M1.

Мета роботи визначити методи та способи реалізації захисту кінцевих точок з використанням Osquery на платформі Mac M1 та розробити рекомендації щодо застосування Osquery для захисту кінцевих точок Організації.

Визначення актуальності проблеми захисту кінцевих точок організації з

використанням платформи Mac M1.

Аналіз наукової та технічної літератури з питань теми кваліфікаційної роботи.

Аналіз методів та способів захисту кінцевих точок Організації з використанням Osquery.

Практична реалізація Osquery для захисту кінцевих точок Організації.

Розроблення рекомендацій щодо застосування Osquery для захисту кінцевих точок Організації.

Оформлення результатів дослідження.

Підготовка доповіді до захисту.

Наукові завдання:

дослідити сутність проблеми захисту кінцевих точок Організацій;

проаналізувати методи та засоби захисту кінцевих точок Організації;

проаналізувати існуючі рішення із захисту кінцевих точок Організації;

проаналізувати методи та способи захисту кінцевих точок організації з використанням Osquery на платформі Mac M1;

запропонувати способи захисту кінцевих точок організації з використанням Osquery на платформі Mac M1;

розробити рекомендацій щодо застосування методів та способів захисту кінцевих точок Організації.

Методи дослідження – опрацювання літератури за даною темою, аналіз експлуатаційної документації, міжнародних стандартів та їх порівняння.

Практичне значення одержаних результатів: запропоновано способи реалізації захисту кінцевих пристроїв з використанням Osquery. Розроблено рекомендації щодо реалізації захисту кінцевих точок Організації з використанням Osquery на платформі Mac M1.

1 АНАЛІЗ ПРОБЛЕМИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ КІНЦЕВИХ ТОЧОК

1.1. Постановка проблеми та роль Osquery у сучасних стратегіях кібербезпеки

Постановка проблеми

Одним з важливих складових кібербезпеки є захист кінцевих точок. Атаки на кінцеві точки, такі як комп’ютери, смартфони та інші пристрої підключені до мережі, можуть призвести до втрати конфіденційних даних, репутації та фінансів організації. Враховуючи швидкоплинність технологій важливо забезпечити достатній рівень безпеки, оскільки традиційні методи захисту не завжди можуть запобігти атакам.

Endpoint Detection and Response (EDR): поняття, принципи роботи та значення

EDR (Endpoint Detection and Response) – це категорія рішень у сфері кібербезпеки, що застосовуються для виконання виявлення, моніторингу, аналізу та реагування на підозрілу та аномальну активність та загрози на кінцевих пристроях, наприклад, комп’ютерах, ноутбуках, серверах та мобільних пристроях. Дані рішення використовуються з метою забезпечення підвищеного рівня захисту кінцевих пристроїв шляхом постійного збору телеметрії з пристроїв, аналізу поведінки, виявлення підозрілих дій та можливості миттєвого реагування на інциденти кібербезпеки [1].

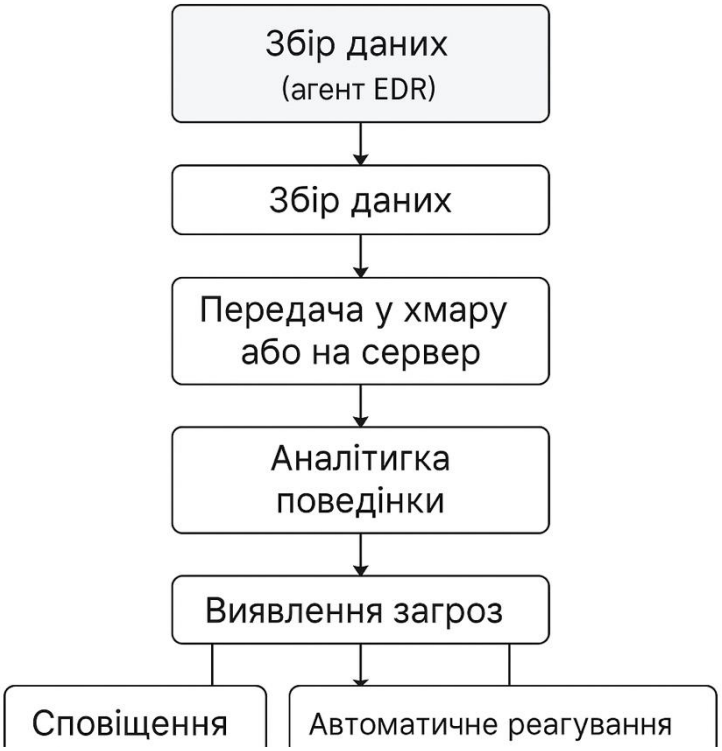

Рис. 1.1. Типова схема роботи EDR-системи

Компоненти EDR:

- Агент кінцевих точок (збирає логи, процеси, з'єднання, журнали тощо);
- Модуль аналізу (правила виявлення);
- Інтерфейс адміністратора (інформаційна панель для реагування та моніторингу);
- Криміналістичний модуль (для розслідування інцидентів).

Основним принципом EDR є поєднання проактивного захисту (шляхом раннього виявлення зловмисних дій) з швидким реагуванням (ізоляція загрози, припинення процесу, збір даних для судової експертизи і т.д.). Системи EDR також дозволяють фахівцям з безпеки аналізувати ланцюжок подій, що призвели до інциденту, що є важливим для розслідування атак [2].

На відміну від традиційних антивірусних систем, які працюють переважно з сигнатурами, EDR використовує поведінковий аналіз і можуть виявляти невідомі загрози, які не мають відповідної сигнатури в базі даних [3].

Основні можливості EDR:

- Моніторинг кінцевих точок у реальному часі;
- Збір та агрегація логів і телеметрія;

- Виявлення загроз на основі правил і поведінкових моделей;
- Автоматизоване або ручне реагування на інциденти;
- Криміналістичний аналіз інцидентів [4].

Сучасні рішення EDR часто є частиною великих платформ розширеного виявлення та реагування (XDR), які поєднують дані з різних джерел (мережа, сервери, хмарне середовище тощо), але EDR все ще залишається важливою частиною захисту кінцевих пристроїв.

Існує також ряд рішень з відкритим вихідним кодом, які дозволяють вільно конфігурувати EDR в програмному забезпеченні, включаючи такі інструменти, як Osquery, який є незалежний від платформи інструмент для отримання системної інформації за допомогою SQL-запитів [2, 5, 6].

З огляду на зростання загроз (особливо для медичних даних, інфраструктури підприємства та Інтернету речей), використання рішень EDR стало майже обов'язковим елементом кіберзахисту організації [7, 8, 9].

Роль Osquery у сучасних стратегіях кібербезпеки

Одним із сучасних рішень для захисту кінцевих точок є Osquery, яке застосовується у сучасних стратегіях кіберзахисту. За допомогою Osquery можна здійснювати моніторинг та отримувати детальну інформацію про стан пристроїв.

Основні переваги Osquery: Використання Osquery у сучасних стратегіях кібербезпеки:

Ключовими перевагами Osquery вважається його здатність забезпечувати високий рівень видимості стану всіх систем та інфраструктури. Така характеристика дозволяє отримувати інформацію в режимі реального часу, що є надважливим при реагуванні на атаки. Також Osquery легко поєднується з іншими інструментами захисту кінцевих точок, можна поєднувати з платформами для розвідки загроз, системами управління активами. Така інтеграція дозволяє підвищити ефективність захисту інфраструктури організації. Варто відзначити і його легкість у впровадженні оскільки для інсталяції системи не потрібно глибоких технологічних знань та принципів. Дана характеристика дозволяє інсталювати Osquery багатьом організаціям [10]. Остання, але не за значенням його перевага це

те, що Osquery дозволяє використовувати SQL-запити. Використання даних запитів полегшує проведення аудитів, дотримання нормативно-правових вимог, виявлення загроз.

Використання системи в сучасних стратегіях кібербезпеки:

Перше, за допомогою Osquery здійснюється відстеження мережевої активності, що надає можливість виявляти підозрілі дій, наприклад, виявлення не авторизованих спроб підключення до мережі.

Друге, здійснення управління конфігураціями безпеки, шляхом отримання інформації про налаштування в усіх пристроях, що під'єднані до мережі, підвищуючи при цьому загальний рівень кібербезпеки.

Третє, Osquery використовується для сканування наявних процесів і файлів на кінцевих точках, з метою виявлення шкідливого програмного забезпечення, яке є часто використовуваним в сучасних методах атаки [11].

Четверте, з допомогою Osquery здійснюється процес реагування на інциденти, оскільки система забезпечує первинне виявлення загроз, що допомагає в подальшому розслідування інцидентів.

П'яте, Osquery використовується для перевірки організації на відповідність нормативно-правовим вимогам в сфері захисту даних, оскільки система надає необхідну інформацію щоб перевірити стан захищеності конфіденційних файлів.

Шосте, для забезпечення аудиту конфігурації безпеки кінцевих точок, також використовується Osquery, оскільки система надає необхідну інформацію, наприклад, про наявні оновлення безпеки чи правильно налаштовані фаєрволи в мережі.

Сьоме, для забезпечення додаткового захисту та моніторингу стану мережі Osquery інтегрується з системами управління інформацію та подіями безпеки (SIEM), що забезпечує комплексний аналіз безпеки та з допомогою якого можливе здійснення автоматизованого реагування на інциденти [12].

1.2. Види та статистика загроз для кінцевих точок

Основні види загроз для кінцевих точок

В сучасному світі з розвиненими технологіями захисту, зловмисники мають проявляти креативність та чималі розумові здібності задля реалізації успішних атак на кінцеві точки. Нижче проілюстровано реальні приклади успішних атак:

1. Безперечно одним з основних вектором атак залишається соціальна інженерія (фішингові атаки). Наприклад отримання спуфінгованих листів від банків або фейкових сповіщень про доставку призводять до викрадення даних або завантаження шкідливого програмного забезпечення [1,7]. Згідно статистики 81% організацій в 2021 році стикнулися з атаками які мали назву malware [3].

2. Неможливо залишити без уваги атаки типу WannaCry або NotPetya, які відомі, тим що було здійснено шифрування даних, вимагаючи викуп. Переглядаючи статистику за 2021 рік та 2020 рік, можна помітити значне збільшення кількості організацій (в 2020 році на 148% більше ніж в 2021 році), які стали жертвами успішних атак типу Ransomware [8, 13]**.** Середні витрати на відновлення подвоюються при сплаті викупу [13].

3. Також набувають популярність DDoS та ботнет-атаки, за допомогою використання IoT-пристроїв. У 2025 році очікується 40 млрд IoT-пристроїв, що значно збільшує ризики [8].

4. Ще одна тенденція в сучасному кіберпросторі це APT атаки. Дана атака є цілеспрямованою атакою, наприклад, як операція SolarWinds, де зловмисники викрадали конфіденційні дані, при цьому залишаючись непоміченими протягом багатьох місяців [9].

5. Незмінним прикладом здійснення атак залишається безфайловий malware, який виконується за допомогою використання легальних інструментів, як наприклад, PowerShell, для впровадження шкідливого коду. Оскільки при цьому не здійснюється запис на диск, це значно ускладнює виявлення даної атаки [9].

6. Також, здійснення атак через сторонніх постачальників, використовуючи вразливості в ланцюгах поставок є одним з прикладів успішних

методів реалізації атак. Наприклад, інцидент з Kaseya VSA у 2021 році, коли через оновлення ПО поширився ransomware [8].

Актуальна статистика (2025 рік)

Переглядаючи актуальну статистику за 2025 рік, можна дійти до висновку, що збитки, які несуть організації у зв'язку із значним збільшенням кількості проведення атак, значно зросли у порівнянні з минулими роками, що підвищує необхідність захисту своїх кінцевих точок.

Таблиця 1.1.

Актуальна статистика за 2025 рік

Параметр	Дані
Частка організацій, що зазнали атак	68%
Прогнозовані річні витрати на кіберзлочинність	$10.5 трлн
Зростання ринку endpoint security до 2032	$31.2 млрд (CAGR 12.1%)
Частка атак з використанням AI	40% (прогноз на 2025 рік)
Організації зі слабкими BYOD-політиками	67%

Тенденції 2025 року

1. Набувають популярності атаки з використанням штучного інтелекту, за допомогою якого можна створити гіперреалістичні фішингові листи або використовувати AI для автоматизації експлойтів. Наприклад, генерування deepfake-аудіо для проведення соціальної інженерії [8].

2. Згідно статистики в 2025 році 55 % фахівців з кібербезпеки вважають, що смартфони є найвразливішими кінцевими пристроями. Адже зараз набуває популярності використання вразливостей у веб-переглядачах для drive-by download [7].

3. Кризові сценарії**:**

- Затримка оновлень**:** 40% компаній відкладають патчі через побоювання конфліктів [13].
- Нешифровані дані**:** Лише 50% компаній шифрують чутливу інформацію на пристроях [13].

Кожного дня загрози для кінцевих пристроїв розвиваються разом із технологіями захисту: від простого фішингу до атак з використанням штучного інтелекту. Недостатня обізнаність працівників в сфері кібербезпеки, популяризація віддаленої роботи та зростання кількості IoT-пристроїв роблять проблему ще більше актуальною. За даними досліджень, кожні 11 секунд відбувається ransomware-атака, що підкреслює критичність проактивних заходів захисту [1].

1.3. Переваги та недоліки Osquery у порівнянні з іншими EDR-системами

Таблиця 1.2.

Порівняльна таблиця основних характеристик

Критерій	Osquery	Традиційні EDR (CrowdStrike, Velociraptor)	Відкриті EDR (Wazuh)
Гнучкість	Висока (SQL-запити, кастомні таблиці)	Обмежена фреймворком	Середня (на основі правил)
Видимість даних	Детальна (200+ системних таблиць)	Селективна (орієнтована на сигнали загроз)	Базова (лог-файли, мережева активність)
Масштабованість	Обмежена без сторонніх рішень [2]	Оптимізована для великих флотів	Середня (залежить від архітектури)
Вартість	Безкоштовний (open-source)	Висока (ліцензійні збори)	Безкоштовний/підписка
Інтеграція з іншими системами	Широка (SIEM, MITRE ATT&CK [5])	Закриті API	Обмежена (переважно власні модулі)
Складність налаштування	Висока (потрібні SQL та C++ навички [2])	Низька (GUI-інтерфейси)	Середня

Ключові переваги Osquery

Однією з ключових переваг використання Osquery для захисту кінцевих точок є її крос-платформовість, оскільки систему можна розгорнути на багатьох

платформах таких як Windows, Linux, macOS та контейнерів (наприклад, Docker) при цьому не здійснюючи змін в коді. За допомогою програми здійснюється моніторинг 95% ОС у гетерогенному середовищі [5].

Також варто звернути увагу на здатність Osquery до глибинного аналізу. За допомогою Osquery можна здійснювати:

- Моніторинг реєстрів Windows (автозавантаження, scheduled tasks)
- Аналіз процесів з використанням YARA-правил [5]
- Виявлення безфайлових атак через PowerShell-активність [4]

За допомогою Osquery також можливе здійснення проективного аудиту. Це корисно тим, що результати такого аудиту інтегруються з платформами типу Splunk для автоматизації алертів [6].

```
osquery> SELECT
    ...>   p.pid,
    ...>   p.name,
    ...>   p.path,
    ...>   n.remote_address,
    ...>   n.remote_port,
    ...>   n.local_port,
    ...>   n.state
    ...> FROM
    ...>   process_open_sockets n
    ...> JOIN
    ...>   processes p
    ...> ON
    ...>   n.pid = p.pid
    ...> WHERE
    ...>   n.remote_address NOT LIKE '192.168.%'
    ...>   AND n.remote_address NOT LIKE '10.%'
    ...>   AND n.remote_address NOT LIKE '172.16.%'
    ...>   AND n.remote_address != ''
    ...>   AND n.remote_address != '::1';
```

Рис. 1.2. Приклад запиту для виявлення підозрілих підключень

pid	name	path	remote_address	remote_port	local_port	state
978	rapportd	/usr/libexec/rapportd	0.0.0.0	0	57237	LISTEN
978	rapportd	/usr/libexec/rapportd	::	0	57237	LISTEN
990	identityservicesd	/System/Library/PrivateFrameworks/IDS.framework/identityservicesd.app/Contents/MacOS/identityservicesd	0.0.0.0	0	0	
990	identityservicesd	/System/Library/PrivateFrameworks/IDS.framework/identityservicesd.app/Contents/MacOS/identityservicesd	0.0.0.0	0	0	
990	identityservicesd	/System/Library/PrivateFrameworks/IDS.framework/identityservicesd.app/Contents/MacOS/identityservicesd	fe80:13::cebb:723f:4a6f:526f	1024	1024	ESTABLISHED
990	identityservicesd	/System/Library/PrivateFrameworks/IDS.framework/identityservicesd.app/Contents/MacOS/identityservicesd	fe80:13::cebb:723f:4a6f:526f	1027	1025	ESTABLISHED
990	identityservicesd	/System/Library/PrivateFrameworks/IDS.framework/identityservicesd.app/Contents/MacOS/identityservicesd	fe80:13::cebb:723f:4a6f:526f	1027	1025	ESTABLISHED
1015	sharingd	/usr/libexec/sharingd	0.0.0.0	0	0	
1015	sharingd	/usr/libexec/sharingd	0.0.0.0	0	61016	
1024	replicatord	/System/Library/PrivateFrameworks/ReplicatorCore.framework/Support/replicatord	0.0.0.0	0	64385	
1166	ControlCenter	/System/Library/CoreServices/ControlCenter.app/Contents/MacOS/ControlCenter	0.0.0.0	0	7000	LISTEN
1166	ControlCenter	/System/Library/CoreServices/ControlCenter.app/Contents/MacOS/ControlCenter	::	0	7000	LISTEN
1166	ControlCenter	/System/Library/CoreServices/ControlCenter.app/Contents/MacOS/ControlCenter	0.0.0.0	0	5000	LISTEN
1166	ControlCenter	/System/Library/CoreServices/ControlCenter.app/Contents/MacOS/ControlCenter	::	0	5000	LISTEN
18517	figma_agent	/Users/anna/Library/Application Support/Figma/FigmaAgent.app/Contents/MacOS/figma_agent	0.0.0.0	0	44950	LISTEN
18517	figma_agent	/Users/anna/Library/Application Support/Figma/FigmaAgent.app/Contents/MacOS/figma_agent	0.0.0.0	0	44960	LISTEN
22386	TelegramShare	/Applications/Telegram.app/Contents/PlugIns/TelegramShare.appex/Contents/MacOS/TelegramShare	149.154.167.41	443	57016	ESTABLISHED
22386	TelegramShare	/Applications/Telegram.app/Contents/PlugIns/TelegramShare.appex/Contents/MacOS/TelegramShare	149.154.167.41	443	57017	ESTABLISHED
31350	Telegram	/Applications/Telegram.app/Contents/MacOS/Telegram	149.154.167.41	443	57238	ESTABLISHED
31350	Telegram	/Applications/Telegram.app/Contents/MacOS/Telegram	149.154.167.41	443	57239	ESTABLISHED
31350	Telegram	/Applications/Telegram.app/Contents/MacOS/Telegram	149.154.167.35	443	57240	ESTABLISHED
31350	Telegram	/Applications/Telegram.app/Contents/MacOS/Telegram	149.154.167.41	443	57241	ESTABLISHED
31350	Telegram	/Applications/Telegram.app/Contents/MacOS/Telegram	149.154.167.41	443	57242	ESTABLISHED
31350	Telegram	/Applications/Telegram.app/Contents/MacOS/Telegram	149.154.167.35	443	57246	ESTABLISHED
31350	Telegram	/Applications/Telegram.app/Contents/MacOS/Telegram	149.154.167.35	443	57244	ESTABLISHED
31350	Telegram	/Applications/Telegram.app/Contents/MacOS/Telegram	149.154.167.35	443	57245	ESTABLISHED

Рис. 1.3. Результат виконання запиту

Ще однією з переваг Osquery є створення кастомних запитів, що відповідають технікам TTP (наприклад, T1059.001 для зловживання Power Shell) [5].

Недоліки Osquery порівняно з конкурентами

Перш за все, одним з недоліків Osquery є відсутність автоматизованого реагування. Osquery не надає механізмів для блокування загроз, як наприклад CrowdStrike, тому виникає потреба в інтеграції з SOAR-системами [4].

Ще один недолік це високі накладні витрати. Для того щоб зберегти всі дані, які 10 1000 кінцевих точок генерують щодня необхідно приблизно 2 терабайти пам’яті щодня, також комплексні запити збільшують використання CPU-ресурсів на 15-20% [2].

Також, Osquery не здійснює моніторинг безсерверних середовищ, як наприклад, AWS Lambda або об’єкти S3 [2], що являє собою обмежену підтримку хмарних сервісів.

Для того щоб здійснювати аналіз інформації необхідне використання додаткових інструментів, наприклад, Uptycs або Elastic Stack, що значно ускладнює процес обробки даних [6].

Сценарії застосування: коли обирати Osquery?

1. Великі організації з DevOps-експертизою

Приклад: Facebook використовує Osquery для моніторингу 4+ млн серверів через інтеграцію з власними системами логування.

2. Регульовані галузі (фінанси, охорона здоров'я)

Приклад: Використання Osquery для отримання інформації для PCI DSS-аудиту.

3. Гібридні інфраструктури

Моніторинг одночасно локальних серверів і віртуальних машин AWS EC2 [2].

Тож, Osquery є оптимальним вибором для організації, які мають власні ресурси для обробки великих даних, мають потреби в гнучкому інструменті для кастомних сценаріїв проведення внутрішнього аудиту та які мають можливість використовувати мультиплатформні середовища. Для малих компаній, які не мають окремого відділу з інформаційної безпеки більш ефективнішим буде використання EDR-рішення з графічним інтерфейсом або SaaS-рішення. Як зазначають експерти "open-source EDR-рішення ефективні лише при наявності 3+ фахівців з інформаційної безпеки" [14].

Висновок до Розділу 1

В даному розділі висвітлюється наявність проблеми забезпечення безпеки кінцевих точок та продемонстрована роль Osquery в сучасних стратегіях побудови кібербезпеки.

Згідно статистик кінцеві точки стають все частіше і частіше об’єктами атаки з боку зловмисників, що призводить до витоку даних та фінансових втрат. Оскільки технології постійно розвиваються, традиційні методи захисту кінцевих точок не можуть забезпечити належного рівня захисту через зростаючу складність загроз.

За допомогою таких інструментів як Osquery, можна забезпечити високу видимість стану пристроїв та здійснювати інтеграцію з іншими інструментами безпеки, що підвищує рівень захисту кінцевих точок. За допомогою Osquery

можливо здійснювати детальний аналіз стану кінцевих точок здій, проводити аудит та виявляти загрози на ранніх стадіях.

Також відповідно до наданої статистики можна дійти до висновку, що відбувається значне зростання кількості та складності атак, що підкреслює необхідність впровадження ефективних заходів захисту кінцевих точок. Адже такі загрози як фішингові атаки, ransomware, DDoS та APT становлять значну загрозу для організацій.

До того ж, в розділі було розглянуто переваги та недоліки використання Osquery. Дана система пропонує високу гнучкість та детальну видимість даних, але вимагає глибоких технічних навичок та значних ресурсів для обробки даних. У порівнянні з іншими EDR-системами, Osquery є безкоштовним рішення, проте підходить для організації з великими ресурсами та потребами в кастомних рішеннях.

Підсумовуючи, Osquery є потужним інструментом для забезпечення безпеки кінцевих точок, особливо для масштабних організацій з наявними потребами в гнучкості та з вимогами до деталізації даних. Однак для впровадження системи присутня потреба в наявності відповідних навичок працівників, необхідно забезпечити належними технічними ресурсами та для забезпечення більшої ефективності захисту кінцевих точок, потрібна інтеграція з іншими інструментами безпеки.

2 МЕТОДИ ТА ЗАСОБИ ЗАХИСТУ КІНЦЕВИХ ТОЧОК В ОРГАНІЗАЦІЇ

2.1. Методи та засоби захисту кінцевих точок

Кінцева точка (endpoint) – це будь-який пристрій, що підключається до мережі організації: комп'ютери, смартфони, сервери, пристрої Інтернету речей тощо. Кінцеві точки відіграють важливу роль у забезпеченні безпеки, оскільки за замовчуванням вони часто мають низький рівень захисту, хоча активно взаємодіють з мережею [15, 16].

Методи захисту кінцевих точок

Традиційні методи:

1. Антивірусне програмне забезпечення

Нижче наведено таблицю порівняння типів антивірусного програмного забезпечення [15, 17]:

Таблиця 2.1

Категорія антивірусного програмного забезпечення	Тип	Характеристика	Приклад
Метод виявлення	Сигнатурний (signature-based)	Використовують базу відомих вірусних сигнатур. Ефективні проти вже відомих загроз.	Avast Free Antivirus, Avira Antivirus, ESET NOD32 Antivirus
	Евристичний (heuristic-based)	Аналізують поведінку або структуру файлів для виявлення нових/невідомих шкідників.	Kaspersky Anti-Virus, Dr.Web, AVG Antivirus
	Поведінковий (behavior-based)	Моніторять активність програм у реальному часі та виявляють підозрілу поведінку.	Bitdefender Total Security, Trend Micro Maximum Security

	AI/ML	Виявляють загрози, аналізуючи великі обсяги даних про шкідливу активність.	Sophos Intercept X, CrowdStrike Falcon, CylancePROTECT
	Sandbox	Запускають файли у віртуальному середовищі для аналізу їх поведінки.	Comodo Antivirus, FireEye Endpoint Security, Fortinet FortiSandbox
Спосіб розгортання	Локальний (standalone)	Встановлюються на окремий пристрій (ПК, сервер).	ESET NOD32, Bitdefender Antivirus Plus, Kaspersky Anti-Virus
	Централізований (network-based / centralized)	Використовуються в корпоративних мережах, мають централізовану консоль керування.	Symantec Endpoint Protection Manager, ESET PROTECT, Kaspersky Security Center
	Хмарний (cloud-based)	Аналіз загроз відбувається на стороні хмарного сервера.	Microsoft Defender for Endpoint, Trend Micro Apex One SaaS, CrowdStrike Falcon
Цільова платформа	Для настільних систем	Windows, macOS, Linux.	Kaspersky, Bitdefender, Sophos, ESET
	Для мобільних пристроїв	Android, iOS	Bitdefender Mobile Security, McAfee Mobile Security, Norton Mobile Security
	Для серверів	Windows Server, Linux Server	Kaspersky Security for Windows Server, ESET Server Security, Symantec Server AV

	Для спеціалізованих систем	IoT, SCADA/ICS	Kaspersky Industrial CyberSecurity, Nozomi Guardian, Claroty xDome
Функціональність	Базовий антивірус	Забезпечують лише базовий захист (сканування, віднесення до карантину та видалення шкідливого програмного забезпечення)	Avast Free Antivirus, ClamAV, Windows Defender Antivirus
	Антивірусні пакети (Internet Security Suites)	Включають фаєрвол, антиспам, VPN, батьківський контроль, захист браузера тощо	Norton 360 Deluxe, Bitdefender Internet Security, Kaspersky Total Security
	EDR (Endpoint Detection and Response)	Розширений захист із можливістю аналізу інцидентів, полювання на загрози (threat hunting) тощо	CrowdStrike Falcon, SentinelOne Singularity, Microsoft Defender for Endpoint
	XDR (Extended Detection and Response)	Охоплює не лише кінцеві точки, а й інші системи (мережа, хмара, пошта)	Palo Alto Cortex XDR, Trend Micro Vision One, Microsoft Defender XDR

2. Персональні брандмауери, що обмежують вхідний і вихідний трафік [17].

Нижче наведено таблицю порівняння типів антивірусного програмного забезпечення:

Таблиця 2.2

Порівняння типів антивірусного програмного забезпечення

Категорія брандмауера	Тип	Характеристика	Приклад
За способом фільтрації трафіку	Мережевий фільтр пакетів (Packet Filtering Firewall)	Аналізує заголовки IP-пакетів (IP-адреса, порт, протокол). Швидкий, але простий.	iptables, pf
	Становий (Stateful Inspection Firewall)	Відстежує стан з’єднань. Більш точний, ніж фільтр пакетів.	Cisco ASA, FortiGate
	Проксі-брандмауер (Application-level Gateway)	Працює на рівні застосунків. Може аналізувати HTTP, FTP, DNS тощо.	Squid, Blue Coat
	Брандмауер з глибокою інспекцією пакетів (DPI)	Аналізує вміст пакетів, шукає загрози, шкідливий код.	Palo Alto NGFW, FortiGate NGFW
	NGFW (Next-Generation Firewall)	Комбінує DPI, IDS/IPS, фільтрацію застосунків, антивірус тощо.	Palo Alto, Check Point
За розміщенням	Мережевий (Network-based Firewall)	Захищає всю мережу, встановлюється на межі мережі.	FortiGate, Cisco ASA
	Хостовий (Host-based Firewall)	Встановлюється на кінцевому пристрої.	Windows Defender Firewall, iptables на Linux
	Хмарний (Cloud Firewall / FWaaS)	Працює у хмарній інфраструктурі, не залежить від пристроїв.	Zscaler, Azure Firewall
За рівнем OSI-моделі	Рівень 3 (Мережевий)	Фільтрація IP-адрес, маршрутизація	Packet Filtering
	Рівень 4 (Транспортний)	Фільтрація за TCP/UDP портами	Stateful Firewall
	Рівень 7 (Застосунків)	Контроль HTTP, DNS, FTP тощо	Application Firewall, Proxy, NGFW

За принципом реалізації	Програмні (Software Firewalls)	Встановлюються як ПЗ на ОС	iptables, Windows Firewall
	Апаратні (Hardware Firewalls)	Фізичні пристрої	Cisco ASA, Fortinet FortiGate, Palo Alto
	Віртуальні (Virtual Firewalls)	Працюють у гіпервізорах, віртуальних середовищах	VMware NSX Distributed FW
	Хмарні (Cloud-native Firewalls)	Інтегровані в хмарну інфраструктуру	AWS WAF, Azure Firewall
За функціональним призначенням	Фільтраційні (Filtering Firewall)	Базова перевірка трафіку	IPFire, Windows Defender Firewall
	Зона безпеки / DMZ Firewall	Розмежування внутрішньої мережі та публічного доступу	OPNsense, Sophos XG Firewall
	Брандмауер з IDS/IPS	Виявлення та попередження / блокування атак	Suricata, Check Point Next Generation Firewall
	Web Application Firewall (WAF)	Захист веб-додатків від SQLi, XSS тощо.	Cloudflare WAF, AWS WAF, F5 WAF

3. Контроль пристроїв для обмеження доступу до USB-накопичувачів або технології Bluetooth [18].

Контроль пристроїв - це технологія інформаційної безпеки, яка дозволяє контролювати доступ до зовнішніх пристроїв, таких як USB-накопичувачі, Bluetooth-адаптери, CD/DVD-приводи, Wi-Fi адаптери тощо. Мета контролю пристроїв- запобігти несанкціонованому копіюванню, крадіжці або втраті інформації, а також встановлення політик безпечного використання зовнішніх пристроїв.

4. Шифрування даних для запобігання доступу до інформації у разі фізичної крадіжки пристрою (наприклад, BitLocker, VeraCrypt) [19].

Шифрування даних передбачає перетворення даних у формат, який неможливо прочитати без спеціального ключа. Основна мета шифрування -

захистити дані, навіть якщо пристрій потрапить до рук сторонньої особи (наприклад, якщо ноутбук, карта пам'яті, жорсткий диск тощо будуть вкрадені або загублені).

Сучасні методи (Next-Gen Endpoint Protection)

1. EDR (Endpoint Detection and Response): аналіз в режимі реального часу, поведінковий моніторинг та автоматизоване реагування на загрози. Це рішення часто використовується у великих організаціях для поглибленої телеметрії та реагування на інциденти [15, 16, 17]. Наприклад, Microsoft Defender for Endpoint , CrowdStrike Falcon, SentinelOne [18,19].

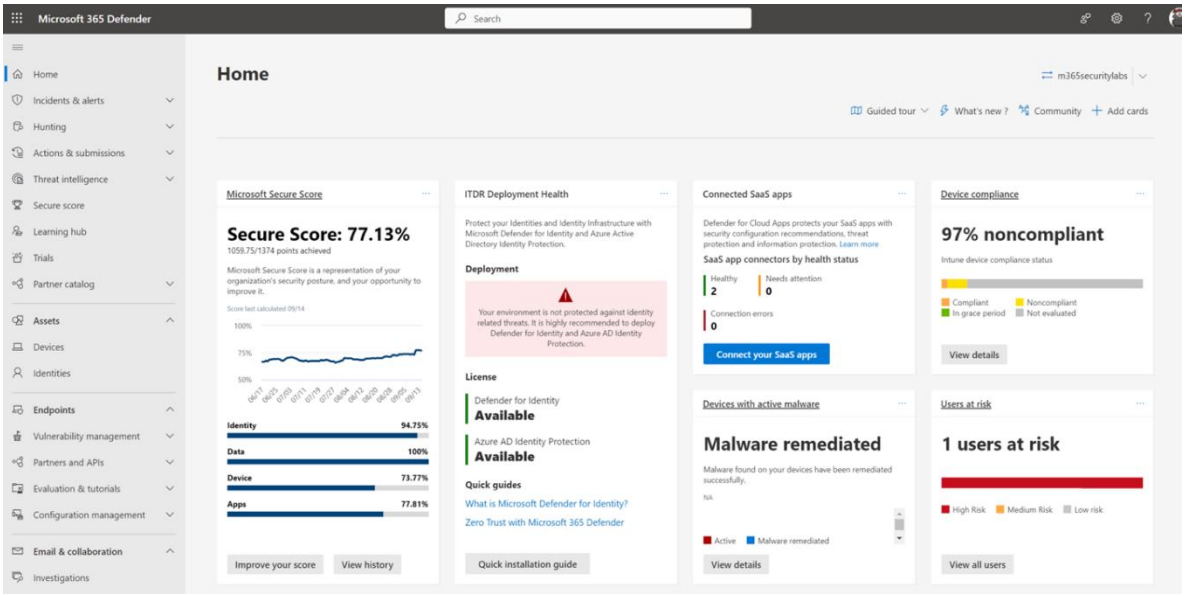

Рис. 2.1. Дашборд керування Microsoft Defender for Endpoint

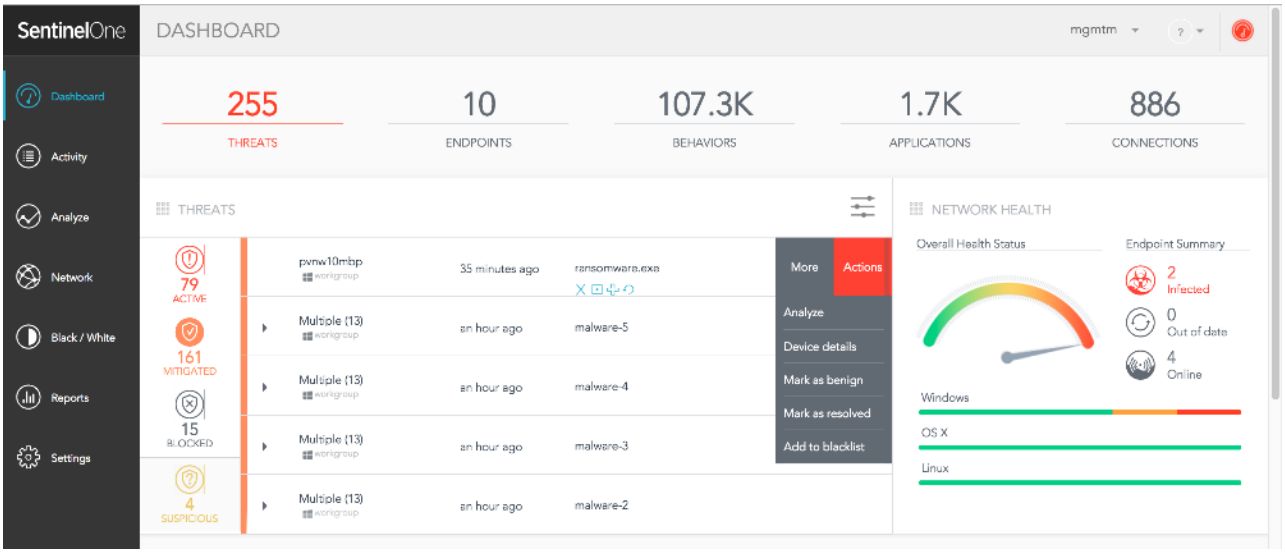

Рис. 2.2. Дашборд керування SentinelOne

2. XDR (Extended Detection and Response): рішення, які інтегрують дані не лише з терміналів, а й з інших систем: мережевої інфраструктури, електронної пошти, хмарних сервісів [16]. Наприклад: Cisco XDR, CrowdStrike Falcon XDR.

Рис. 2.3. Дашборд керування Cisco XDR

Рис. 2.4. Дашборд керування CrowdStrike Falcon XDR

3. Архітектура нульової довіри (ZTA): архітектура, в якій жодному користувачеві або пристрою не довіряють автоматично, навіть якщо він знаходиться в мережі [20, 21].

4. AI/ML-аналіз поведінки: метод машинного навчання для виявлення нетипової поведінки користувачів або процесів [15, 18].

5. Whitelisting додатків: запускає лише дозволені додатки та блокує всі інші [19].

6. Patch Management: перевірка оновлень системи для виправлення відомих вразливостей (наприклад, Tenable) [17].

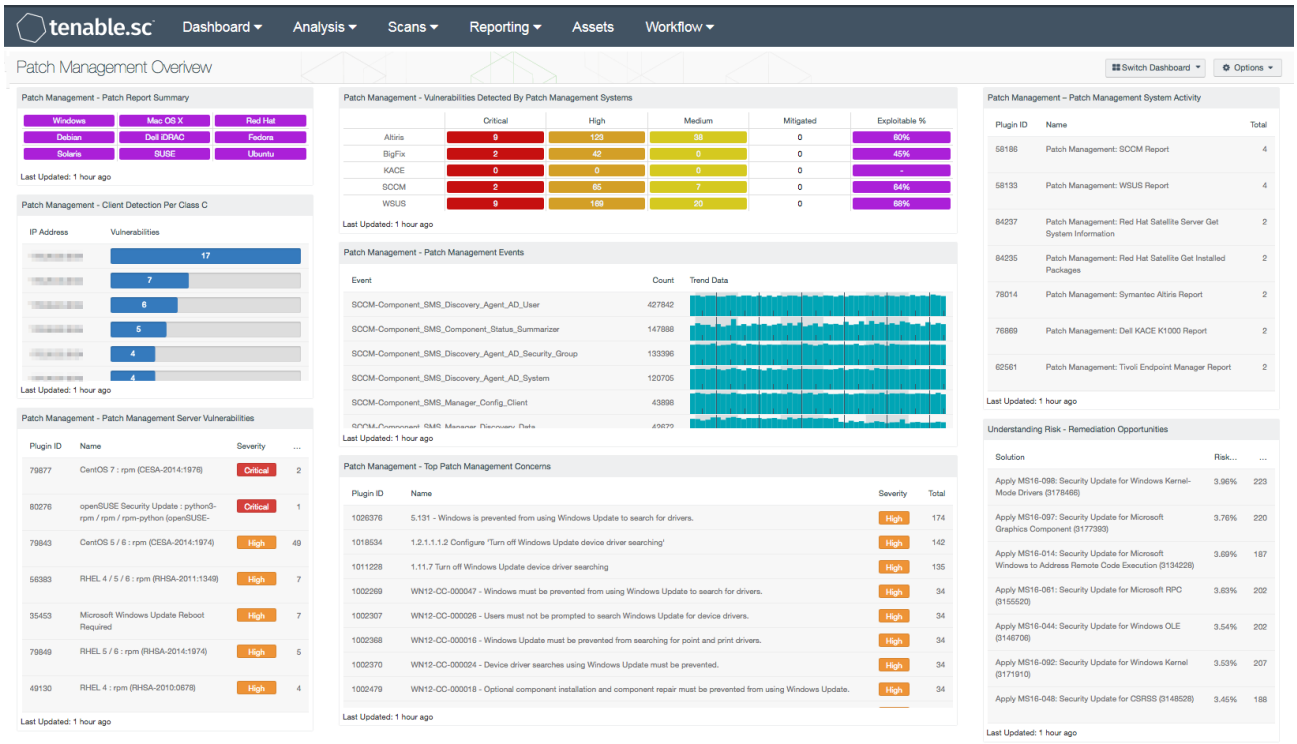

Рис. 2.5. Дашборд розгортання патчів Tenable

7. Sandboxing: запуск програм в ізольованому середовищі для зменшення ризиків [18].

8. MDM: централізоване управління мобільними пристроями, політиками безпеки та блокуванням пристроїв у разі втрати [17].

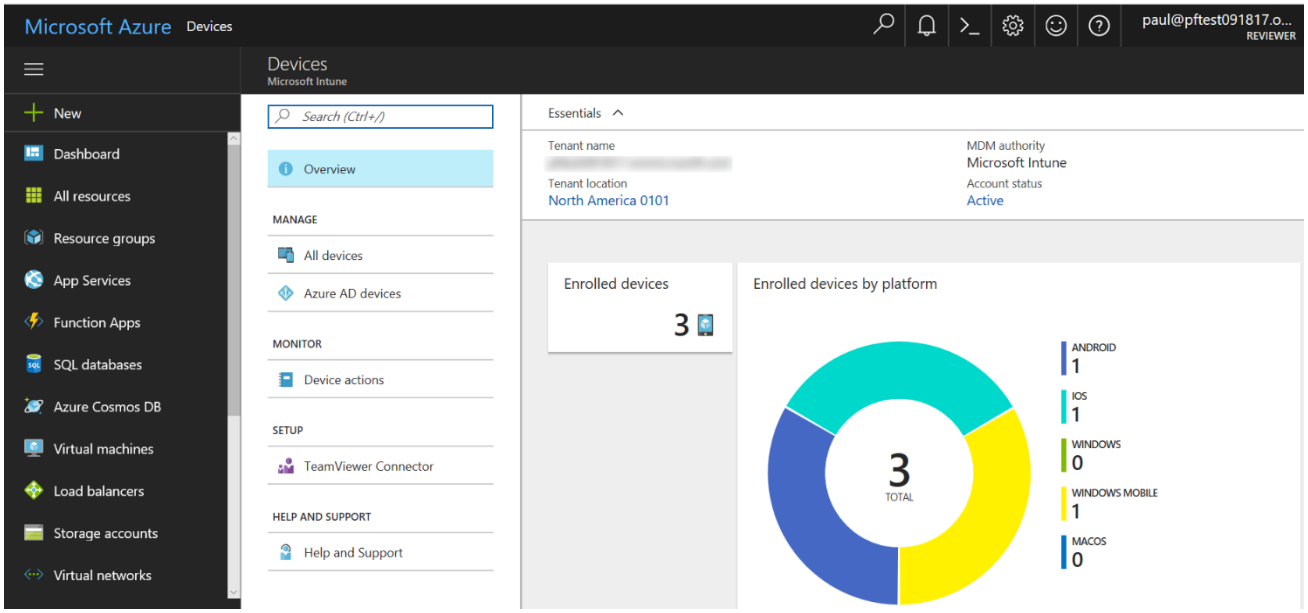

Рис. 2.6. Дашборд керування Microsoft Azure

Засоби захисту кінцевих точок (огляд)

Нижче в таблиці наведено приклади рішень, що використовуються в сучасних організаціях:

Таблиця 2.3

Рішення, що використовуються в сучасних організаціях

Назва	Тип	Опис
Microsoft Defender for Endpoint	EDR/XDR	Інтегрована безпека в Windows, інтеграція з Azure, Microsoft 365.
CrowdStrike Falcon	EDR	Поведінковий аналіз, cloud-first підхід, аналітика.
SentinelOne	EPP+EDR	AI-підхід до виявлення загроз, rollback функції.
Sophos Intercept X	EDR+AI	Висока ефективність проти ransomware, експлойтів.
Trend Micro Apex One	EPP	Центральне управління, захист на основі політик.
OSQuery	Open-source	CLI-інструмент для Linux/macOS, телеметрія та аналіз.

2.2. Архітектура та основні компоненти Osquery

Osquery – це інструмент з відкритим вихідним кодом, який використовується для здійснення моніторингу стану операційних систем за допомогою SQL-запитів. Він широко використовується в кібербезпеці, зокрема для забезпечення прозорості кінцевих точок, збору телеметричних даних, аудиту системної активності та виявлення аномалій [22, 23].

Osquery – це інструмент, розроблений компанією Facebook (Meta), який забезпечує видимість кінцевих точок через SQL-подібний інтерфейс. Його основна мета – забезпечити уніфікований доступ до операційної інформації про систему у вигляді структурованих таблиць. Це зробило Osquery потужним інструментом в арсеналі фахівців з кібербезпеки, системних адміністраторів та DevOps-інженерів [22].

Основна ідея Osquery

Основна концепція Osquery полягає у представленні системної інформації (процеси, служби, мережеві з'єднання, файлова система, встановлене програмне забезпечення що) у вигляді таблиць, до яких можна отримати доступ за допомогою синтаксису, подібного до SQL:

```
osquery> SELECT name FROM processes LIMIT 5;
+----------------+
| name           |
+----------------+
| kernel_task    |
| launchd        |
| logd           |
| smd            |
| UserEventAgent |
+----------------+
```

Рис. 2.7. Приклад реалізації SQL–запиту

Це дозволяє отримувати важливу інформацію з операційної системи безпечним, уніфікованим і масштабованим способом без необхідності розробляти власні скрипти.

Архітектура Osquery

Osquery - це модульна клієнт-серверна архітектура, яка дозволяє розгортати окремі агенти та централізовану систему збору даних. Основні компоненти:

1. Osqueryi (інтерактивний інтерфейс)

- Інструмент CLI для інтерактивного виконання локальних запитів.
- Призначений для тестування, налагодження та ручного аналізу.
- Він не виконує фонових завдань і не зберігає дані.

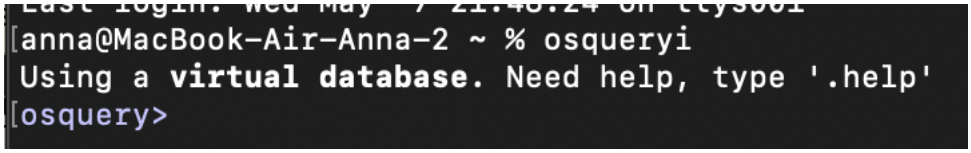

Рис. 2.8. Інтерактивний інтерфейс

2.Osqueryd （демон）

Основний агент, що виконує завдання фонового моніторингу.

Відповідальність за:

- Розклад запитів (scheduled queries): виконуються періодично в залежності від конфігурації;
- Детекцію змін (differential logging): записуються лише нові або змінені записи;
- Інтеграція з віддаленим бекендом (через TLS або інший API).

3. Table Facility (віртуальні таблиці)

Osquery використовує віртуальні таблиці, кожна з яких витягує дані за допомогою плагінів, що викликають API операційної системи [22]. Нижче наведено приклади віртуальних таблиць Osquery.

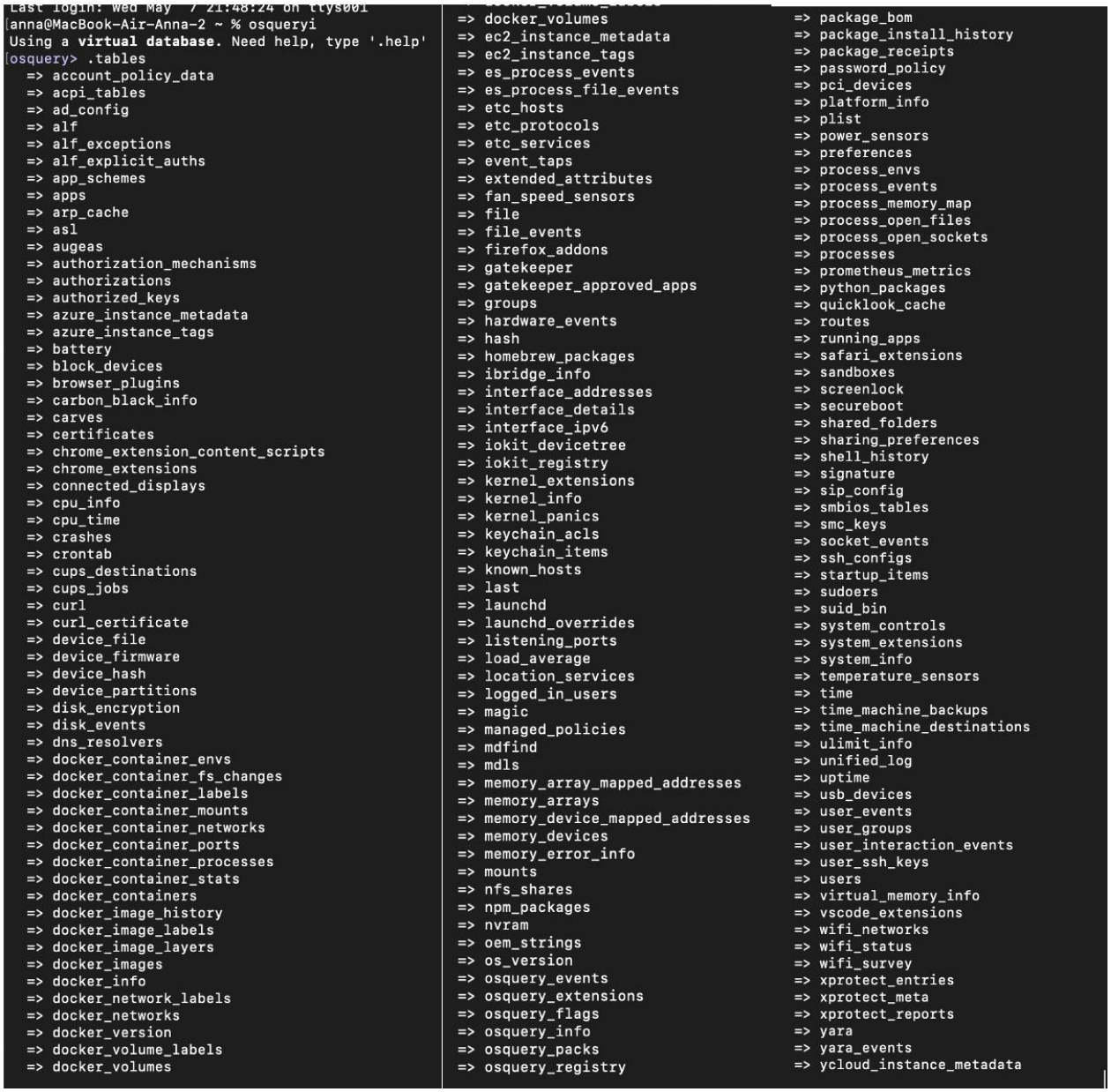

Рис. 2.9. Віртуальні таблиці Osquery

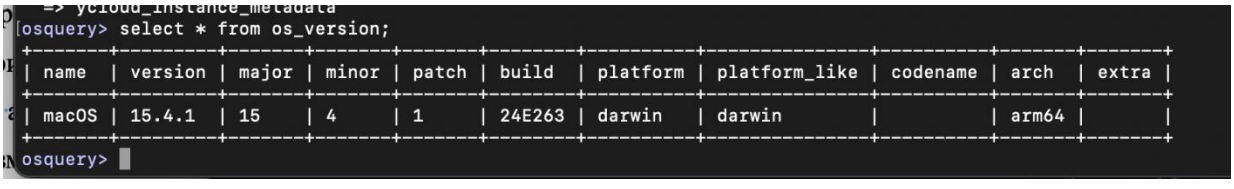

Рис. 2.10. Приклад таблиці Osquery

4. Pluggable Framework (плагіни)

Osquery має плагінну архітектуру, яка дозволяє розширювати його функціональність без зміни основного коду. Основними типами плагінів є

- Logger Plugins — передають лог-файли до зовнішніх систем (наприклад, Kafka, TLS, syslog);
- Config Plugins — відповідають за отримання конфігурацій (локально або з віддаленого сервера);
- Distributed Query Plugins — забезпечують централізоване управління запитами;
- Extension Plugins — дозволяють розширити функціональність за допомогою власних SQL-таблиць, написаних, наприклад, на Python чи C++.

5. Extension API

Основа для підключення сторонніх модулів. Може бути створена за допомогою:

- Нових віртуальних таблиць;
- Нових типів логерів;
- Інтеграції з SIEM, EDR, тощо [23].

6. Fleet / Colid / Doorman (управління)

Для централізованого керування агентами Osquery можна використовувати такі платформи, як [24]:

- Fleet — найпопулярніша open-source система керування Osquery;
- Kolide — SaaS-рішення з інтерфейсом для користувачів;
- Doorman — рішення для видачі запитів агентам у реальному часі.

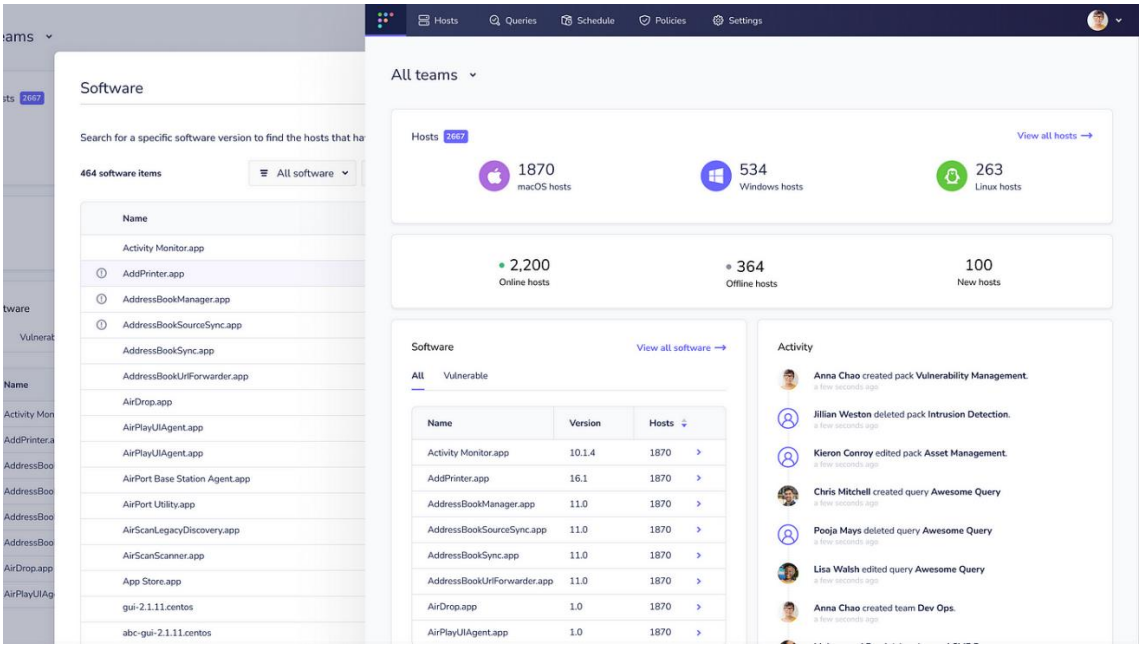

Рис. 2.11. Приклад інтерфейсу Fleet

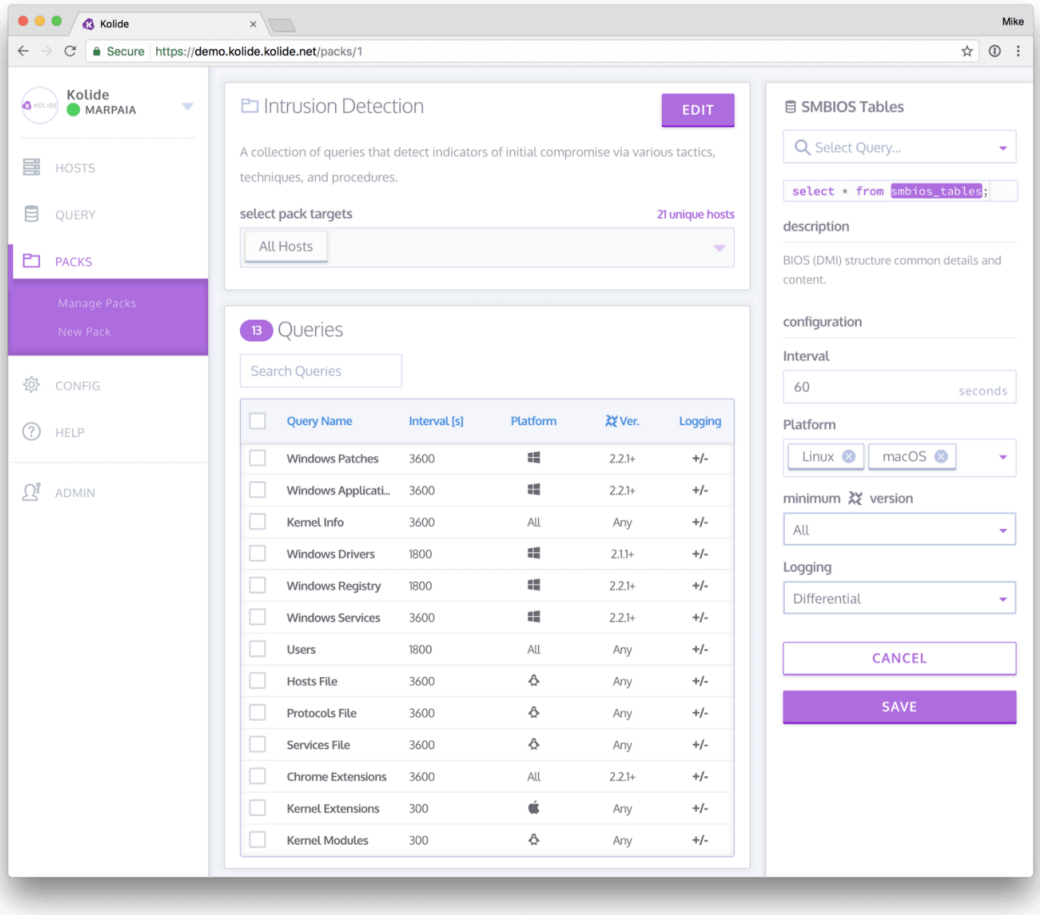

Рис. 2.12. Приклад інтерфейсу Kolide

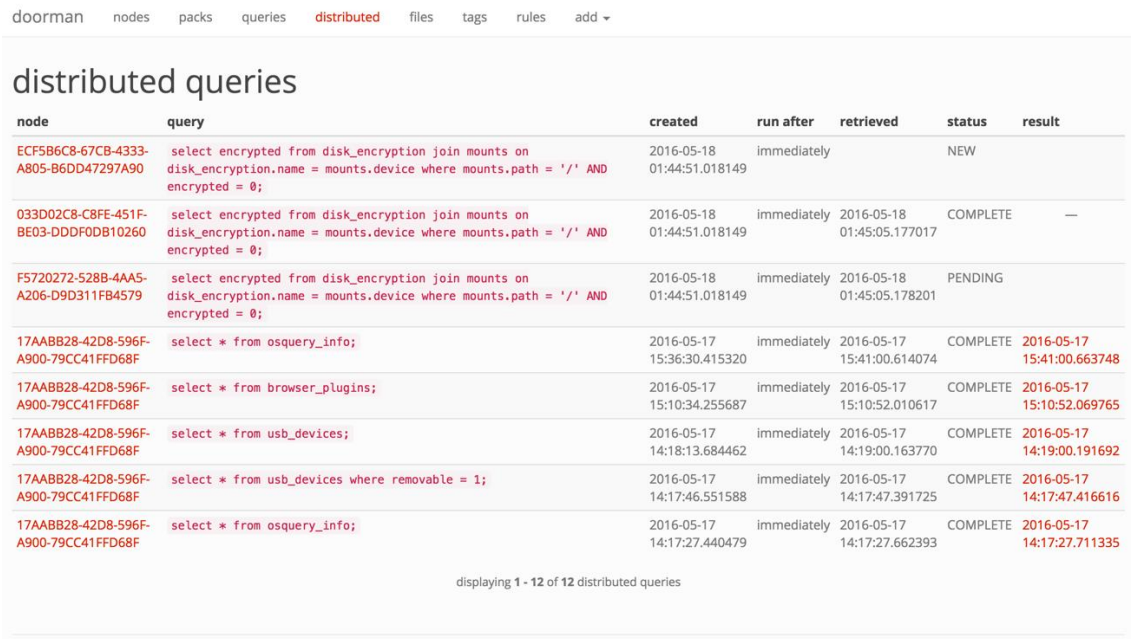

Рис. 2.13. Приклад інтерфейсу Doorman

Робочий цикл Osqueryd

1. 'osqueryd' завантажує конфігурацію з вказаного джерела.

2. Виконує планові запити відповідно до розкладу.

3. Визначає відмінності між поточними та минулими результатами.

4. Здійснюється передача даних через логер до зовнішніх систем (наприклад, Splunk, Kafka, Graylog).

5. Отримує розподілені запити з сервера керування (Fleet або Doorman) за необхідності.

Переваги Osquery

- Кросплатформеність: сумісний з Windows, macOS та Linux [22];
- Масштабованість: можливість централізовано керувати тисячами агентів;
- Гнучкість: підтримка настроюваних таблиць і логіки;
- Безпека: read-only доступ до системної інформації, ізоляція від критичних процесів [23].

2.3. Призначення та функції Osquery

Призначення Osquery

1. Моніторинг безпеки кінцевих точок

Osquery дозволяє відстежувати конфігурації, активність процесів, мережеві з’єднання, встановлені пакети, активних користувачів та інші аспекти роботи системи. Це необхідно для виявлення аномалій, ознак компрометації або відхилень від норми [25].

2. Аудит системної цілісності

Цей інструмент може порівнювати поточний стан таблиці з попередніми станами (використовуючи механізм differential logging), що дозволяє:

- Виявляти нові або змінені бінарні файли;
- Відстежувати зміни прав доступу;
- Реєструвати появу нових сервісів або відкритих портів.

3. Підтримка відповідності (compliance)

Інструмент Osquery широко використовується для контролю відповідності стандартам, таким як CIS Benchmarks, NIST 800-53, PCI DSS, HIPAA, оскільки він генерує звіти про стан системи та виявляє порушення політик на регулярній основі [26].

4. Live-розслідування інцидентів

Аналітики безпеки використовують інтерактивну версію Osquery (osqueryi) для оперативного дослідження інцидентів:

- Перегляду логів;
- Визначення джерел з'єднання;
- Аналізу запуску шкідливого програмного забезпечення;
- Виявлення lateral movement або privilege escalation.

5. Інвентаризація активів

Osquery автоматично збирає інформацію про програмне та апаратне забезпечення кінцевої точки:

- Версії ОС;
- Серійні номери;
- Стан шифрування диску;

- Наявність EDR/Patch-менеджерів тощо.

6. Централізоване керування флотом пристроїв

У зв’язці з рішеннями Fleet, Kolide або Doorman, Osquery дозволяє централізовано:

- Виконувати SQL-запити на великій кількості кінцевих пристроїв;
- Збирати відповіді в режимі реального часу;
- Проводити розслідування у рамках DFIR або MDR сервісів [26].

Ключові функціональні можливості Osquery

Таблиця 2.2

Функція	Опис
SQL-запити	Звернення до віртуальних таблиць, що відображають стан операційної системи
Планування запитів	Регулярне виконання запитів з логуванням результатів
Логування змін (differential)	Зберігання лише зміненої інформації для скорочення об’єму логів
Live Query	Виконання одноразових запитів до агентів у реальному часі
Плагінна система	Підтримка розширень: нові таблиці, логери, джерела конфігурацій
Кросплатформеність	Підтримка Linux, macOS, Windows
Реалтайм-моніторинг	Таблиці типу event-based (наприклад, *file_events, process_events*) [2]

Приклад практичного використання

```
osquery> select uid, gid, username, shell from users where uid >= 100;
+------------+------------+------------------------+----------------+
| uid        | gid        | username               | shell          |
+------------+------------+------------------------+----------------+
| 278        | 278        | _accessoryupdater      | /usr/bin/false |
| 263        | 263        | _analyticsd            | /usr/bin/false |
| 300        | 300        | _aonsensed             | /usr/bin/false |
| 273        | 273        | _appinstalld           | /usr/bin/false |
| 260        | 260        | _applepay              | /usr/bin/false |
| 235        | 235        | _assetcache            | /usr/bin/false |
| 245        | 245        | _astris                | /usr/bin/false |
| 294        | 294        | _audiomxd              | /usr/bin/false |
| 229        | 4294967294 | _avbdeviced            | /usr/bin/false |
| 288        | 288        | _avphidbridge          | /usr/bin/false |
| 291        | 291        | _backgroundassets      | /usr/bin/false |
| 289        | 289        | _biome                 | /usr/bin/false |
```

Рис. 2.14. Приклад демонстрації таблиць, де uid має значення менше 100

```
osquery> select * from cpu_time;
+------+---------+------+---------+----------+--------+-----+---------+-------+-------+------------+
| core | user    | nice | system  | idle     | iowait | irq | softirq | steal | guest | guest_nice |
+------+---------+------+---------+----------+--------+-----+---------+-------+-------+------------+
| 0    | 3193477 | 0    | 1448274 | 15284497 |        |     |         |       |       |            |
| 1    | 3003663 | 0    | 1344090 | 15588300 |        |     |         |       |       |            |
| 2    | 2595606 | 0    | 1143674 | 16213166 |        |     |         |       |       |            |
| 3    | 2379142 | 0    | 985052  | 16605294 |        |     |         |       |       |            |
| 4    | 1108720 | 0    | 250298  | 18656381 |        |     |         |       |       |            |
| 5    | 615425  | 0    | 153664  | 19252361 |        |     |         |       |       |            |
| 6    | 320424  | 0    | 81990   | 19624958 |        |     |         |       |       |            |
| 7    | 202687  | 0    | 47615   | 19781200 |        |     |         |       |       |            |
+------+---------+------+---------+----------+--------+-----+---------+-------+-------+------------+
```

Рис. 2.15. Приклад демонстрації таблиць для CPU Time

```
osquery> SELECT * FROM users WHERE uid < 1000;
+-----+-----+------------+------------+-------------------+-----------------------------------+-----------------------------+----------------+--------------------------------------+
| uid | gid | uid_signed | gid_signed | username          | description                       | directory                   | shell          | uuid                                 |
| is_hidden |
+-----+-----+------------+------------+-------------------+-----------------------------------+-----------------------------+----------------+--------------------------------------+
| 278 | 278 | 278        | 278        | _accessoryupdater | Accessory Update Daemon           | /var/db/accessoryupdater    | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000116-0 |
| 83  | 83  | 83         | 83         | _amavisd          | AMaViS Daemon                     | /var/virusmails             | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000053-0 |
| 263 | 263 | 263        | 263        | _analyticsd       | Analytics Daemon                  | /var/db/analyticsd          | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000107-0 |
| 300 | 300 | 300        | 300        | _aonsensed        | Always On Sense Daemon            | /var/db/aonsensed           | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA000012C-0  |
| 273 | 273 | 273        | 273        | _appinstalld      | App Install Daemon                | /var/db/appinstalld         | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000111-0 |
| 55  | 55  | 55         | 55         | _appleevents      | AppleEvents Daemon                | /var/empty                  | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000037-0 |
| 260 | 260 | 260        | 260        | _applepay         | applepay Account                  | /var/db/applepay            | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000104-0 |
| 87  | 87  | 87         | 87         | _appowner         | Application Owner                 | /var/empty                  | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000057-0 |
| 79  | 79  | 79         | 79         | _appserver        | Application Server                | /var/empty                  | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA0000004F-0 |
| 33  | 33  | 33         | 33         | _appstore         | Mac App Store Service             | /var/db/appstore            | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000021-0 |
| 67  | 67  | 67         | 67         | _ard              | Apple Remote Desktop              | /var/empty                  | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000043-0 |
| 235 | 235 | 235        | 235        | _assetcache       | Asset Cache Service               | /var/empty                  | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA000000EB-0 |
| 245 | 245 | 245        | 245        | _astris           | Astris Services                   | /var/db/astris              | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA000000F5-0 |
| 97  | 97  | 97         | 97         | _atsserver        | ATS Server                        | /var/empty                  | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000061-0 |
| 294 | 294 | 294        | 294        | _audiomxd         | Audio and MediaExperience Daemon  | /var/db/audiomxd            | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000126-0 |
| 229 | 4294967294 | 229 | -2         | _avbdeviced       | Ethernet AVB Device Daemon        | /var/empty                  | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA000000E5-0 |
| 288 | 288 | 288        | 288        | _avphidbridge     | Apple Virtual Platform HID Bridge | /var/empty                  | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000120-0 |
| 291 | 291 | 291        | 291        | _backgroundassets | Background Assets Service         | /var/empty                  | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000123-0 |
| 289 | 289 | 289        | 289        | _biome            | Biome                             | /var/db/biome               | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA00000121-0 |
| 93  | 93  | 93         | 93         | _calendar         | Calendar                          | /var/empty                  | /usr/bin/false | FFFFEEEE-DDDD-CCCC-BBBB-AA          |
```

Рис. 2.16. Приклад демонстрації таблиць локальних користувачів

2.4. Проблеми та обмеження інтеграції Osquery

Незважаючи на популярність Osquery як інструменту моніторингу кінцевих точок, його гнучкість, масштабованість і відкритий вихідний код, існує ряд проблем і обмежень, які виникають під час розгортання, інтеграції та застосування в корпоративному середовищі.

1. Відсутність нативної системи зберігання та аналітики даних

Osquery не є повноцінною SIEM/EDR системою, а лише агентом який збирає телеметричні дані. Самостійно він не забезпечує:

- Довгострокове зберігання логів;
- Кореляцію подій;

- Виявлення загроз на основі поведінки;
- Побудову діаграм, дашбордів чи візуалізацій.

Для цього необхідна стороння система, наприклад: Splunk, Elasticsearch, Wazuh, Graylog або FleetDM, що призводить до додаткових витрат та складнощів інтеграції [27].

2. Високий поріг входу для налаштування

Хоча синтаксис Osquery подібний до синтаксису SQL, він вимагає деякого розуміння, початкового розгортання та конфігурування:

- Поглиблене розуміння структури операційної системи;
- Створення специфічних конфігурацій (запити, правила логування, плагіни);
- Побудови інфраструктури з TLS-сервером або Fleet [28].

Це ускладнює використання невеликими командами без навичок DevOps.

3. Відсутність real-time обробки без використання додаткових модулів

Більшість запитів в Osquery працюють на основі опитування (polling) — тобто з часовими інтервалами (30 сек, 5 хв тощо). Це може призвести до:

- Затримки у виявленні загроз, особливо короткочасних процесів (наприклад, файл швидко створився, а потім зник);
- Необхідності використання event-based таблиць (`process_events`, `file_events`), які працюють лише на Linux/macOS і потребують підключення системних механізмів (auditd, FIM) [27].

4. Проблеми сумісності та стабільності на Windows

Історично склалось так, що Osquery краще функціонує на Linux/macOS. У середовищі Windows можуть виникати певні проблеми:

- Нестабільність деяких таблиць (`process_events`, `file_events`);
- Конфлікти з антивірусним програмним забезпечення або системними службами;
- Складність у використанні WMI чи реєстру через SQL [28].

Це обмежує його застосування у Windows-орієнтованих організаціях без додаткової обробки винятків.

5. Відсутність поведінкової аналітики (behavioral analysis)

На відміну від комплексних EDR-систем (наприклад, CrowdStrike, SentinelOne, Microsoft Defender for Endpoint), Osquery:

- Не здійснює аналіз поведінки процесів або динаміку їхньої взаємодії;
- Не має вбудованих сигнатур загроз, ML-модулів чи TTP-моделей;
- Не виявляє техніки на основі MITRE ATT&CK автоматично.

Це вимагає інтеграції з іншими аналітичними системами або ручного аналізу зібраної інформації.

6. Високе навантаження при неправильній конфігурації

Якщо запити Osquery налаштовані без урахування частоти, фільтрів і структури:

- Агент може спричиняти надмірне навантаження на CPU/диск;
- Лог-файли швидко переповнюють файлову систему;
- Спостерігається падіння продуктивності серверів або ноутбуків користувачів.

7. Обмежена кількість prebuilt запитів

На відміну від продуктів EDR або SIEM, Osquery:

- Не містить готових наборів запитів для виявлення загроз;
- Вимагає створення власних правил або використання зовнішніх репозиторіїв GitHub/ATT&CK Packs;
- Не містить вбудованої підтримки сценаріїв таких як: “Alert → Enrichment → Response”.

8. Потреба у зовнішніх засобах візуалізації

Оскільки Osquery не має інтегрованого GUI, наявна потреба у використанні сторонніх інструментів:

- Kibana (з використанням Elastic);
- Fleet (web-панель управління);
- Graylog;
- Grafana (з використанням Prometheus адаптерів).

Це ускладнює його використання "з коробки" аналітиками SOC [28].

Висновок до розділу 2

В даному розділі було розглянуто методи та інструменти захисту кінцевих точок, необхідні для забезпечення цілісності, конфіденційності та доступності корпоративної
IT-інфраструктури. Зокрема, приділено увагу антивірусним рішенням, системам виявлення вторгнень (IDS/IPS), багатофакторній автентифікації, контролі доступу та передових рішеннях EDR. Особливу увагу було приділено переходу від реактивного до проактивного та аналітичного підходу, інтеграції в SIEM-системи та використанню поведінкового моделювання.

Також, описано архітектуру та основні компоненти платформи моніторингу стану кінцевих точок з відкритим кодом Osquery. Архітектура Osquery, яка включає крос-платформні агенти, систему віртуальних таблиць, модуль програмування запитів і механізм розширення, дозволяє ефективно інтегрувати її в складні інфраструктури. Завдяки використанню SQL як основного інтерфейсу, Osquery забезпечує гнучкість і зрозумілість навіть для професіоналів без глибоких знань програмування.

До того ж описано функціональне призначення Osquery. Інструмент забезпечує безперервний моніторинг працездатності системи, аудит безпеки, розслідування інцидентів у режимі реального часу, перевірку відповідності та інвентаризацію активів. Також, наведені реальні приклади застосування, які демонструють його практичну цінність для SOC, DevOps та аналітиків безпеки.

Водночас, було зосереджено увагу на практичних проблемах використання Osquery. Зокрема, було виявлено такі обмеження: відсутність поведінкового аналізу, складність конфігурації, обмежена підтримка Windows, відсутність готових скриптів для виявлення загроз та необхідність інтеграції з іншими сервісами (Fleet, Elasticsearch, SIEM). Таким чином, незважаючи на відмінну функціональність, ефективне впровадження Osquery вимагає технічної експертизи та відповідної інфраструктури.

Загалом, Osquery – це потужний, але низькорівневий інструмент, який вимагає стратегічного підходу. У поєднанні з іншими рішеннями (наприклад, SIEM, EDR) він може стати невід’ємною частиною сучасної системи безпеки кінцевих точок.

3 РОЗРОБКА РЕКОМЕНДАЦІЙ ЩОДО ЗАХИСТУ КІНЦЕВИХ ТОЧОК З ВИКОРИСТАННЯМ OSQUERY НА ПЛАТФОРМІ MAC M1

3.1. Встановлення та налаштування Osquery

Системні вимоги

- Для інсталяції Osquery необхідно macOS 11 (Big Sur) або новішої версії.
- Також необхідно переконатись, що Osquery має дозволи в системі безпеки macOS (Security & Privacy → Full Disk Access).

Інсталяція Osquery

Інсталювати Osquery можна з офіційного сайту: https://www.osquery.io/downloads/official/5.16.0 двома способами:

1. За допомогою інсталяційних пакетів:

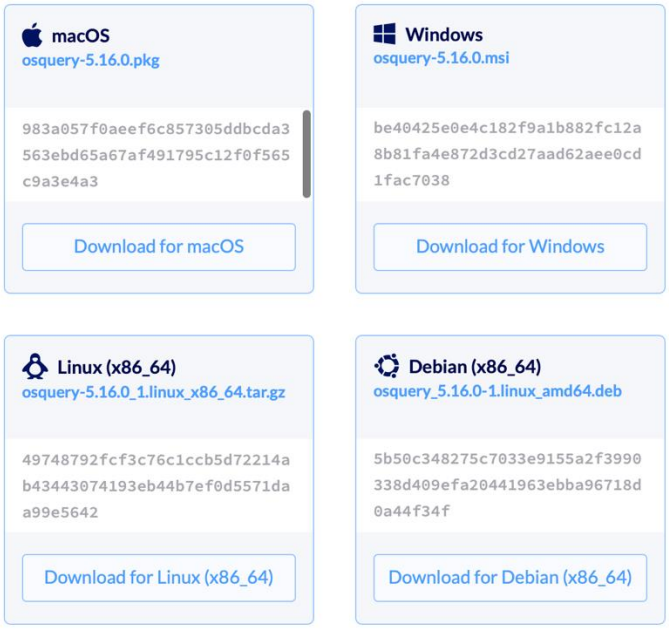

Рис. 3.1. Інсталяційні пакети Osquery

2. За допомогою терміналу та команди:

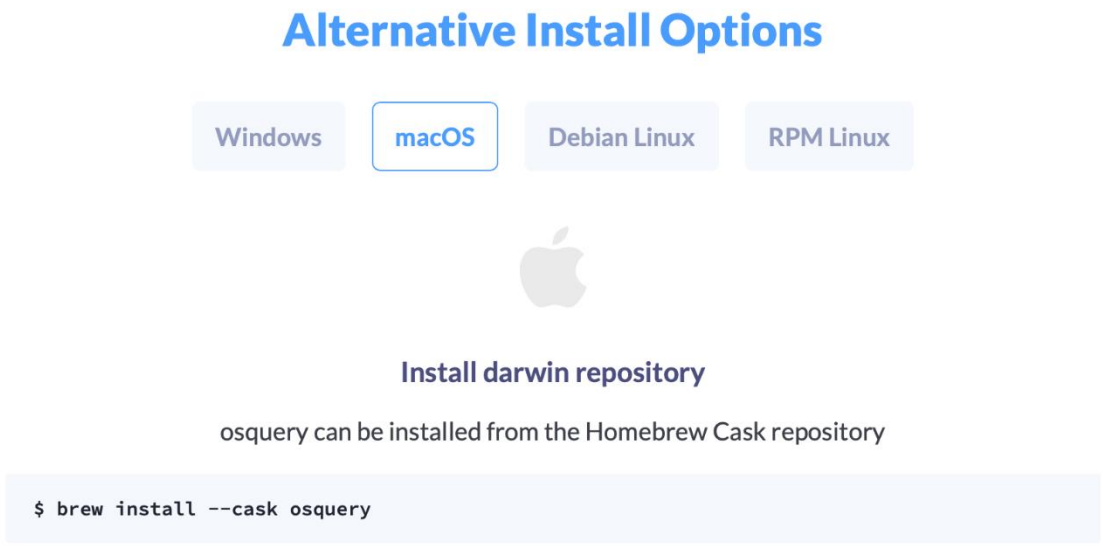

Рис. 3.2. Команда для інсталювання Osquery з використанням терміналу

Проте, перш ніж інсталювати необхідно переконатись в наявності Homebrew, та оновити систему використавши команду:

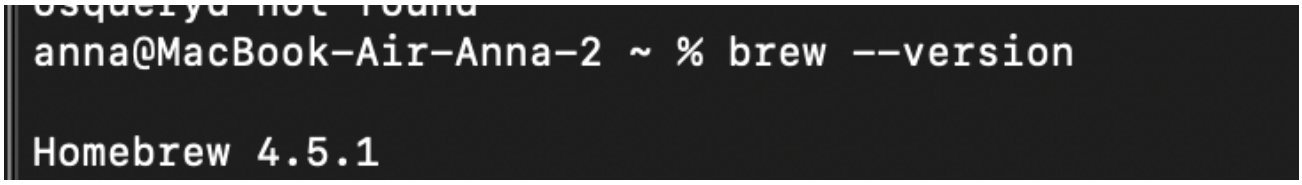

Рис. 3.3. Перевірка версії Homebrew

Рис. 3.4. Інсталяція Osquery за допомогою терміналу

Створення конфігураційного файлу

Конфігураційний файл – текстовий файл у форматі JSON.

Конфігураційний файл osquery.conf необхідний для визначення параметрів, запитів та функцій, які будуть активовані під час дії агента. Його основні завдання:

- Визначення налаштувань демона (osqueryd), шляхи до розміщення логів, способи логування та ідентифікатори хоста тощо.
- Визначення розкладу виконання SQL-запитів для моніторингу системи, наприклад, перевірка мережевих підключень, змін у файлах, процесів тощо.
- Визначення файлів чи каталогів, які необхідно відстежувати на наявність змін.
- Визначення додаткових пакетів, які необхідні для функціонування Osquery.

Основні компоненти JSON-файлу:

1. Options: містить глобальні налаштування роботи Osquery, такі як:

- Спосіб зчитування конфігурації: confid_plugin
- Спосіб логування та шлях до логів: logger_plugin, logger_path
- Ідентифікатор хоста: host_identifier
- Параметри безпеки та продуктивності: disable_logging, verbose, worker_threads, database_path, enable_syslog, schedule_splay_percent тощо.

2. Schedule: розділ для планування SQL-запитів, які будуть виконуватись автоматично з вказаною регулярністю. Для кожного запиту має бути вказано:

- Унікальна назва
- SQL-запит
- Інтервал у секундах (interval)
- Додаткові параметри.

3. Packs: додаткові набори запитів, які знаходяться в окремих файлах, для підключення типових сценаріїв моніторингу, наприклад, для контролю цілісності файлів, реагування на інциденти, відповідності стандартам тощо.

4. Decorators: додає додаткові SQL-запити, які виконуються одночасно з головними та додають метадані до кожного результату, наприклад, ім’я користувача, UUID хоста.

osquery.conf

```
{
        "options": {
                "host identifier": "hostname"
                "utc": "true"
},
"schedule": {
"users": {
        "query": "SELECT 'users' AS query_name, * from users;"
        "interval": 10
}},
"decorators": {
        "load": [
        "SELECT uuid AS host_uuid FROM system_info;",
        "SELECT hostname FROM system_info;"
        ]
    }
  }
```

Рис. 3.5. Приклад конфігураційного файлу

Сповіщення про виявлені аномалії та інциденти

Як зазначалось раніше, Osquery потребує інтеграції з іншими платформами. Тому для того, щоб отримувати сповіщення про виявлені аномалії, інциденти і т.д. необхідно здійснити інтеграцію з платформою FleetDM для централізованого управління Osquery агентами.

Для цього запустимо віртуальну машину Kali Linux та за допомогою терміналу встановимо Docker:

```
┌──(kali㉿kali)-[~]
└─$ sudo apt install docker-compose git -y
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
docker-compose is already the newest version (1.29.2-6).
git is already the newest version (1:2.43.0-1).
git set to manually installed.
```

Рис. 3.6. Команда для інсталяції docker-compose

Також, необхідно встановити Notejs для веб-інтерфейсу FleetDM.

Та за допомогою команди **git clone** завантажимо FleetDM:

Рис. 3.7. Команда для інсталяції fleet

Тепер, з допомогою команди **docker-compose up -d**, запускаємо контейнери, та переходимо в браузер та вводимо посилання: **https://localhost:8080:**

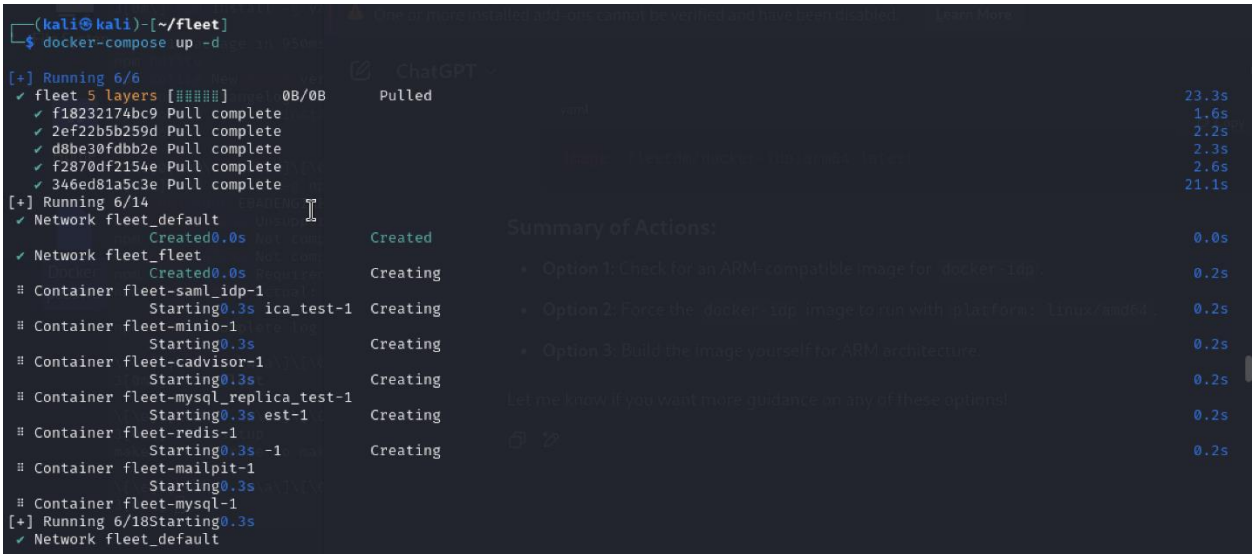

Рис. 3.8. Команда для запуску контейнерів

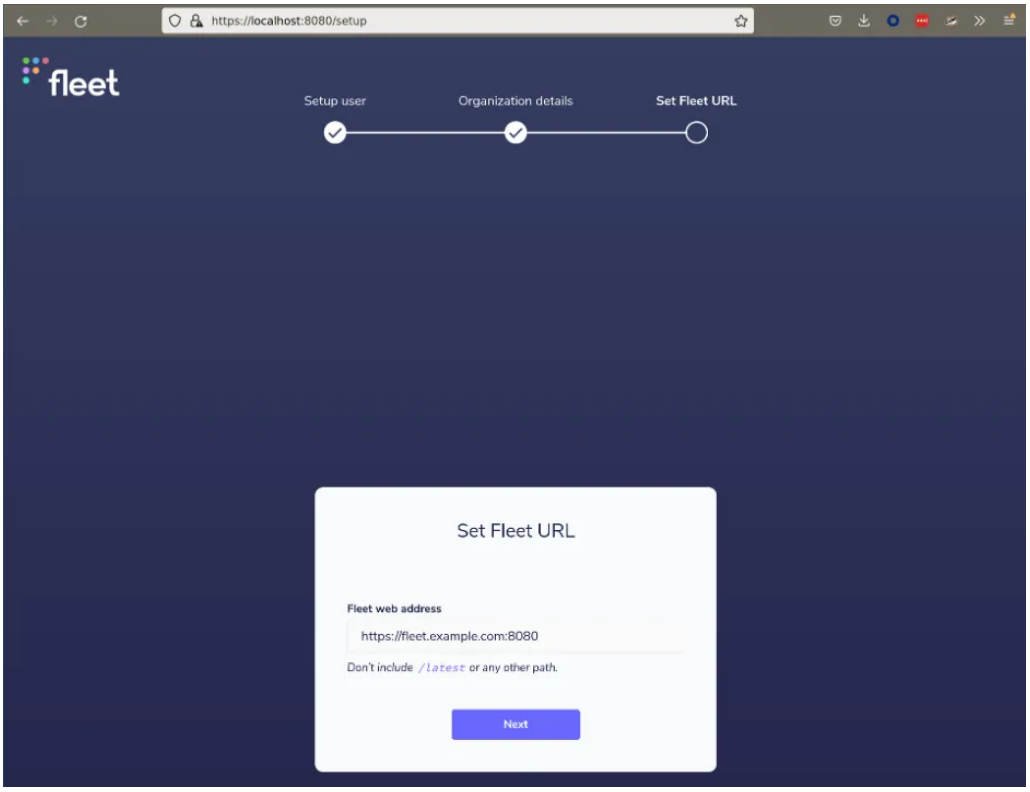

Рис. 3.9. Веб-інтерфейс fleet

Тепер, коли Fleet запущено, необхідно додати агента Osquery, для того щоб це зробити необхідно перейти до сторінки Hosts > Add hosts > Advanced, та завантажити CA сертифікат, також необхідно завантажити і файл секретів реєстрації та файл прапорів. Секрет реєстрації містить токен, який використовується Osquery для реєстрації на сервері FleetDM. Файл flags містить прапори запуску, які передаються бінарному файлу Osquery під час його запуску. Ці файли необхідно перенести в каталог etc/Osquery. Після того як всі файли та параметри будуть встановлені, необхідно перезапустити Osquery для застосування змін. Після чого, можна здійснювати моніторинг за допомогою веб-інтерфейсу.

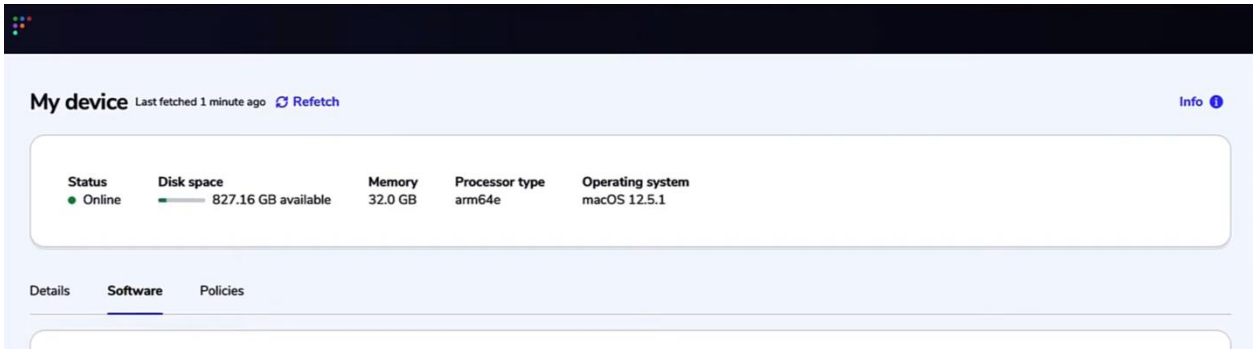

Рис. 3.10. Додано кінцеву точку в Fleet

У вкладці Policies можна налаштувати політики та у вкладці Schedule сформувати графік для застосування політик та у разі виявлення аномальної поведінки (або вразливості) буде сформовано звіт [29].

Рис. 3.11. Панель управління Fleet

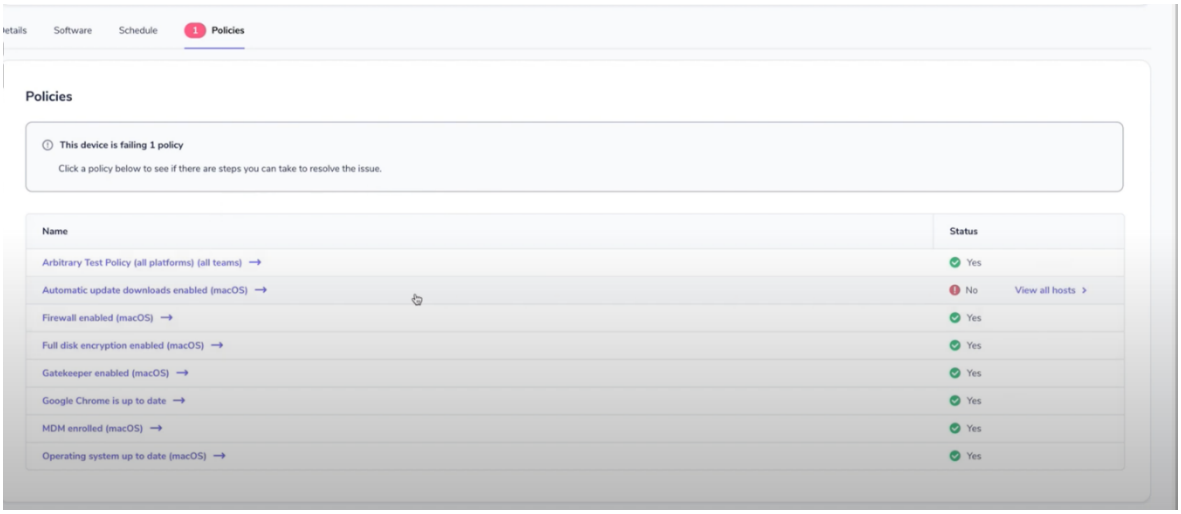

Рис. 3.12. Встановлені Політики

Для того щоб Політики виконувалось їх необхідно запланувати, тому у вкладці Schedule необхідно створити запит, як зображено на рисунку 3.25.

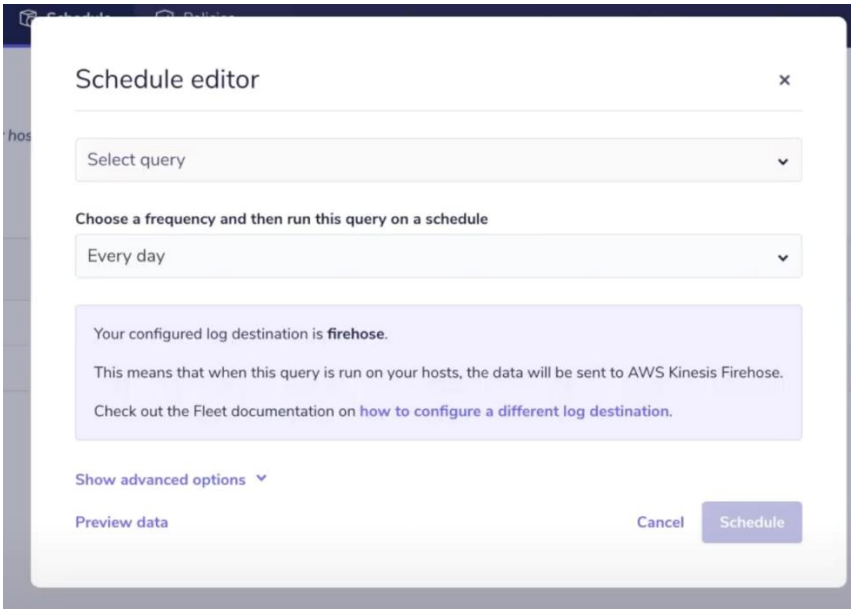

Рис. 3.13. Додавання Політик до розкладу

Після створення розкладу для Політик, вони будуть відображатись у таблиці, як зображено на рисунку 3.26.

Рис. 3.14. Заплановане виконання Політик

3.2. Результати застосування Osquery та порівняння ефективності з іншими EDR-рішеннями

Запуск конфігураційного файлу osquery.conf в терміналі

Для запуску osqueryd (демона) для здійснення моніторингу необхідно в терміналі ввести команду продемонстровану нижче:

```
anna@MacBook-Air-Anna-2 ~ % sudo osqueryd --verbose --config_path=/var/osquery/osquery.conf
```

Рис. 3.15. Команда для запуску osqueryd

Також, наявна потреба запуску osqueryd, з використанням sudo для доступу до всіх систем та сервісів

```
anna@MacBook-Air-Anna-2 ~ % sudo osqueryd --config_path=/var/osquery/osquery.conf --verbose
Password:
I0511 21:09:27.794216 -246231936 init.cpp:413] osquery initialized [version=5.17.0]
I0511 21:09:27.795468 -246231936 extensions.cpp:453] Could not autoload extensions: Cannot open file for reading: /var/osquery/extensions.load
I0511 21:09:27.795507 -246231936 dispatcher.cpp:78] Adding new service: WatcherRunner (0x600001730018) to thread: 0x16b4ff000 (0x600002a3c000) in process 37515
I0511 21:09:27.795874 1800400896 watcher.cpp:721] osqueryd watcher (37515) executing worker (37516)
I0511 21:09:27.803354 -246231936 init.cpp:410] osquery worker initialized [watcher=37515]
I0511 21:09:27.803879 -246231936 dispatcher.cpp:78] Adding new service: WatcherWatcherRunner (0x6000038c8018) to thread: 0x16db0b000 (0x6000001c4000) in process 37516
I0511 21:09:27.803933 -246231936 rocksdb.cpp:90] Opening RocksDB handle: /var/osquery/osquery.db
I0511 21:09:27.813652 -246231936 dispatcher.cpp:78] Adding new service: ExtensionWatcher (0x159611d88) to thread: 0x16dedf000 (0x6000001ca7f0) in process 37516
I0511 21:09:27.813688 -246231936 dispatcher.cpp:78] Adding new service: ExtensionRunnerCore (0x1596127d8) to thread: 0x16df6b000 (0x6000001ca6b0) in process 37516
I0511 21:09:27.813712 -246231936 auto_constructed_tables.cpp:107] Removing stale ATC entries
I0511 21:09:27.813726 1844883456 interface.cpp:299] Extension manager service starting: /var/osquery/osquery.em
E0511 21:09:27.826342 -246231936 config.cpp:967] updateSource failed to parse config, of source: /var/osquery/osquery.conf and content: {
        "options": {
                "host identifier": "hostname"
                "utc": "true"
},
"schedule": {
"users": {
        "query": "SELECT 'users' AS query_name, * from users;"
        "interval": 10
}},
"decorators": {
        "load": [
        "SELECT uuid AS host_uuid FROM system_info;",
        "SELECT hostname FROM system_info;"
        ]
    }
  }
```

Рис. 3.16. Результат запуску конфігураційного файлу

Оскільки у файлі, були виявлені певні помилки, що демонструє нам необхідність мати розуміння та знання в написанні JSON-файлів.

```
{
        "options": {
                "host identifier": "hostname",
                "utc": "true"
},
"schedule": {
"users": {
        "query": "SELECT 'users' AS query_name, * from users;",
        "interval": 10
}
},
"decorators": {
        "load": [
        "SELECT uuid AS host_uuid FROM system_info;",
        "SELECT hostname FROM system_info;"
        ]
    }
  }
```

Рис. 3.17. Виправлена версія конфігураційного файлу

Запускаємо osqueryd та отримуємо ось таку відповідь, що демонструє нам коректність конфігураційного файлу, та запуск постійного моніторингу системи відповідно до заданих параметрів:

```
I0511 21:12:00.279696 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:12:10.307780 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:12:20.347692 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:12:30.383940 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:12:40.426570 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:12:50.465672 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:13:00.546929 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:13:10.578045 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:13:20.616480 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:13:30.655977 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:13:40.690174 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:13:50.730306 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:14:00.762854 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:14:10.790226 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:14:20.834849 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:14:30.869746 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:14:40.907459 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:14:50.952229 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
I0511 21:15:00.991321 1874358272 scheduler.cpp:120] Executing scheduled query users: SELECT 'users' AS query_name, * from users;
```

Рис. 3.18. Моніторинг системи

SQL-запити для виявлення підозрілих дій

Також, додатково, можна здійснювати SQL-запити для виявлення підозрілих дій, нижче продемонстровані приклади таких запитів та результати цих запитів:

```
osquery> SELECT pid, name, path, start_time
    ...> FROM processes
    ...> WHERE name NOT LIKE '%osquery%' AND start_time > (strftime('%s', 'now') - 300);
```

Рис. 3.19. Запит для виявлення підозрілих процесів

```
+-------+----------------+-----------------------------------------------------------------------------------------------------------------------------------------------------+-
-----------+
| pid   | name           | path                                                                                                                                                |
start_time |
+-------+----------------+-----------------------------------------------------------------------------------------------------------------------------------------------------+-
-----------+
| 37570 | screencaptureui | /System/Library/CoreServices/screencaptureui.app/Contents/MacOS/screencaptureui                                                                     |
1746987301 |
| 37578 | mdworker_shared | /System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/Metadata.framework/Versions/A/Support/mdworker_shared |
1746987537 |
| 37579 | mdworker_shared | /System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/Metadata.framework/Versions/A/Support/mdworker_shared |
1746987537 |
| 37580 | mdworker_shared | /System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/Metadata.framework/Versions/A/Support/mdworker_shared |
1746987537 |
| 37581 | mdworker_shared | /System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/Metadata.framework/Versions/A/Support/mdworker_shared |
1746987537 |
| 37582 | mdworker_shared | /System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/Metadata.framework/Versions/A/Support/mdworker_shared |
1746987537 |
+-------+----------------+-----------------------------------------------------------------------------------------------------------------------------------------------------+-
```

Рис. 3.20. Результат виконаного запиту для виявлення підозрілих процесів

```
osquery> SELECT address, port, pid, protocol
    ...> FROM listening_ports
    ...> WHERE port NOT IN (22, 80, 443) -- стандартні порти
    ...> AND pid NOT IN (SELECT pid FROM processes WHERE name IN ('sshd', 'httpd', 'nginx'));
```

Рис. 3.21. Запит для виявлення нових або незвичних портів

```
+-----------+-------+-------+----------+
| address   | port  | pid   | protocol |
+-----------+-------+-------+----------+
| 0.0.0.0   | 58101 | 978   | 6        |
| ::        | 58101 | 978   | 6        |
| ::        | 3722  | 978   | 17       |
| 0.0.0.0   | 0     | 990   | 17       |
| 0.0.0.0   | 0     | 990   | 17       |
| 0.0.0.0   | 0     | 1015  | 17       |
| 0.0.0.0   | 58798 | 1015  | 17       |
| 0.0.0.0   | 64385 | 1024  | 17       |
| 0.0.0.0   | 7000  | 1166  | 6        |
| ::        | 7000  | 1166  | 6        |
| 0.0.0.0   | 5000  | 1166  | 6        |
| ::        | 5000  | 1166  | 6        |
| 127.0.0.1 | 44950 | 18517 | 6        |
| 127.0.0.1 | 44960 | 18517 | 6        |
|           | 0     | 26618 | 0        |
+-----------+-------+-------+----------+
```

Рис. 3.22. Результат виконаного запиту для виявлення нових або незвичних портів

```
osquery> SELECT * FROM last LIMIT 10;
```

Рис. 3.23. Запит для виявлення останніх виконаних дій

username	tty	pid	type	type_name	time	host
anna	ttys000	35213	7	user-process	1746984872	
anna	ttys000	35053	8	dead-process	1746983775	
anna	ttys000	35053	7	user-process	1746983775	
anna	ttys000	34004	8	dead-process	1746960854	
anna	ttys000	34004	7	user-process	1746960854	
anna	ttys000	33913	8	dead-process	1746959621	
anna	ttys000	33913	7	user-process	1746959621	
anna	ttys000	30060	8	dead-process	1746816865	
anna	ttys000	30060	7	user-process	1746816865	
anna	ttys000	29989	8	dead-process	1746816611	

Рис. 3.24. Результат виконаного запиту для виявлення останніх виконаних дій

```
Error: no such column: path
osquery> SELECT pid, name, path, cmdline
    ...> FROM processes
    ...> WHERE cmdline NOT LIKE '%/bin/%' -- запуски без системного шляху
    ...> AND name NOT IN ('sshd', 'bash', 'init', 'osqueryd');
```

Рис. 3.25. Запит для виявлення неавторизованих спроб запуску програм

pid	name	path	cmdline
0	kernel_task		
1	launchd	/sbin/launchd	
550	logd	/usr/libexec/logd	
551	smd	/usr/libexec/smd	

Рис. 3.26. Результат виконаного запиту для неавторизованих спроб запуску програм

Здійснення моніторингу кінцевих точок з допомогою веб-інтерфейсу Fleet

Результати виконання Політик відображаються у вигляді таблиці, як зображено на рисунку 3.27.

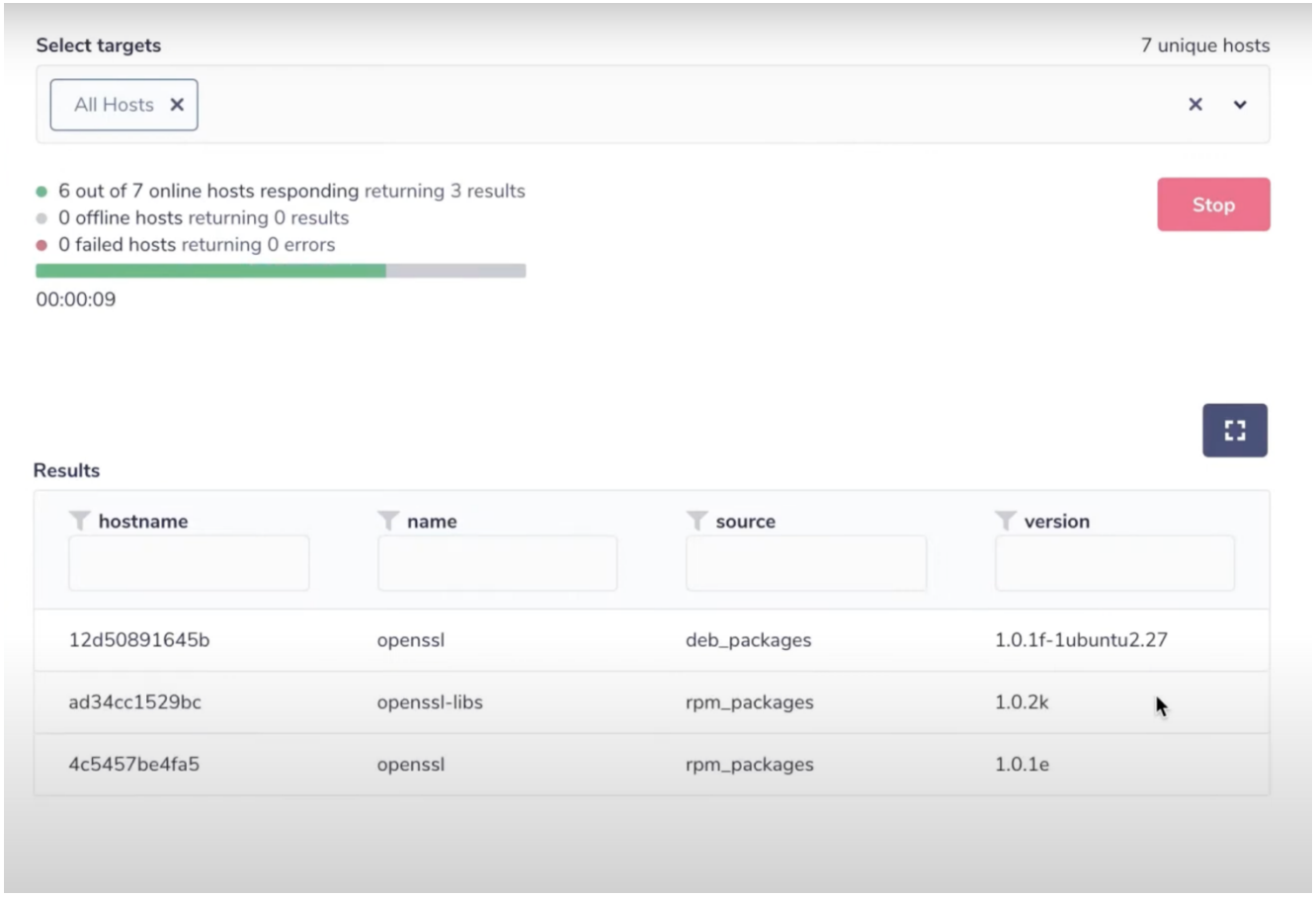

Рис. 3.27. Результати виконаних запланованих Політик

На сторінці кінцевого пристрою відображається, які Політики на нього поширюються. Якщо буде виявлена не відповідність Політиці на кінцевому пристрої, то на вкладці цього пристрою буде відображено дану проблему, як зображено на рисунку 3.24.

Також, за допомогою результатів виконання запланованих Політик можна виявляти не відповідності на кінцевому пристрою, та з допомогою додаткових засобів захисту здійснювати подальше розслідування.

Зробивши порівняння Osquery та інших рішень (наприклад, CrowdStrike Falcon, SentinelOne, Bitdefender) можна дійти до висновку:

Osquery дуже гнучка платформа, яку можна налаштовувати відповідно до потреб Організації. Платформа розгортається зручними для користувача способами та не потребує особливих характеристик операційної системи. Також, її перевагою є сумісність з ARM-архітектурою. Ще однією перевагою, є можливість виконання запланованих запитів завдяки побудові конфігураційних файлів та

запуску демона (osqueryd). Завдяки інтеграції платформи з Fleet можна здійснювати централізоване управління декількома кінцевими точками та налаштовувати сповіщення про виявлені аномалії в системах цих кінцевих точок.

Дане рішення ідеально підходить для Організацій з обмеженим бюджетом та з потребою в кастомізації системи під її вимоги. Проте для більшої ефективності наявна необхідність інтеграції з іншими платформами для повноцінної EDR-функціональності.

Проте під час встановлення та налаштування додаткових систем (наприклад, Fleet) виникали певні проблеми та труднощі. Перш за все, це встановлення систем на платформу mac m1, оскільки дана платформа має ARM-архітектуру, що ускладнює встановлення та налаштування систем, на відміну від операційної системи Windows, що має ARD-архітектуру, та є сумісною майже з усіма системами.

Також, для того щоб коректно налаштувати Fleet необхідно мати:

- Знання операційної системи Linux.
- Знання та навички використання терміналу.
- Навички та досвід з написання yml-файлів.
- Знання та навички розгортання контейнерів.

Врахувавши вище згадані необхідні знання та навички, можна дійти до висновку, що для здійснення централізованого управління агентами Osquery є потреба в глибокому розумінні технологій та процесів IT.

3.3. Рекомендації щодо захисту кінцевих точок з використанням Osquery на платформі Mac M1

З урахуванням вимог NIST CSF 2.0 та з використанням Osquery Організації рекомендується:

Використовуючи Osquery та платформу для централізованого управління Fleet рекомендовано здійснювати автоматизовану інвентаризація апаратних та

програмних активів, вказуючи при цьому версію операційних систем, встановлені додатки та конфігурації.

Наприклад: Створення запланованих запитів для здійснення аудиту системи (наприклад, перевірка шифрування диска) для виявлення пристроїв, процесів, які не відповідають налаштованим політикам безпеки.

Організації рекомендовано здійснювати моніторинг стану даних та контролю доступу до конфіденційних даних.

Наприклад: Здійснювати перевірку списків авторизованих SSH-ключів для запобігання неавторизованого доступу.

Організації рекомендовано налаштувати Osquery для моніторингу систем на аномальну поведінку.

Наприклад: Моніторинг систем на виявлення запуску невідомих або привілейованих процесів, змін в системних файлах або каталогах, підозрілих мережевих з’єднань.

З урахуванням вимог ISO 27001 та з використанням Osquery Організації рекомендується:

Організації рекомендовано налаштувати Osquery з метою централізованого збору логів та їх подальше збереження. Це необхідно для проведення аудиту та здійснення розслідування у випадку виявлення інцидентів.

Наприклад: Інтеграція Osquery та SIEM-систем для автоматизації процесу аналізу логів.

Організації рекомендовано використовувати Osquery для постійного відстеження оновлень безпеки та відстеження наявності застарілих версій програмного забезпечення.

З урахуванням вимог CIS Controls та з використанням Osquery Організації рекомендується:

Організації рекомендовано щогодини відстежувати встановлені сторонні додатки, системні сервіси та запущені процеси, стан шифрування диска.

Організації рекомендовано здійснювати аудит списків користувачів, груп та привілеїв на наявність неактивних або непотрібних облікових записів, налаштування sudo-прав.

Організації рекомендовано відстежувати зміни в критичних файлах (наприклад, в конфігураційних), здійснювати аналіз журналів системних подій (наприклад, журнал автентифікації, журнал підключення змінних носіїв).

Для забезпечення вищого рівня безпеки кінцевих точок рекомендується заборонити модифікацію системних файлів та каталогів, обмежити доступ додатків до камери, мікрофону та локації за допомогою MDM-профілів.

Також, рекомендується розробити політики безпеки в Організації, які визначають відповідальних за конфігурацію Osquery та визначають частоту проведення аудитів за допомогою Osquery.

Висновок до розділу 3

В даному розділі було детально описано процес впровадження Osquery як інструменту захисту кінцевих точок на платформі Mac M1. За допомогою коректної підготовки середовища, створення конфігураційного файлу та використання SQL-запитів Організації мають можливість налаштувати гнучкий і масштабований моніторинг системних подій, процесів і мережевої активності. Налаштування тригера моніторингу забезпечує безперервний збір даних для раннього виявлення потенційних загроз.

Аналіз результатів використання Osquery показав, що інструмент може ефективно доповнити традиційні рішення EDR, особливо в багатоплатформних середовищах, де потрібен високий ступінь кастомізації та прозорості. Osquery надає глибоке розуміння кінцевих точок, дозволяє створювати індивідуальні сценарії моніторингу і може бути інтегрований з існуючими системами безпеки для підвищення загальної ефективності захисту.

Рекомендації щодо захисту кінцевих точок на основі Osquery для платформи Mac M1 включають проведення регулярного аудиту системи, моніторинг облікових записів користувачів, відстеження змін критично важливих файлів і

мережевої активності, а також інтеграцію з іншими системами (наприклад, SIEM) для автоматизації аналізу подій. Враховуючи специфіку архітектури платформи Mac та дотримання найкращих практик кібербезпеки, Osquery може максимально широко використовуватися в Організаціях.

ВИСНОВКИ

Osquery зарекомендував себе як потужний і гнучкий інструмент моніторингу безпеки кінцевих точок, що дозволяє детально відстежувати системні події, процеси, мережеву активність та інвентаризацію пристроїв за допомогою SQL-запитів. Архітектура, що включає крос-платформні агенти, систему віртуального робочого столу та механізм розширення, забезпечує ефективну інтеграцію в складні корпоративні інфраструктури, включаючи Mac M1. Osquery дозволяє Організаціям отримувати глибоке розуміння стану пристроїв, виявляти аномалії на ранніх стадіях і проводити аудит безпеки в режимі реального часу, що є критично важливим в умовах зростаючих кіберзагроз, таких як фішинг, програми-вимагачі, DDoS і APT, що є критично важливим в сучасних умовах.

Однак використання Osquery має певні труднощі. Налаштування та підтримка системи вимагає значних технічних знань, особливо для новачків, і потребує достатніх ресурсів для обробки великих обсягів даних. Відсутність автоматизованих механізмів реагування на інциденти та поведінкового аналізу обмежує її ефективність без інтеграції з іншими рішеннями безпеки, такими як SIEM, Fleet, Elasticsearch. Крім того, специфікації платформи Mac M1 накладають додаткові вимоги до конфігурації, які можна подолати, дотримуючись найкращих практик і регулярно проводячи аудит системи.

Osquery – це безкоштовне рішення з відкритим вихідним кодом, що робить його привабливим для Організацій з великими ресурсами, обмеженим бюджетом і потребою в індивідуальному моніторингу. Інтеграція з іншими інструментами безпеки підвищує загальний рівень захисту кінцевих точок і дозволяє перейти від реактивного підходу до кібербезпеки до проактивного та аналітичного. Таким чином, Osquery є перспективним елементом сучасних стратегій безпеки, особливо для великих організацій, які можуть надати технічну експертизу та інфраструктуру для його ефективного впровадження та експлуатації.

Таким чином, Osquery на платформі Mac M1 – це потужний інструмент моніторингу та захисту кінцевих точок, який при правильному налаштуванні та

інтеграції може значно покращити інформаційну безпеку організації, забезпечуючи гнучкість, деталізацію та видимість даних про стан системи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SecurityScorecard. The Rise of Endpoint Security Risks: 6 Common Types. URL: https://www.linkedin.com/pulse/rise-endpoint-security-risks-6-common-types-security-scorecard.

2. Uptycs Threat Research Team. Osquery Guide. URL: https://www.uptycs.com/blog/threat-research-report-team/osquery-guide.

3. 50 Endpoint Security Stats You Should Know. Expert Insights. URL: https://expertinsights.com/insights/50-endpoint-security-stats-you-should-know/.

4. CrowdStrike. Osquery – Overview. URL: https://www.crowdstrike.com/en-us/cybersecurity-101/it-automation/osquery/.

5. Open Source EDR Solutions. PacketLabs Blog. URL: https://www.packetlabs.net/posts/open-source-edr-solutions/.

6. How Osquery Works and How To Use It. Uptycs. URL: https://www.uptycs.com/osquery-how-it-works-and-how-to-use.

7. 16 Types of Endpoint Security Risks To Watch Out For. Teramind Blog | Content For Business. URL: https://www.teramind.co/blog/endpoint-security-risks/.

8. Top Endpoint Security Risks to Watch in 2025 and How to Mitigate Them. CyberUpgrade. URL: https://cyberupgrade.net/blog/cybersecurity/top-endpoint-security-risks-to-watch-in-2025-and-how-to-mitigate-them/.

9. Endpoint Security Risks. SentinelOne Cybersecurity 101. URL: https://www.sentinelone.com/cybersecurity-101/endpoint-security/endpoint-security-risks/.

10. UH.ua. Встановлення Osquery на VPS. URL: https://uh.ua/ua/kb/vps/software-installation/install-osquery.html.

11. Інститут проблем програмування НАН України. Інформаційні технології. URL: https://ippi.org.ua/sites/default/files/2024-1.pdf.

12. Містерман І.М. Інтелектуальна система виявлення загроз на основі Osquery : бакалаврська кваліфікаційна робота. Тернопільський національний технічний університет імені Івана Пулюя, 2024. URL:

https://elartu.tntu.edu.ua/bitstream/lib/45833/2/Bachelor_Thesis_SB41_Misterman_I_M_2024.pdf .

13. Endpoint Security Statistics: Key Trends And Threats. SpyHunter for Mac. URL: https://www.spyhunter.com/shm/endpoint-security-statistics/.

14. Are Open-Source EDR Efficient? Дискусія на Reddit. URL: https://www.reddit.com/r/AskNetsec/comments/11emub3/are_opensource_edr_efficient/.

15. Tools to make smarter, faster decisions. *Gartner*. URL: https://www.gartner.com/en/tools.

16. MITRE ATT&CK®. *MITRE ATT&CK®*. URL: https://attack.mitre.org.

17. Microsoft Defender for Endpoint documentation - Microsoft Defender for Endpoint. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/.

18. The CrowdStrike Falcon® platform. *crowdstrike.com*. URL: https://www.crowdstrike.com/platform/.

19. Singularity™ XDR AI-Plattform – SentinelOne. *SentinelOne DE*. URL: https://www.sentinelone.com/platform/.

20. NIST Computer Security Resource Center | CSRC. *NIST Computer Security Resource Center | CSRC*. URL: https://csrc.nist.gov.

21. ISO/IEC 27001:2022. *ISO*. URL: https://www.iso.org/standard/27001.

22. Welcome to osquery - osquery. *Welcome to osquery - osquery*. URL: https://osquery.readthedocs.io/en/stable/.

23. Announcing osquery 5: Now with EndpointSecurity on macOS. *The Trail of Bits Blog*. URL: https://blog.trailofbits.com/2021/11/10/announcing-osquery-5-now-with-endpointsecurity-on-macos/.

24. Fleet | Better with Fleet. *Fleet | Open device management for everyone*. URL: https://fleetdm.com/better.

25. What are the current pain points of osquery?. *The Trail of Bits Blog*. URL: https://blog.trailofbits.com/2017/12/21/osquery-pain-points/.

26. Fleet | Using Fleet | Fleet documentation. *Fleet | Open device management for everyone*. URL: https://fleetdm.com/docs/using-fleet.

27. The Trail of Bits Blog. *The Trail of Bits Blog*. URL: https://blog.trailofbits.com.

28. Fleet. Why Fleet. Fleet documentation. *Fleet. Open device management for everyone*. URL: https://fleetdm.com/docs.

29. Fleet | Automations. *Fleet | Open device management for everyone*. URL: https://fleetdm.com/guides/automations#policy-automations