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

КАФЕДРА СИСТЕМ ТА ТЕХНОЛОГІЙ КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Технологія захисту кінцевих точок від несанкціонованого доступу на базі SIEM-системи Wazuh»

зі спеціальності *125 Кібербезпека та захист інформації*
(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

_____ Богдан ЯЦЕНТІЙ
(підпис)

Виконав: здобувач вищої освіти групи БСДМ-63
ЯЦЕНТІЙ Богдан
(прізвище, ім'я)

Керівник д-р філос., доц., МАРЧЕНКО Віталій
(науковий ступінь, вчене звання, прізвище, ім'я)

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

Київ 2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

XDR	–	Extended Detection and Response
SOC	–	Security Operating Center
IOC	–	Indicator of compromise
TLS	–	Transport Layer Security
JSON	–	JavaScript Object Notation
SQL	–	Structured Query Language
API	–	Application Programming Interface
CTI	–	Cyber Threat Intelligence
SIEM	–	Security Information and Event Management
EDR	–	Endpoint Detection and Response
RBAC	–	Role-Based Access Control
UEBA	-	User and Entity Behavior Analytics
PID	-	Process Identifier
CIS	-	Center for Internet Security
NIST	-	National Institute of Standards and Technology
MTTD	-	Mean time to detection
MTTR	-	Mean time to reaction

ВСТУП

Актуальність дослідження. У сучасних умовах цифрової трансформації організацій та стрімкого розвитку інформаційних технологій питання кібербезпеки набуває особливої значущості. Одним із найвразливіших елементів корпоративної IT-інфраструктури є кінцеві точки — пристрої, що безпосередньо взаємодіють з користувачами та є першою лінією контакту з зовнішнім середовищем. Саме через них найчастіше здійснюються атаки, що призводять до витоку конфіденційної інформації, порушення роботи систем та фінансових втрат.

Традиційні антивірусні рішення вже не здатні ефективно протистояти сучасним кіберзагрозам, які використовують складні техніки обходу захисту, соціальну інженерію, zero-day вразливості та багатовекторні атаки. У цьому контексті особливого значення набуває впровадження багаторівневих систем захисту, здатних забезпечити моніторинг, виявлення та реагування на інциденти безпеки в режимі реального часу.

Одним із сучасних рішень, що відповідає зазначеним вимогам, є платформа Wazuh — SIEM-система з відкритим кодом, яка дозволяє реалізувати ефективну систему захисту кінцевих точок з високою масштабованістю, гнучкістю налаштувань та можливістю інтеграції з іншими інструментами безпеки. Wazuh забезпечує збір телеметрії, аналіз подій безпеки, виявлення аномалій, формування політик реагування та централізований моніторинг IT-інфраструктури.

Об'єкт дослідження – процес забезпечення кібербезпеки кінцевих точок у корпоративному середовищі.

Предмет дослідження – технологія побудови системи захисту кінцевих точок на базі платформи Wazuh.

Мета роботи – розробити порядок впровадження Wazuh для захисту кінцевих точок організації, дослідити її функціональні можливості та сформулювати рекомендації щодо практичного застосування в корпоративній IT-інфраструктурі.

Наукові завдання:

Дослідити сутність проблеми захисту кінцевих точок у кібербезпеці.

- Проаналізувати сучасні підходи та інструменти для побудови системи захисту.
- Визначити обмеження традиційних рішень та виклики, пов’язані з масштабуванням і складністю загроз.
- Дослідити можливості Wazuh для моніторингу, кореляції подій та реагування на інциденти.
- Розробити технологію впровадження системи захисту кінцевих точок та запропонувати рекомендації для її застосування у практиці кіберзахисту.
- Розкрити порядок реалізації технології забезпечення безпечної роботи гібридних працівників організації.

Методи дослідження – аналіз наукової та технічної літератури, огляд архітектури Wazuh, експериментальне впровадження агентів Wazuh, налаштування правил виявлення та реагування, тестування ефективності системи, а також порівняння з міжнародними практиками у сфері кібербезпеки.

Практичне значення одержаних результатів - Запропонована технологія дозволяє ефективно захищати кінцеві точки організації, зменшити час реагування на інциденти, підвищити якість виявлення загроз і надати фахівцям із кібербезпеки гнучкий інструмент для моніторингу, аналізу та реагування.

Результати кваліфікаційної роботи апробовані на Всеукраїнській науковій конференції «Актуальні проблеми кібербезпеки», яка відбулася 29 жовтня 2025 року в Державному університеті інформаційно-комунікаційних технологій, м. Київ.

1 ДОСЛІДЖЕННЯ ПРОБЛЕМИ ЗАХИСТУ КІНЦЕВИХ ТОЧОК ВІД НЕСАНКЦІОНОВАНОГО ДОСТУПУ

1.1. Дослідження проблеми захисту кінцевих точок та аналіз ризиків пов’язаних з несанкціонованим доступом

Кінцеві точки давно вийшли за межі ролі «терміналів» і стали вузлом, де сходяться ідентичність користувача, конфігурація пристрою, дані, мережевий контекст і бізнес-функції. Розмивання периметра через хмаризацію, гібридну зайнятість і модель BYOD означає, що доступ до корпоративних ресурсів ініціюється за межами «фортеці» — з дому, дороги, коворкінгу, з особистих ноутбуків і смартфонів. Саме тому несанкціонований доступ (НСД) до кінцевої точки перестав бути локальною технічною проблемою й став організаційним ризиком першого порядку. Емпіричні огляди загроз у ЄС фіксують домінування ідентифікаційних і соціотехнічних векторів, наполегливу експлуатацію відомих уразливостей та «кампанійність» дій зловмисників: атаки розгортаються не одиничним експлойтом, а послідовностями кроків — початковий доступ, закріплення, підвищення привілеїв, латеральний рух, ексфільтрація. У цій динаміці саме endpoint стає найзручнішою точкою входу, бо поєднує людський фактор із великим обсягом локальних секретів (ключі, токени, кеші), а також із конфігураційними «дірками» у ПЗ, що оновлюється нерівномірно [1].

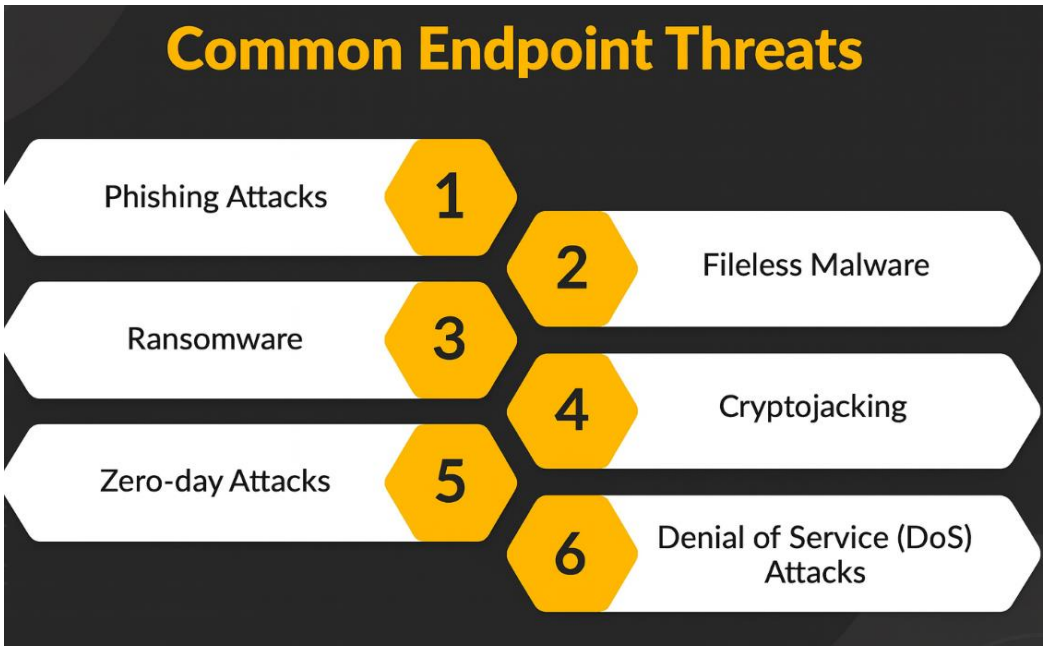

Рис. 1.1. Типові атака на кінцеві точки

Якщо зіставити технічні та поведінкові виміри проблеми, видно, що НСД на кінцевих точках рідко виглядає як «миттєвий злам». Частіше це повільна, «малошумна» кампанія. Спочатку — фішинговий ланцюжок, який краде пароль або cookie/токен сесії; далі — створення стійкості на пристрої через легітимні механізми автозапуску, «розумне» маскування під сервіс; потім — збір локальних облікових даних, секретів і конфігурацій; зрештою — латеральний рух до цінніших цілей. Тактика living-off-the-land дозволяє зловмиснику діяти штатними утилітами ОС і вписуватися у «нормальний шум» адміністрування. Саме тому класичні сигнатурні підходи (антивірус, «чорні списки») дедалі частіше не дають раннього сигналу: подія схожа на звичайну роботу адміністратора або сервісу. Ситуацію ускладнює гетерогенність парку пристроїв: Windows, Linux, macOS, iOS, Android, периферійні IoT і навіть OT-вузли мають різні моделі безпеки, цикли патчів і журналювання — забезпечити рівномірну спостережність без централізованих сенсорів практично неможливо [1].

Рамка контролів NIST сформулювала відповідь на цю багатовимірну загрозу через системний каталог керованості ідентичностей, доступу, журналювання, конфігурацій і безперервного моніторингу. Суть — не у «чарівному продукті», а у відтворюваності: однакові класи пристроїв зводяться до «золотих» еталонів

конфігурації, мінімізуються привілеї, автентифікація підсилюється багатофакторністю, усі значущі дії фіксуються у журналах і центрально корелюються. Принцип найменших привілеїв «зменшує радіус вибуху» у разі компрометації; сегментація обмежує латеральний рух; керований change-менеджмент і аудит конфігурацій знижують шанс «тихої» підміни політик; безперервний моніторинг перетворює набір розрізнених подій на історію інциденту, яку можна вчасно розпізнати. Контролі родин AC/IA/AU/CM/SI з SP 800-53 прямо адресують специфіку endpoint: від ідентифікації та доступу до журналювання і цілісності системи; ключовим є те, що телеметрія з кінцевої точки мусить надходити централізовано — інакше пристрій перетворюється на «чорну скриньку», де НСД може довго залишатися непоміченим [2].

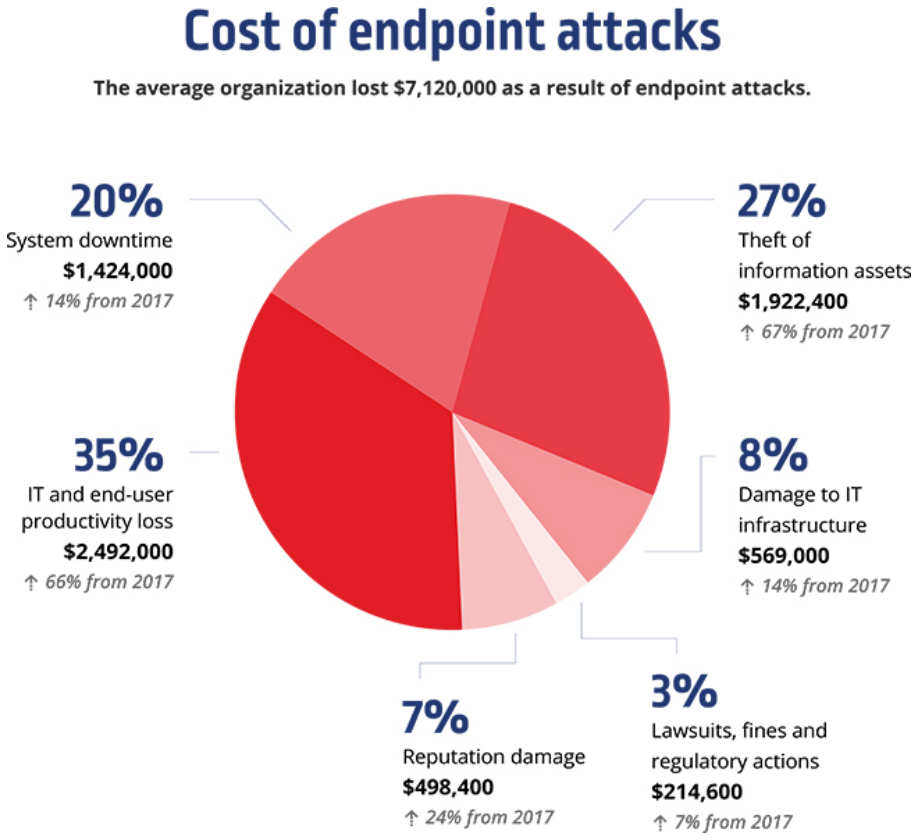

Рис 1.2 Об’єм завданої шкоди при успішній реалізації атаки

Практична площина цих принципів — у пріоритизованому чек-листі CIS Controls v8. Цей фреймворк не стільки додає «ще один список хороших справ», скільки пропонує послідовність мінімально необхідних кроків, щоб endpoint перестав бути хаотичною сумою налаштувань. Спочатку — точний інвентар

апаратури й ПЗ, без якого неможливий ні патч-менеджмент, ні контроль дрифтів конфігурації; далі — керування уразливостями та конфігураціями; потім — захист ідентичностей (MFA, контроль привілеїв, розмежування адміністраторських доступів), політики даних, постійний моніторинг і валідація. «Офенс інформз дефенс»: пріоритети вибудовано з огляду на реальну поведінку нападників — саме ці кроки найчастіше відрізають їм шлях або принаймні різко підвищують їхню «вартість атаки». У контексті НСД на кінцевих точках такий набір — це операційна азбука: без інвентаря немає об’єкта захисту, без патчів — вразливі компоненти для ескалації, без MFA — «другий фактор» не зупинить reuse пароля, без журналювання — нікому зшивати слабкі сигнали у сценарій [3].

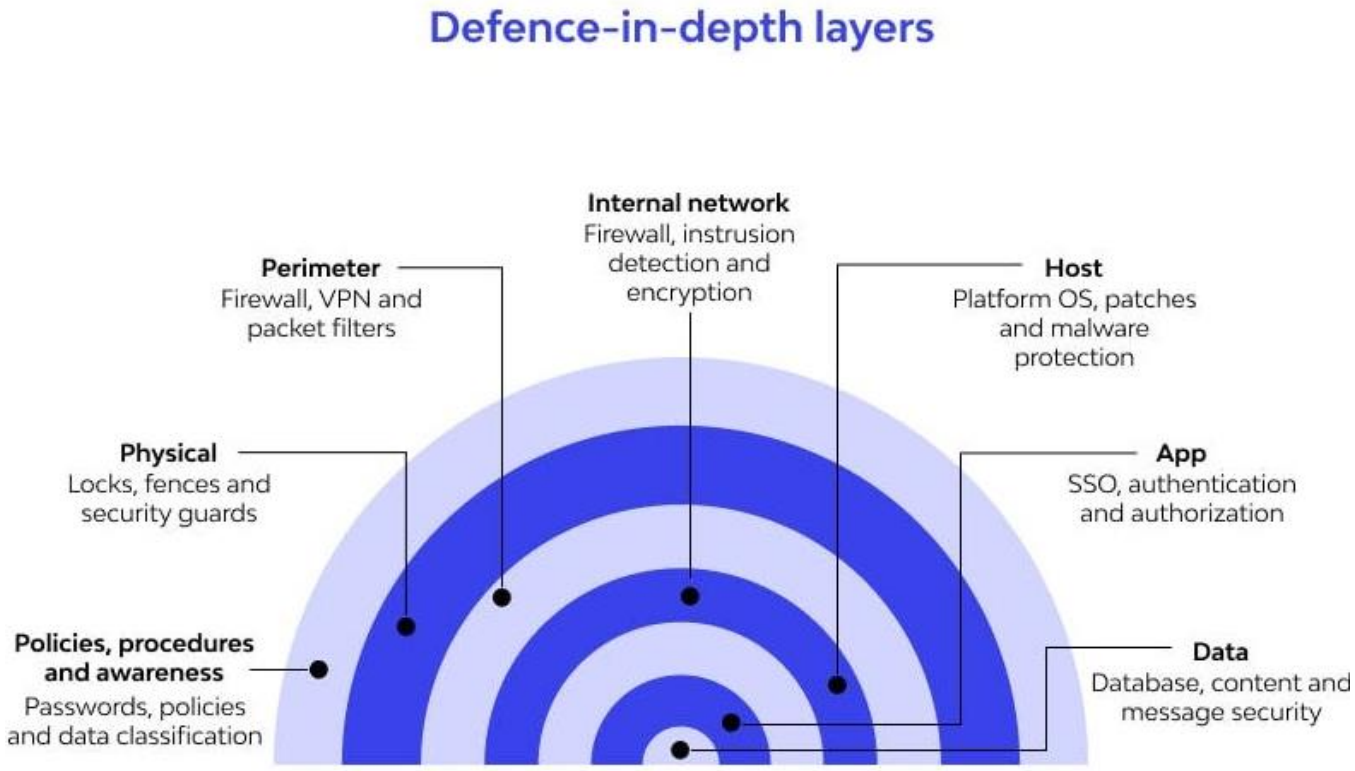

Рис 1.3 Модель Defense-in-Depth

Ключовий наслідок цієї логіки — потреба у спостережності (observability) кінцевих точок і кореляції подій у масштабі всього середовища. Тут на перший план виходять SIEM/XDR-платформи, здатні збирати, нормалізувати й пов’язувати журнали автентифікації, системні події, мережеві факти, сигнали про уразливості та конфігураційні зміни. У відкритому стеку саме Wazuh є показовою відповіддю:

агент на endpoint збирає телеметрію ОС, журнали безпеки, інвентар ПЗ і параметри конфігурації, а сервер аналізу декодує події, застосовує кореляційні правила, індексує результати та робить їх видимими через дашборд. Найважливішим для теми НСД є те, що Wazuh не обмежується пасивним лог-колектом: модуль контролю цілісності (File Integrity Monitoring, FIM) формує «еталон» контрольних сум і атрибутів для визначених файлів і каталогів та сигналізує про будь-який дрейф — створення, модифікацію, видалення. Коли зміна стається, агент надає контекст — хто, коли й яким процесом ініціював дію — і це дозволяє відрізнити адміністративну операцію від «тихої» підміни, типової для НСД [4].

З практичної перспективи FIM часто є «першою сиреною» щодо НСД, бо зачіпає те, що нападник намагається зробити непомітно: додати ключ до authorized_keys у профілі користувача, «безшумно» увімкнути RDP, змінити політику SSH/sshd, підмінити довірені сертифікати, прописати новий сервіс автозапуску чи змінити реєстр/служби у Windows. У відриві від контексту така поодинока подія не є інцидентом. Але у Wazuh вона одразу корелюється з журналами логонів (час/місце/метод), невдалими спробами автентифікації, появою нетипових процесів і зовнішніми індикаторами компрометації. Саме ця «зшивка» робить із набору слабких сигналів верифікований сценарій НСД, з яким уже можна працювати у реальному часі. Далі увімкнеться кероване стримування (Active Response): автоматичне блокування IP, завершення процесу, відключення або ізоляція вузла до завершення розслідування. Таким чином, оборона переходить із реактивної «після події» у проактивну «на місці події» — прямо на кінцевій точці [4].

Не менш важливий другий стовп — керування уразливостями та «дрифтами» конфігурацій. Агент Wazuh інвентаризує встановлене ПЗ і версії компонентів (syscollector), а модуль виявлення уразливостей співставляє їх із базами CVE. Навіть якщо первинний доступ відбувся, наявність непатченого драйвера, бібліотеки або служби дає нападнику легку «сходинку» для ескалації привілеїв. Поєднання цієї інформації з FIM створює цикл контролю: якщо критично вразливий компонент не може бути негайно пропатчений (операційне вікно,

сумісність), політики доступу тимчасово обмежуються, а моніторинг навколо відповідних файлів/служб підсилюється; при ознаках зловживання кореляційне правило запускає автоматичну відповідь. У регуляторному вимірі це прямо лягає на контролі SP 800-53, а у практичному — відповідає пріоритизації CIS. Wazuh документовано асоціює FIM із релевантними NIST-контролями (зокрема родиною SI/CM), що полегшує відповідність і доказову базу для аудитів [4].

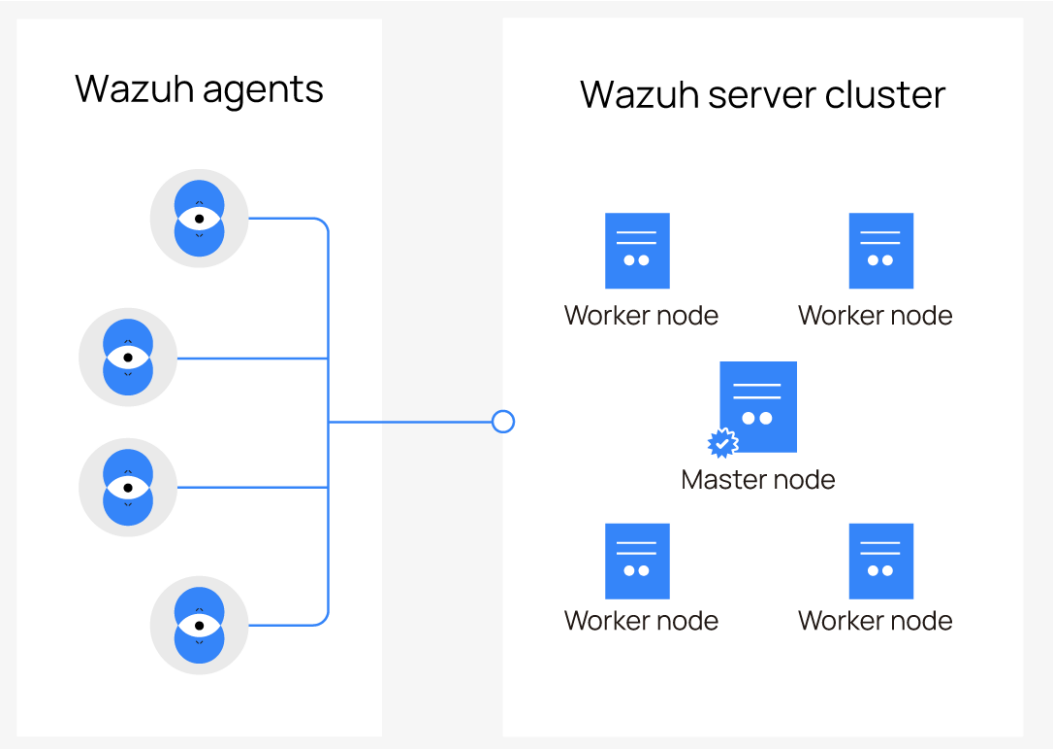

Рис 1.4 Архітектура Wazuh-agent

Поза «ядром» сенсорики ключовим для успіху є операційний підхід: дисципліна конфігурацій (базові еталони для класів пристроїв), керований життєвий цикл змін (усе через change-заявки), валідований доступ (MFA, just-in-time і just-enough-access замість постійних привілеїв), регулярний аналіз журналів та пост-інцидентний розбір. Це той самий міст між «стандартами» і «практикою», який у CIS описаний як послідовний перехід від інвентаризації та гігієни до моніторингу та реагування. Без цього навіть найкраща SIEM/XDR залишиться «телескопом без астронома»: дані є, але спостереження і дії — епізодичні. У зрілих командах метрики MTTD/MTTR, частка інцидентів, виявлених у межах однієї сесії користувача, середній час до ізоляції вузла, відсоток пристроїв із «золотими»

конфігами і з критичними CVE — стають не звітними KPI, а робочими інструментами пріоритетизації змін [3].

Окремого аналізу потребують класи кінцевих точок. На робочих станціях користувачів основним «паливом» атак є фішинг і крадіжка токенів/куків браузера; тут FIM корисний для відслідковування дрібних конфіг-змін і «прописування» стійкості, а журнали автентифікації — для виявлення аномальної географії й часових патернів. На серверах картина інша: значущими стають зміни конфігурацій сервісів, створення нових службових обліковок, підміна бінарів чи бібліотек; тут FIM разом із контрольованими еталонами конфігурацій критично знижує «тихий час» зловмисника. На мобільних пристроях ризики частіше пов'язані з компрометацією ідентичності через мобільний фішинг, підроблені застосунки й викрадення токенів; у таких середовищах без MDM/UEM, політик шифрування, блокування рут/джейлбрейків і умовної автентифікації (device posture) питання НСД не розв'язати. Для IoT/OT-вузлів проблема ускладнюється ще й відсутністю повноцінного агента або оновлень: тут доводиться комбінувати мережевий моніторинг, «тонкі» сенсори та сегментацію, а контроль цілісності налаштувань часто стає єдиним «якорем» спостережності [1].

Загрози ідентичності заслуговують на окреме висвітлення, бо у більшості інцидентів на endpoint зловмисник діє «під виглядом» легітимного користувача. MFA не є «срібною кулею», але вона на порядок підвищує вартість атаки, особливо якщо поєднана з контекстними політиками доступу (географія, ризиковість сесії, стан пристрою). Підміна браузерних сесій і AiTM-схеми (adversary-in-the-middle) піднімають планку вимог: довіряти слід не тільки факту логіну, а й «здоров'ю» пристрою в момент доступу (шифрування диска, актуальність патчів, відсутність критичних CVE, цілісність критичних конфігів). Це спонукає поєднувати контроль доступу з телеметрією endpoint: якщо FIM чи інвентар конфігурації бачать дрейф, політики доступу мають реагувати — від примусової повторної автентифікації до тимчасової блокування чутливих дій [4].

Архітектурно впровадження Wazuh під задачі НСД доцільно робити як «сенсорний килим»: усі керовані endpoints отримують агент; для «напівкерованих»

(BYOD, партнери) визначається мінімальний рівень телеметрії або ізольовані шлюзи доступу; для серверів та критичних вузлів — розширений набір правил FIM і кореляції. Базова лінія формується на «чистих» конфігураціях, після чого будь-який дрейф — привід для автоматизованої перевірки: чи є заявка на зміну, чи збіглися часові вікна, хто ініціював, чи не збіглася подія з нетиповим логіном. Правила кореляції збирають ці маркери в «сюжети»: наприклад, «новий локальний адміністратор + зміна політики RDP + логін із нової ASN поза робочим часом» — це не три дрібні події, а кандидат на ізоляцію вузла і блокування облікової до з’ясування [4].

Важливим нюансом є управління хибнопозитивними спрацьовуваннями. Надмірна чутливість FIM та кореляції швидко виснажує SOC: «шум» демотивує аналітиків і знижує якість реагування. Тому правила мають бути тісно прив’язані до change-процесу: якщо зміна санкціонована і відповідає вікну, спрацювання «приглушується» або маркується як очікуване; якщо зміна не має заявки або події «розійшлися у часі», тривога підвищує пріоритет. Тут же доцільно використовувати «етикетки ризику» для пристроїв і користувачів: BYOD із мінімальним контролем, хости з відкладеними патчами, облікові із розширеними правами — усе це множники ваги приоритезації інцидентів [3].

Рис 1.5 File integrity monitoring process

Нарешті, питання конфіденційності й відповідності. Моніторинг кінцевих точок неминуче торкається персональних даних і контекстів поведінки користувачів. Вирішується це політиками мінімізації даних, прозорими повідомленнями, технічними обмеженнями на доступ до вмісту файлів (моніторинг метаданих і цілісності замість інспекції контенту), сегрегацією ролей у SOC та аудитами доступу до журналів. Доброю практикою є документація відповідностей: які контролі SP 800-53 і які пункти CIS забезпечуються якими саме можливостями Wazuh (FIM, вразливості, журналювання, кореляція, Active Response). Це знімає питання «чому ми збираємо саме ці події» і полегшує зовнішні перевірки [2].

Узагальнюючи, проблема НСД на кінцевих точках — це не «вірус на ПК», а системна динаміка, що живиться сумішшю людського фактору, конфігураційних дрейфів, непатчених компонентів і легітимних інструментів ОС. Відповідь на неї не зводиться до єдиного засобу, а складається з чотирьох взаємопідсилювальних площин: стандарти й контролі (щоб знати, що й як вимірювати), практичні фреймворки пріоритизації (щоб робити правильні кроки в правильній послідовності), сенсорика й аналітика (щоб бачити і розуміти події) та операційна

дисципліна (щоб дані ставали діями). У цій архітектурі Wazuh надає саме ту спостережність на endpoint, без якої жодна Zero-Trust-політика не працює на практиці: FIM показує «коли і що змінилося», журналювання — «хто і як увійшов», вразливості — «де слабко», кореляція — «як це пов’язано», а Active Response — «що автоматично зробити зараз». І саме так НСД перетворюється з прихованої загрози, яка роками визріває на краю мережі, на керований операційний ризик із коротким життєвим циклом: побачити — інтерпретувати — ізолювати — усунути — навчитися [1].

1.2. Аналіз типових кіберзагроз, спрямованих на отримання доступу до конфіденційної інформації

У сучасному цифровому світі найціннішим ресурсом є не програмне забезпечення, не апаратне обладнання і навіть не канали комунікації, а саме інформація. Дані стали основою управління бізнесом, прийняття рішень, розробки нових продуктів, контролю над технологіями й навіть інструментом політичної влади. Саме тому несанкціонований доступ до конфіденційної інформації сьогодні є одним із головних викликів кібербезпеки. Від найменшої приватної компанії до урядових структур держави — усі стикаються з загрозами, спрямованими на викрадення, модифікацію або публікацію критичних даних. За оцінками ENISA Threat Landscape 2024, понад 80% усіх атак прямо чи опосередковано мають за мету саме отримання даних, а не руйнування інфраструктури [5].

Рис 1.6 Інфографіка ENISA з топ загроз

Цей факт свідчить про глибинну трансформацію природи кіберзлочинності. Якщо ще 20 років тому більшість атак були зумовлені бажанням викликати хаос чи продемонструвати технічну майстерність, то сьогодні атаки стали комерційно мотивованими. Зловмисники мислять категоріями ринку: дані мають ціну, попит і пропозицію. Вони можуть бути продані на даркнет-форумах, використані для корпоративного шантажу, монетизовані через вимагання (як у випадку ransomware), або застосовані для конкурентної чи політичної боротьби.

Аналіз типових загроз показує, що існує кілька основних класів атак, які найчастіше використовуються для доступу до даних. Це, насамперед, фішинг та соціальна інженерія, використання валідних облікових даних, експлуатація

програмних уразливостей, атаки програм-вимагачів (ransomware) та інсайдерські загрози. Кожен із цих класів є надзвичайно складним у виявленні, має свою історію розвитку, механізми реалізації та практичні приклади, що доводять їхню небезпечність.

Фішинг залишається найпоширенішим і найбільш результативним методом отримання несанкціонованого доступу до даних. Він працює тому, що націлений не на машини, а на людей. Людська довіра, психологічний тиск, поспіх, страх чи жадоба — усе це використовують зловмисники, аби змусити жертву виконати дію: відкрити шкідливе вкладення, перейти за посиланням чи ввести свій пароль на підробленому сайті.

У минулому фішинг мав вигляд примітивних листів, однак сьогодні ситуація докорінно змінилася. Сучасні кампанії надзвичайно персоналізовані: зловмисники аналізують профілі у соцмережах, публічні бази даних, використовують інформацію з попередніх витоків, аби створити максимально достовірне повідомлення. Особливим різновидом є spearfishing (цільовий фішинг), коли атака спрямована на конкретну людину, наприклад, керівника відділу або адміністратора систем. Якщо таку атаку здійснює держава або висококваліфікована група, її часто називають whaling — полювання на «велику рибу».

За даними Verizon DBIR 2025, близько 36% усіх зламаних систем починають шлях саме з фішингових атак [6]. І що ще небезпечніше — майже половина співробітників хоча б раз клікають на підозріле посилання. Навіть якщо компанія впроваджує багатофакторну автентифікацію, нові методики (AiTM, consent phishing, MFA fatigue) дають змогу обходити й цей бар’єр. У реальності це означає: жодна технічна система не може гарантувати повний захист, якщо людина не навчена розпізнавати маніпуляції.

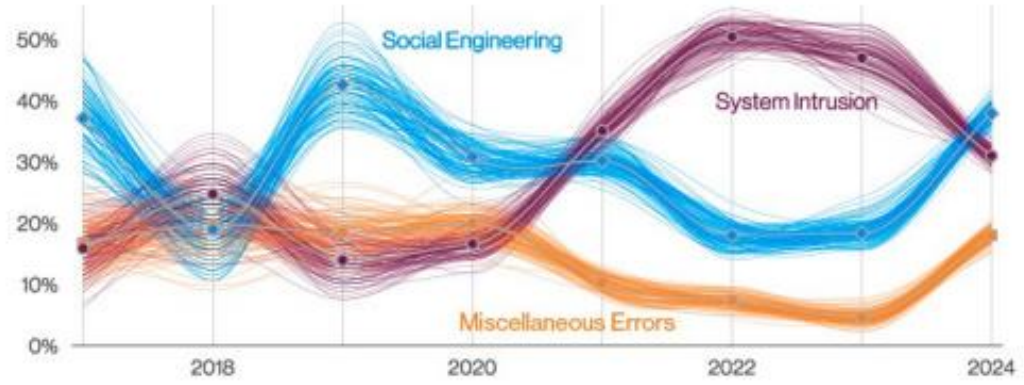

Рис 1.7 Графік Verizon DBIR з векторами атак

Іншим дуже поширеним шляхом отримання даних є використання чинних акаунтів. Це робить атаку «невидимою», адже дії виглядають як звичайна активність користувача. Паролі можуть бути викрадені через інфостілери, куплені на даркнеті, отримані зі злитих баз інших сервісів, де співробітники використовували однакові облікові дані. Злочинці активно користуються практикою повторного використання паролів, і ця звичка працівників стає для них золотим ключем.

Звіт Mandiant M-Trends 2025 свідчить, що «valid accounts» сьогодні є найпопулярнішою технікою забезпечення довготривалого доступу до корпоративних мереж [7]. Час перебування зловмисника у системі може обчислюватися місяцями, а іноді й роками. Це створює величезні ризики, адже нападник має час не лише зібрати дані, а й поступово рухатися мережею, підвищувати привілеї, готувати бекдори. Компанія, яка думає, що «немає слідів атаки», насправді може жити в умовах прихованої компрометації.

Третім потужним інструментом атак є використання уразливостей у програмному забезпеченні та мережевих сервісах. Кількість нових CVE зростає щороку, і багато організацій просто не встигають їх закривати. «Patch gap» стає вікном можливостей для атакувальників. Відомі інциденти, як-от атаки на Microsoft Exchange через ProxyShell, або масштабна компрометація SolarWinds, показали, що одна єдина вразливість може відкрити доступ до десятків тисяч організацій.

Зловмисники часто діють у рамках концепції «zero-day», коли вразливість ще невідома спільноті, або ж використовують відомі «one-day» — ті, що вже описані, але не виправлені в конкретних компаніях. Автоматизовані сканери дають можливість знаходити такі цілі в лічені години. А оскільки уразливості можуть стосуватися систем управління віддаленим доступом (VPN, RDP), це одразу відкриває прямий канал до внутрішніх ресурсів компанії.

Однією з найбільш руйнівних загроз є ransomware. Його сучасна модель базується не лише на шифруванні, а й на викраденні даних. Це дає зловмисникам подвійний або навіть потрійний інструмент тиску: по-перше, вони блокують доступ до інформації, по-друге, погрожують її оприлюднити, а по-третє, тиснуть на партнерів і клієнтів компанії.

Рис 1.8 Успішні реалізації програм-вимагачів

Прикладом є атака на Colonial Pipeline у 2021 році, коли компанія була змушена зупинити роботу на кілька днів, що спричинило дефіцит пального у США. Інші гучні випадки пов’язані з групами REvil, Conti, LockBit, які публікували сотні гігабайтів викрадених даних на власних «сайтах-витоках». Сьогоднішні вимагачі працюють як повноцінні бізнес-структури, які мають PR-відділи, переговорників, інтерфейси для «клієнтів» і навіть служби підтримки. Це підтверджує, що ринок даних став основою кримінальної економіки.

Нарешті, окремо слід виділити інсайдерів. Вони мають законний доступ до систем і можуть використовувати його з особистих чи фінансових мотивів. Інсайдери є небезпечними саме тому, що діють у рамках легітимних прав і часто залишаються непоміченими. Вони можуть копіювати файли, передавати інформацію конкурентам або продавати її на чорному ринку. За різними оцінками, близько 20% витоків даних у світі мають інсайдерське походження [8].

Для протидії інсайдерам використовуються політики найменших привілеїв, контроль доступу до чутливих ресурсів, системи DLP (Data Loss Prevention) та моніторинг дій користувачів. Проте навіть найкращі технічні засоби не зможуть повністю виключити людський фактор. Саме тому інсайдерські загрози вважаються одними з найскладніших для запобігання.

Усі перелічені загрози — фішинг, валідні акаунти, уразливості, ransomware, інсайдери — різні за методами, але однакові за суттю. Їх кінцева мета — конфіденційні дані. Сучасні атаки зазвичай комбіновані: починаються з соціальної інженерії, переходять у використання облікових даних, доповнюються експлуатацією вразливостей і завершуються викраденням та шифруванням файлів. Це створює новий рівень викликів для кіберзахисту, адже жодна окрема технологія не може дати повний захист.

Рис 1.9 Приклад веб-ресурсу вимагача

Відповіддю на ці виклики є комплексний підхід: поєднання технологій, процесів і навчання користувачів. Стандарти безпеки, такі як NIST SP 800-53 чи CIS Controls, рекомендують багаторівневу модель захисту, яка включає автентифікацію, сегментацію мереж, патч-менеджмент, контроль привілеїв та моніторинг [9]. Лише завдяки інтеграції цих підходів можна знизити ризики втрати даних.

1.3. Аналіз існуючих рішень контролю доступу та моніторингу цілісності даних

Контроль доступу та моніторинг цілісності даних є одними з найважливіших складових системи кіберзахисту сучасних організацій. У цифрову епоху дані стали основним активом, який визначає конкурентоспроможність бізнесу, ефективність державного управління та безпеку суспільства загалом. Якщо у минулому інформаційні системи будувалися переважно навколо забезпечення конфіденційності через шифрування або блокування зовнішнього доступу, то сьогодні ключовим стає питання того, хто і на яких умовах може взаємодіяти з даними, і як швидко можна виявити несанкціоновану зміну чи втрату інформації. Таким чином, аналіз існуючих рішень у сфері контролю доступу та моніторингу цілісності дає змогу зрозуміти сильні та слабкі сторони сучасної архітектури кіберзахисту.

Історично перші моделі контролю доступу виникли у 1970-х роках, коли з’явилися багатокористувацькі операційні системи. Найпростішою з них була дискреційна модель (DAC), яка дозволяла власнику файлу чи ресурсу визначати, хто має право на читання, запис чи виконання. Це був зручний механізм для невеликих систем, але він швидко показав свої обмеження: у великих організаціях, де працюють сотні користувачів, управління індивідуальними дозволами ставало некерованим, а передача прав могла створювати нові вразливості. Для військових і урядових структур було розроблено обов’язковий контроль доступу (MAC), що ґрунтувався на рівнях секретності. У цій моделі користувачі отримували доступ

лише до тих ресурсів, які відповідали їхньому рівню, а будь-яка взаємодія перевірялася на основі суворих правил. MAC забезпечував високий рівень захисту, проте залишався надто жорстким для динамічного середовища бізнесу, де потрібна була оперативність.

Розвиток корпоративних IT у 1990-х роках привів до створення моделі контролю доступу на основі ролей (RBAC). Вона стала стандартом у великих компаніях і державних організаціях, адже дозволила групувати права доступу за функціональними ролями. Це значно спростило адміністрування: тепер права надавалися не окремим користувачам, а ролям — наприклад, «бухгалтер», «системний адміністратор», «аналітик». Однак навіть RBAC виявилася недостатньо гнучкою, адже вона не враховувала контекст: користувач мав однакові права незалежно від того, чи входив він у систему з корпоративного комп’ютера в офісі чи з особистого пристрою у громадському Wi-Fi.

Сучасний етап розвитку представлений атрибутивним контролем доступу (ABAC) і концепцією Zero Trust. ABAC враховує не лише роль, а й атрибути користувача, пристрою, середовища та дії. Це дозволяє будувати дуже складні політики, де доступ залежить від багатьох параметрів. Наприклад, лікар може мати доступ до медичних карток лише зі службового ноутбука у межах лікарні та у робочий час. Zero Trust іде ще далі: він виходить з того, що мережевий периметр більше не існує, і навіть внутрішнім користувачам не можна довіряти автоматично. Кожен запит на доступ повинен перевірятися, а ідентичність підтверджуватися повторно. Ця концепція, закріплена у стандарті NIST SP 800-207 [11], сьогодні активно впроваджується у великих компаніях та державних агенціях.

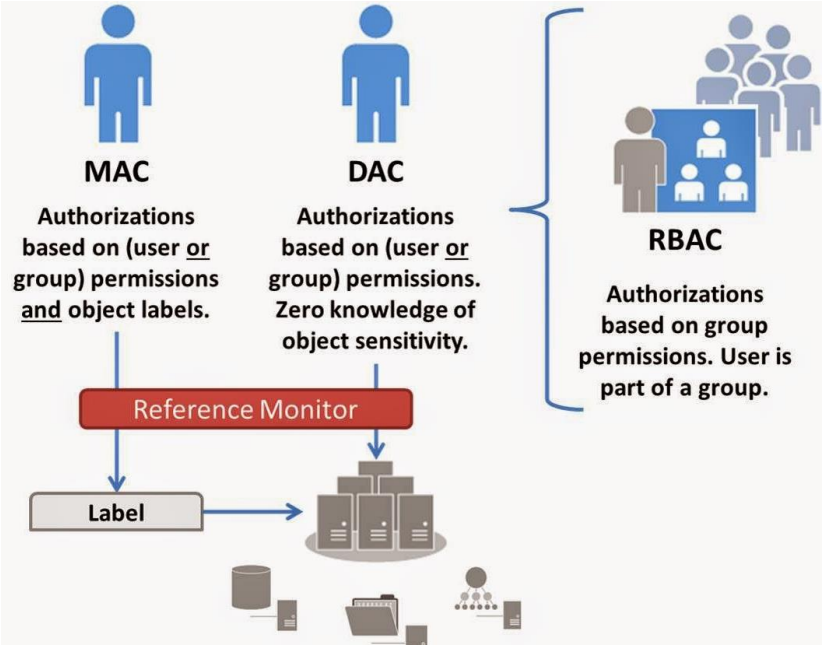

Рис 1.10 Схема моделей доступу

У практичному вимірі всі ці концепції реалізуються через системи управління ідентичністю та доступом (IAM). Вони забезпечують централізоване створення, управління та моніторинг облікових записів користувачів у масштабах організації. Сучасні IAM-платформи, такі як Okta чи Microsoft Entra ID, дозволяють інтегрувати локальні й хмарні сервіси, застосовувати багатофакторну автентифікацію (MFA), впроваджувати політики умовного доступу та федерацію ідентичностей. Особливе значення мають рішення PAM (Privileged Access Management), що фокусуються на захисті привілейованих облікових записів. Оскільки саме адміністратори і технічні користувачі мають найбільші права, контроль за їхніми діями є критично важливим. CyberArk чи BeyondTrust дозволяють записувати адмінські сесії, автоматично змінювати паролі після використання, блокувати підозрілі дії навіть під час активного сеансу.

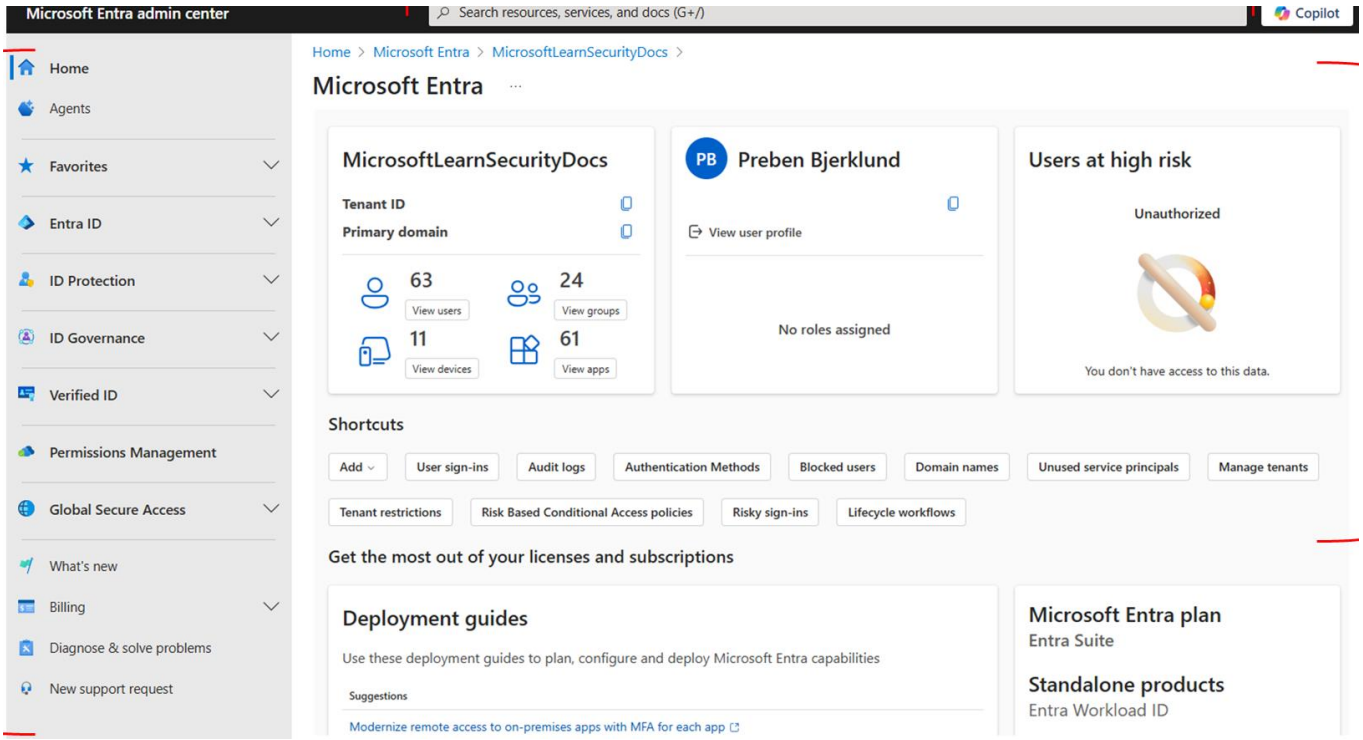

Рис 1.11 IAM-панель Microsoft Entra ID

Однак контроль доступу не вичерпує проблему. Якщо користувач (або зловмисник, що отримав його облікові дані) все ж таки отримав право доступу, потрібно мати механізми, які відстежують подальші дії. Тут на перший план виходить моніторинг цілісності даних. Цілісність означає, що дані залишаються у незмінному стані, якщо їхня модифікація не була санкціонована. Будь-які неочікувані зміни можуть свідчити як про атаку, так і про помилки чи саботаж. У великих організаціях просте логування не вирішує проблеми: потрібні системи, які автоматично перевіряють стан файлів і конфігурацій, створюють еталонні хеші та реагують на будь-яке відхилення.

Найпоширенішим підходом є File Integrity Monitoring (FIM). Цей метод застосовується для контролю критичних файлів, конфігураційних документів, реєстру Windows, баз даних. FIM-система створює контрольні суми та відстежує всі подальші зміни. Якщо файл змінено, додано чи видалено, генерується тривога. Це дозволяє виявити встановлення бекдорів, несанкціоновані модифікації системних бібліотек, маніпуляції з конфігураціями безпеки. Перші такі системи з’явилися ще у 1990-х, серед них особливе місце посідає Tripwire, що стала

стандартом у багатьох фінансових і промислових компаніях [13]. Сьогодні існують десятки рішень — від комерційних (Qualys, McAfee, Symantec) до відкритих (OSSEC, Wazuh).

Моніторинг цілісності часто інтегрується у системи HIDS (Host-based Intrusion Detection Systems), які відстежують не лише зміни файлів, а й логін-сесії, процеси, мережеві з'єднання. Вони дозволяють створювати більш повну картину подій на хості. У сучасних умовах FIM є невід'ємною частиною платформ SIEM (Security Information and Event Management), де дані про зміни файлів поєднуються з інформацією про автентифікацію, мережевий трафік, вразливості. Це дозволяє виявляти складні сценарії атак, де, наприклад, компрометація акаунту супроводжується зміною критичного файлу і підозрілими мережевими підключеннями.

Варто також відзначити, що регуляторні стандарти прямо вимагають наявності контролю доступу і моніторингу цілісності. Так, ISO/IEC 27001 наголошує на необхідності контролю прав доступу до інформаційних активів, а CIS Controls вказують на важливість постійного моніторингу цілісності критичних системних файлів [12]. У документі NIST SP 800-207 підкреслюється, що концепція Zero Trust неможлива без ефективного IAM і FIM, адже лише так можна забезпечити повний контроль над користувачами і даними у гібридних та хмарних середовищах [11].

Рис 1.12 Zero Trust з NIST

Разом з тим існуючі рішення мають і свої слабкі сторони. IAM-платформи є складними в адмініструванні і потребують ретельного налаштування. Їхнє впровадження часто супроводжується опором з боку користувачів, яким здається, що багатофакторна автентифікація чи постійні перевірки ускладнюють роботу. Системи FIM генерують значну кількість хибнопозитивних тривог: звичайні оновлення чи зміни конфігурацій можуть викликати сповіщення, що перевантажує аналітиків безпеки. Додатковим викликом є вартість: комерційні платформи IAM і FIM потребують значних інвестицій, тому малий та середній бізнес часто змушений використовувати спрощені або безкоштовні рішення.

Проте перспективи розвитку цієї сфери є надзвичайно динамічними. По-перше, зростає роль штучного інтелекту і машинного навчання, які дозволяють зменшувати кількість хибних спрацювань і краще аналізувати поведінку користувачів. По-друге, активно впроваджується автоматизація реакції: якщо FIM-система виявляє підозрілу зміну, вона може одразу ізолювати вузол, відкотити файл до попередньої версії, заблокувати акаунт. По-третє, дедалі більшої ваги набувають хмарні IAM-сервіси, які дозволяють компаніям не витрачати ресурси на локальне адміністрування.

Таким чином, аналіз існуючих рішень доводить, що контроль доступу та моніторинг цілісності є взаємодоповнюючими стовпами сучасної кібербезпеки. Без ефективного контролю доступу навіть найкраща система шифрування не гарантує захисту даних, а без моніторингу цілісності навіть найкращі політики доступу не дають впевненості у збереженні інформації. Разом вони утворюють основу безпеки, яка дозволяє організаціям протистояти складним і багаторівневим загрозам.

2 АНАЛІЗ МОЖЛИВОСТЕЙ ЗАБЕЗПЕЧЕННЯ ЗАХИСТУ КІНЦЕВИХ ТОЧОК НА ОСНОВІ SIEM WAZUH

2.1. Основні компоненти архітектури та функціональні можливості Wazuh

Сучасні системи виявлення та реагування на інциденти безпеки еволюціонували від «збирачів логів» до комплексних платформ, здатних охоплювати увесь життєвий цикл події: від спостереження за кінцевими точками та хмарними сервісами до кореляції, аналітики, візуалізації й автоматизованих дій у відповідь. Wazuh — репрезентативний представник цього класу: це відкрита SIEM/XDR-платформа з модульною архітектурою, де збір і попередній аналіз виконують агенти на кінцевих точках, центральний сервер займається кореляцією та керуванням, а шар індексації та візуалізації забезпечує масштабоване збереження й дослідження подій у часі [15]. На високому рівні архітектура Wazuh складається з багатоплатформенного агента, центральних компонентів Wazuh server (менеджер), Wazuh indexer (реальний пошук/аналітика) і Wazuh dashboard (візуалізація, керування). Дані, які збирає агент, передаються на сервер для аналізу, далі — до кластеризованого індексера для збереження й пошуку, а потім — на дашборд для моніторингу та розслідувань [15][20]. Завдяки такому поділу обов’язків платформа масштабовано переварює великі обсяги телеметрії, забезпечуючи одночасно оперативність (онлайн-алерти) та історичність (ретроспектива за місяці/роки).

Рис 2.1 SIEM-система Wazuh

Ядром концепції Wazuh є агент — легкий процес, який працює на Windows, Linux чи macOS і безперервно спостерігає за системними подіями: автентифікаціями, запусками процесів, мережевими з'єднаннями, змінами у файловій системі, конфігураціями ОС, станом пакунків та оновлень, подіями служб, журналами застосунків тощо [18]. Модульність агента дозволяє увімкнути рівно ті можливості, які потрібні для конкретного профілю ризику, зберігаючи мінімальний вплив на продуктивність. Наприклад, Logcollector читає журнали з файлів, каналів або journald, працюючи багатопотоково для підтримання стабільної швидкості інжесту; службовий файл статистики дає змогу контролювати сталість збору логів і виявляти «вузькі місця» у конвеєрі [18]. Для інвентаризації середовища використовується Syscollector: він збирає апаратні характеристики, версії ОС, список ПЗ, мережеві інтерфейси, відкриті порти, процеси, оновлення Windows — це дозволяє будувати asset context і корелювати події з реальним станом вузла [15].

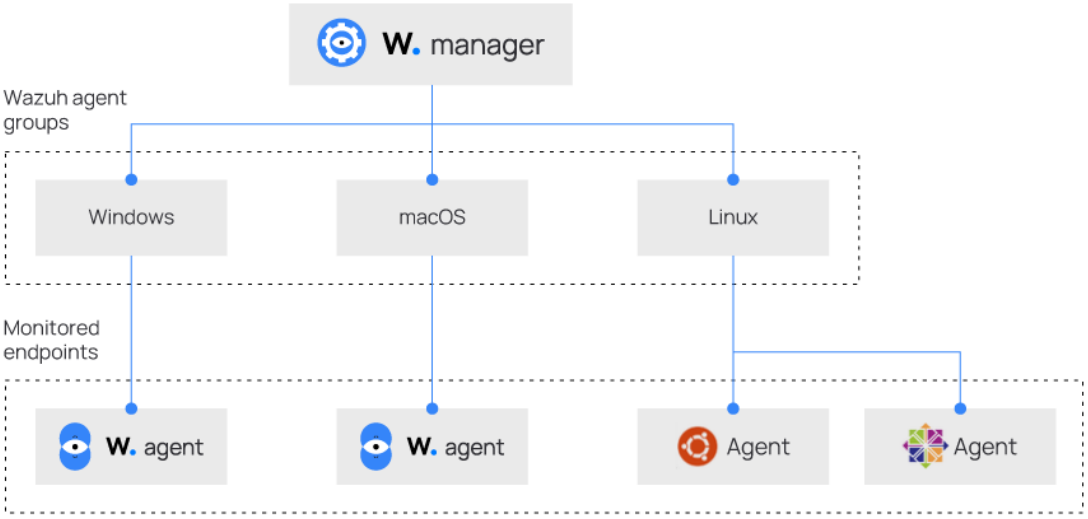

Рис 2.2 Типи агентів Wazuh

Зібрані агентом події потрапляють на Wazuh server (менеджер), де відбувається декодування і кореляція. Механізми decoders і rules є серцем аналітики: вони перетворюють «сирі» рядки логів різних виробників на уніфіковані поля й семантику безпеки, придатні для правил, дашбордів і звітності. Для складних форматів логів передбачено custom decoders і утиліту wazuh-logtest (через

консоль, API або з дашборду), що дозволяє відлагоджувати власні декодери/правила на зразках логів ще до «бойового» застосування [19]. На цьому етапі формується алерт з пріоритетом, тегами відповідності (PCI/HIPAA/NIST/GDPR), контекстом і посиланнями на вихідні події; далі алерти індексуються у Wazuh indexer для швидкого пошуку та аналітики [15][20].

Wazuh indexer — це повнотекстовий пошук і аналітичний рушій, який масштабується горизонтально, підтримує кластерні розгортання (реплікації, розподіл шард), індекс-лайфсайкл менеджмент та API для керування індексами/запитами. У виробничих середовищах індексер розгортають кластером для високої доступності та продуктивності; практики побудови кластерів (на базі OpenSearch-стеку) рекомендують виділяти окремі керівні вузли для кворуму та розносити ролі data/ingest/coordination задля балансу [20]. Це важливо для довгострокового зберігання великих масивів подій і побудови складних дашбордів без деградації швидкодії.

Над шаром індексації працює Wazuh dashboard. Це веб-інтерфейс, який одночасно і керує середовищем (підключення агентів, вибір модулів, RBAC для користувачів/ролей), і є «вітриною» безпеки: інтерактивні графіки, таймлайни, карти, списки інцидентів, готові модулі для threat hunting, PCI DSS, HIPAA, FIM, Vulnerability Detection, SCA, хмарного моніторингу тощо [15]. Комунікація з менеджером іде через RESTful API (TLS, логін/пароль або токен), що дозволяє уніфікувати керування, автоматизувати операції та обмежувати права через політики доступу [15].

Однією з найсильніших сторін Wazuh є рідні модулі безпеки. File Integrity Monitoring (FIM) створює базові «зліпки» (хеші, атрибути) для критичних шляхів і відстежує створення/модифікацію/видалення файлів у реальному часі; будь-який розбіг з базовою лінією одразу породжує сповіщення. Це класичний спосіб виявлення персистентності, несанкціонованих конфіг-змін і порушень політик, що водночас допомагає закривати вимоги комплаєнсу (PCI DSS, HIPAA) [16][17]. Rootcheck (виявлення руткітів) періодично сканує вузол, поєднуючи сигнатурні та поведінкові методи, аби знаходити приховані процеси, модифіковані бібліотеки,

аномальні системні артефакти [15]. Vulnerability Detection взаємодіє з CTI-репозиторіями, порівнює інвентар ПЗ із відомими CVE й видає зведення по хостах/додатках, що потребують патчів [20]. Security Configuration Assessment (SCA) перевіряє відповідність «харденінг»-політикам і формує агреговану оцінку стану вузла з рекомендаціями, підключаючись політиками у конфігурації агента [15]. Завершує набір Active Response — механізм автоматизованих дій у відповідь: від блокування IP на файрволі/iptables до завершення процесів або виклику зовнішніх скриптів; у комбінації з правилами це дозволяє «зрізати» частину інцидентів до втручання аналітика і підтримує операційні вимоги комплаєнсу [17].

Важлива практична особливість Wazuh — гнучкість розгортання. Для малих середовищ достатньо одинарного сервера з індексером; для середніх і великих — кластери Wazuh server (master/worker) за балансувальником і кластери Wazuh indexer (шардинг + реплікація), що забезпечують горизонтальне масштабування та відмовостійкість [20]. Якщо потрібен керований сервіс, доступний Wazuh Cloud, де вся інфраструктура платформи надається як сервіс: інсталяція, оновлення, масштабування та доступ до архівних даних через API [15].

Окремо важливі інтеграції з AWS CloudTrail, Microsoft Azure та Google Cloud: модулі збору дозволяють отримувати події керування (API-виклики, автентифікації, зміни налаштувань безпеки) і сервісні журнали провайдерів. Це формує єдину «стрічку подій безпеки» для гібридних інфраструктур і дає змогу автоматично реагувати на ризикові зміни (наприклад, нетипові модифікації ролей/груп безпеки) [15][20].

З погляду відповідності, Wazuh має попередньо налаштовані правила/декодери, дашборди та мапінги на регуляторні вимоги PCI DSS, HIPAA, NIST 800-53, GDPR: алерти отримують теги контролів, а тематичні панелі дають огляд, де і коли порушено вимогу та які агенти генерують відхилення. Це суттєво спрощує аудит і регулярну звітність [17][18][20].

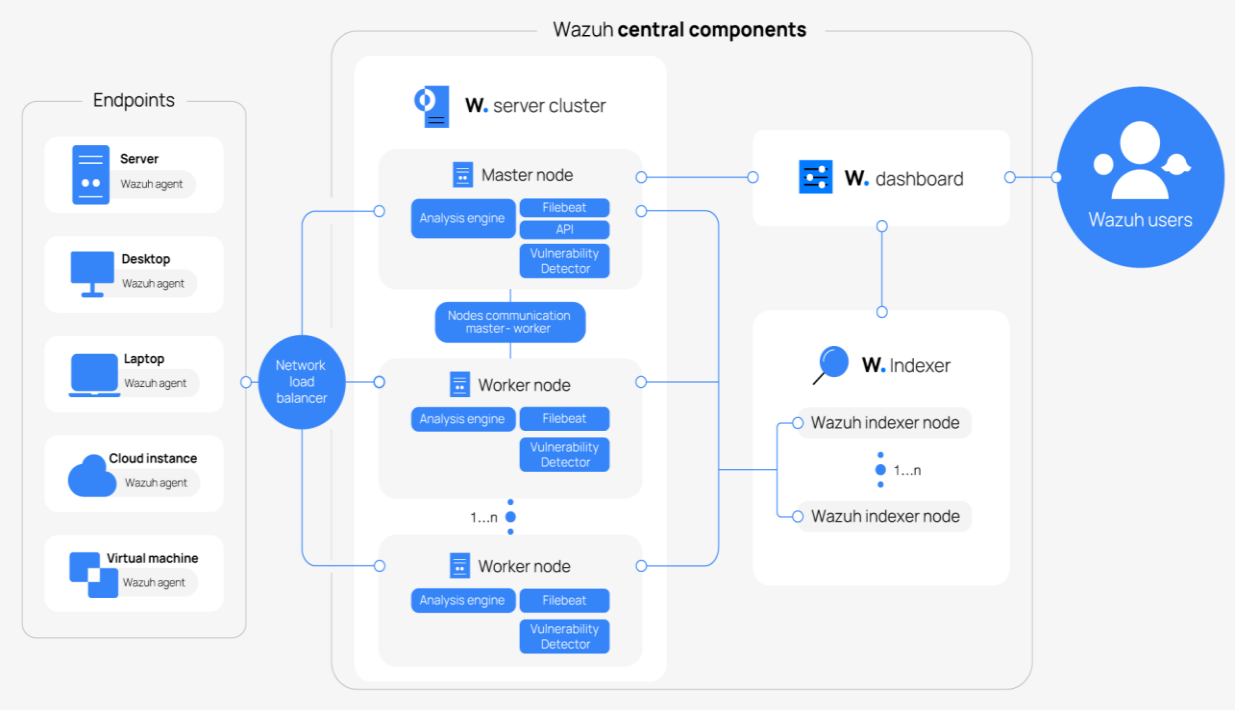

Рис 2.3 Архітектура Wazuh

З інженерної точки зору сильна сторона Wazuh — відкритість і розширюваність. Власні декодери/правила додаються декларативно, тестуються логтестом, прив'язуються до груп агентів; через REST API і RBAC усе це автоматизується від розгортання до оновлень. Для нетипових джерел логів можливий інжест через syslog або API, а за потреби — форвардинг у зовнішні сховища/платформи в межах того ж стеку індексації та візуалізації [19][20].

Таблиця 2.1.

Модулі Wazuh

Модуль	Функція безпеки	Типові події	Результат виявлення
Logcollector	Збір журналів ОС/застосунків із підозрілими scope	Auth, process, network, service logs	Єдина точка інжесту; живий контекст вузла
Syscollector	Інвентаризація ПЗ/апаратури/мережі	Packages, processes, ports, NICs, updates	Контекст активів; підтримка Vulnerability

Продовження таблиці 2.1.

Модулі Wazuh

Decoders/ Rules	Нормалізація та кореляція	Розмічені поля подій, зіставлення патернів	Змістовні алерти, теги комплаєнсу
Syscheck	Контроль цілісності	Hash/ACL/attrs змін файлів/реєстру	Раннє виявлення НСД/персистентності; відповідність PCI/HIPAA
Rootcheck	Виявлення руткітів/аномалій	Приховані процеси, модифіковані бібліотеки	Виявлення прихованих загроз ядра/юзерспейсу
Vulnerability detection	Ідентифікація CVE	Інвентар ПЗ + довідники вразливостей	Пріоритизація патчів; регулярний «пульс» ризиків
SCA	Оцінка конфіг-відповідності	Політики hardening, тести конфігурацій	«Shift-left» комплаєнсу; керовані рекомендації
Active Response	Автоматизоване реагування	Скрипти block/kill/quarantine	Скорочення MTTR; виконання операційних вимог комплаєнс
Indexer	Пошук/аналітика/збереження	Індекси, шардинг/репліка, ILM	Історичні розслідування, високий SLA
Dashboard +API+RBAC	Огляд/керування/автоматизація	Модулі PCI/HIPAA/FIM/SCA; REST; ролі	Керованість, розмежування доступу, інтеграції

Практична цінність Wazuh розкривається у «end-to-end» сценаріях. Наприклад, компрометація хосту через фішинг починається з аномальної події входу в незвичний час/географію (Logcollector), продовжується запуском нетипових процесів і зміною автозапуску (Syscollector + FIM), встановленням

бекдору (Rootcheck), а завершується появою вразливих бінарів (Vulnerability Detection). Правильно побудовані правила складають ці кроки в єдиний інцидент, Active Response блокує джерело, а комплаєнс-дашборд показує, які вимоги порушені та які політики SCA слід підсилити [15][16][17][20].

Не менш важливо, що Wazuh не обмежується локальним периметром: інтеграції з AWS, Azure і GCP додають до картини події керування і доступів у хмарі — створення ключів доступу, зміну ролей/політик, модифікацію мережевих правил. Це критично для сучасних Zero Trust-підходів, де кожен запит має бути «обґрунтований» контекстом пристрою, користувача, часу та ефекту [15][20]. Підсумовуючи, Wazuh поєднує риси класичного SIEM (універсальний збір, нормалізація, кореляція, збереження, візуалізація) та XDR (глибока телеметрія з хоста, FIM/Rootcheck/SCA/Vuln, Active Response). Відкритість коду і модульність знижують бар’єр входу й дають простір для гнучкої адаптації під доменну специфіку. Масштабована архітектура, зрозумілі механізми RBAC і API, готові комплаєнс-модулі та хмарні інтеграції формують платформу, здатну виконувати роль «нервової системи» кіберзахисту від SMB до ентерпрайзу [15][20].

2.2. Інтеграція джерел логів та налаштування модуля моніторингу цілісності файлів (FIM)

Успіх будь-якої SIEM/XDR-побудови починається не з красивих дашбордів, а з дисципліни інтеграції джерел даних і якості телеметрії. Якщо дані неповні, несинхронізовані за часом, не нормалізовані або надмірно «шумні», аналітика неминуче втрачає свою точність, зростає кількість хибних спрацювань, а справжні інциденти губляться у фоні. Саме тому інтеграція джерел логів і налаштування модуля моніторингу цілісності файлів (FIM) у Wazuh стають критично важливим етапом, що визначає точність виявлення загроз, швидкість реагування та можливість відповідати вимогам регуляторики.

Побудова інтеграції завжди починається з визначення бізнес-критичних процесів та активів, а також сценаріїв загроз, які найбільш актуальні саме для них.

Виходячи з цього визначається, які саме події потрібно відстежувати та які джерела логів необхідно інтегрувати. У Wazuh це відображається у плані розгортання агентів на кінцевих точках, увімкненні необхідних підмодулів, виборі способу збору (агентський чи агентless/syslog), а також створенні декодерів та правил для спеціалізованих форматів даних.

Найбільш очевидним і базовим джерелом є кінцеві точки та сервери, що працюють під керуванням Windows, Linux чи macOS. На них Wazuh-агент здійснює збір системних журналів, подій автентифікацій, запусків процесів, мережевих підключень, зміни конфігурацій системи та застосунків. Для Windows пріоритетними є канали Security, System, Application, PowerShell, а також Sysmon, якщо він увімкнений. Вони дають детальну картину активності користувачів і процесів, що критично для виявлення атак, які обходять стандартні журнали. У Linux першочерговими стають журнали автентифікацій, SSH, sudo, журнали веб-серверів і проксі. Усі ці дані потрапляють на сервер Wazuh, де проходять через декодери та правила, нормалізуються та збагачуються контекстом.

Не менш важливими є мережеві пристрої: маршрутизатори, балансувальники, міжмережеві екрани, IPS/IDS. Тут частіше використовується агентless-збір через Syslog. Події, які генеруються цими пристроями, надходять до Wazuh за допомогою колектора, що дозволяє контролювати роботу всієї мережевої інфраструктури. На цьому етапі критично важливо враховувати формати логів, часові пояси та коректність структурованих полів, щоб уникнути помилок у подальшій аналітиці.

Велику цінність становлять журнали веб-серверів, проксі, баз даних і корпоративних застосунків. Саме тут відображаються спроби SQL-ін’єкцій, підбору паролів, обходу WAF чи втручання у бізнес-логіку. Якщо програма використовує власний нестандартний формат логів, у Wazuh можна створити кастомні декодери. Це дає змогу інтегрувати будь-яке джерело, навіть внутрішньоорганізаційні застосунки, у єдину систему спостереження [19].

З розвитком хмарних сервісів важливо інтегрувати журнали керування з AWS, Azure та Google Cloud. У Wazuh для цього є готові модулі, які збирають події

CloudTrail, Azure Activity чи Google Cloud Audit. Ці події дозволяють відстежувати зміну ролей, створення ключів доступу, модифікацію правил безпеки чи розгортання нових екземплярів. Таким чином, організація отримує єдину панель управління, що об’єднує локальні сервери та хмарну інфраструктуру.

Не менш важливими є події автентифікації та доступу з систем ідентифікації — Active Directory, Azure AD, Okta, VPN-шлюзів чи брокерів VDI. Саме вони дозволяють побачити спроби перебору паролів, несанкціоновані входи, збої багатофакторної автентифікації, використання викрадених токенів. Усі ці дані інтегруються у Wazuh та стають основою для сценаріїв кореляції.

Ключовим аспектом успішної інтеграції є забезпечення якості даних. Тут обов’язковими є синхронізація часу через NTP, шифрування каналів TLS між агентами та сервером, нормалізація часових поясів і стандартизація назв полів. Якщо ці вимоги не дотримуються, навіть найсильніші аналітичні правила будуть генерувати хаос.

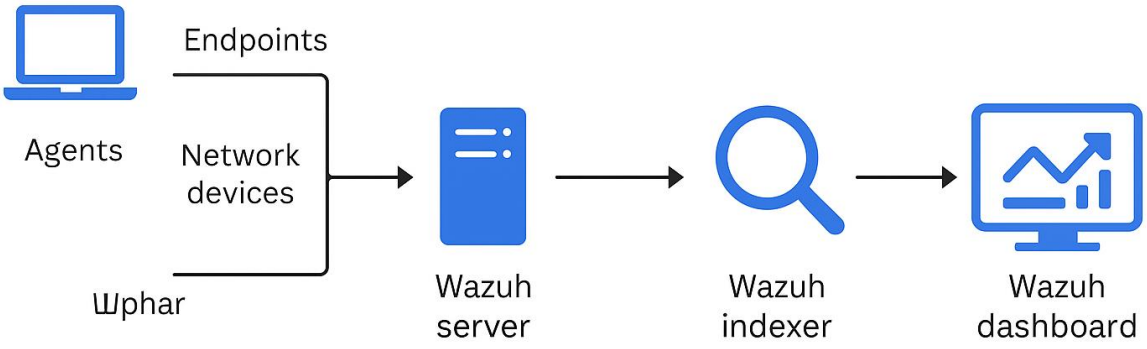

Рис 2.4 Загальна схема інтеграції джерел логів в Wazuh

На наступному рівні йде моніторинг цілісності файлів (FIM). Це модуль Wazuh, який забезпечує контроль критичних системних файлів, конфігурацій і реєстру Windows. Його завдання — відслідковувати всі зміни та вчасно сповіщати про них. FIM дозволяє створити базові «зліпки» файлів і перевіряти їхню цілісність за допомогою хеш-функцій. Якщо файл змінено, видалено або додано, система

генерує алерт. Це надзвичайно ефективний механізм проти атак, що намагаються закріпитися у системі через модифікацію конфігурацій чи додавання бекдорів [16].

FIM може працювати у двох режимах: періодичне сканування та режим «whodata», який на Linux використовує механізми inotify чи auditd, а на Windows — USN Journal. Саме режим whodata дозволяє отримати повний контекст зміни: хто, коли і яким процесом модифікував файл. Це значно спрощує розслідування та дозволяє відрізняти легальні дії від атак. Однак whodata потребує більшої обчислювальної потужності, тому зазвичай його вмикають лише для критичних директорій і конфігурацій. Для великих каталогів застосовується періодичне сканування з контрольними сумами [18].

```
{
  "rule": { "id": 5542, "level": 10, "description": "FIM: Critical system file modified (whodata)" },
  "agent": { "id": "015", "name": "srv-app-01", "ip": "10.10.4.27", "os": "Ubuntu 22.04" },
  "fim": {
    "path": "/etc/ssh/sshd_config",
    "event": "modified",
    "whodata": {
      "user": "root",
      "uid": "0",
      "process_name": "/usr/bin/vim",
      "inode": "131229",
      "ppid": 2677,
      "audit_uid": 0,
      "login_user": "root",
      "tty": "pts/2"
    },
    "hash": {
      "old": "6a8c5b0a2f8a2d66f51b3d2f8a6b0ec5e1a0f4d2b6b7f9f2c1e3b5a9d7c0e123",
      "new": "8b9f2e1a6c0d4f2e3b5a9d7c0e1236a8c5b0a2f8a2d66f51b3d2f8a6b0ec5e1"
    },
    "perm": { "old": "rw-r--r--", "new": "rw-r--r--" },
```


Рис 2.5 Повний контекст FIM-події зі зміною хешу та користувачем

Важливо пам’ятати про продуктивність. Якщо ввімкнути FIM на всі файли системи, це створить лавину подій, яку буде неможливо аналізувати. Тому рекомендується визначити «золотий список» директорій: /etc, /var/www, /usr/local/bin у Linux, а також системні каталоги Windows, ключі автозапуску, служби та політики безпеки. Решту директорій можна додати у винятки або відстежувати періодично.

Налаштування FIM має враховувати і винятки. Тимчасові файли, журнали або кеш можуть змінюватися щосекунди, і відстеження кожної з цих змін створить шум. Винятки допомагають залишити лише важливі події. Водночас робити

винятки слід обережно: саме в тимчасових каталогах іноді залишаються копії викрадених даних.

FIM також дозволяє збирати додаткові атрибути — права доступу, власників, SELinux-мітки. Це важливо для відповідності вимогам стандартів, таких як PCI DSS чи HIPAA. Окремо налаштовується контроль реєстру Windows, включаючи ключі автозапуску, служби, LSA Providers, політики безпеки. Це допомагає виявляти спроби створення бекдорів і персистентних загроз [16].

```
<ossec_config>
  <syscheck>
    <directories whodata="yes">/etc/ssh</directories>
    <directories whodata="yes" report_changes="yes">/etc</directories>
    <directories realtime="no" check_all="yes">/usr/local/bin</directories>
    <registry>HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services</registry>
    <registry>HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run</registry>
    <ignore>/var/log</ignore>
    <ignore>/tmp</ignore>
    <nodiff>/etc/ssl/private</nodiff>
    <frequency>3600</frequency>
    <scan_on_start>yes</scan_on_start>
    <auto_ignore frequency="120">no</auto_ignore>
    <alert_new_files>yes</alert_new_files>
  </syscheck>
</ossec_config>
```

Рис 2.6 Налаштування моніторингу цілісності у Wazuh

Особливу роль відіграє зв’язка FIM і Active Response. Наприклад, якщо виявлено зміну у файлі конфігурації SSH, система може автоматично повернути «золотий» варіант або заблокувати обліковий запис, який ініціював зміну. Це дозволяє скоротити час реагування з годин до секунд і запобігти розвитку атаки [18].

```
2025-10-06T14:22:32Z wazuh-agentd[srv-app-01]: [ar] Action triggered by rule_id=5542 (FIM critical change)
 - command: firewall-drop
 - srcip: 185.71.67.29
 - user: root
 - file: /etc/ssh/sshd_config
 - reason: unauthorized modification detected (whodata)
 -> result: SUCCESS (iptables -I INPUT -s 185.71.67.29 -j DROP)

2025-10-06T14:22:33Z wazuh-agentd[srv-app-01]: [ar] Action triggered by rule_id=5542
 - command: integrity-restore
 - file: /etc/ssh/sshd_config
 - baseline: sha256 6a8c5b...e123
 -> result: SUCCESS (file restored from golden copy)

2025-10-06T14:22:35Z wazuh-agentd[srv-app-01]: [ar] Action triggered by rule_id=5542
 - command: notify-soc
 - channel: webhook https://soc.local/hooks/fim
 -> result: DELIVERED (200 OK)
```

Рис 2.7Автоматизована дія Active Response у відповідь на FIM-подію

FIM безпосередньо підтримує вимоги багатьох стандартів. PCI DSS вимагає контролю системних компонентів, HIPAA — аудит змін, що впливають на ePHI, а NIST 800-53 — перевірки цілісності. У Wazuh є готові дашборди для цих стандартів, що автоматично позначають FIM-події відповідними контролями, значно спрощуючи аудит [16].

Тестування FIM перед впровадженням у продакшн включає перевірку часу реакції, навантаження на диски, рівня хибних спрацювань, повноти контексту та коректності позначок комплаєнсу. Успішне тестування гарантує, що система генерує лише релевантні алерти та підтримує баланс між продуктивністю і точністю.

У підсумку, інтеграція джерел логів у Wazuh і правильне налаштування FIM формують основу для точного виявлення загроз, швидкого реагування та відповідності регуляторним вимогам. Грамотна стратегія інтеграції, контроль часу, якість даних і ретельно налаштований моніторинг цілісності забезпечують ефективність всієї платформи.

2.3. Розробка та впровадження кореляційних правил та механізмів виявлення аномального доступу до кінцевих точок

Розробка та впровадження кореляційних правил і механізмів виявлення аномального доступу до кінцевих точок є одночасно інженерною задачею і задачою побудови знань про те, що в конкретній інфраструктурі вважати нормою, а що — відхиленням. Підхід має починатися з формалізації загроз і перетворення цих сценаріїв у набір атомарних логічних індикаторів, які можна надійно знайти в телеметрії. Типова атака на кінцеву точку розкладається на послідовність кроків: ініціальна компрометація (фішинг, drive-by, забруднений інсталятор), локальна ескалація привілеїв або встановлення персистентності (модифікація автозапусків, драйверів, служб), lateral movement (використання протоколів віддаленого доступу, передачі креденшалів), ексфільтрація і очищення слідів. Кожен з цих кроків породжує унікальні ідентифікатори в логах: невдалі/успішні автентифікації, запуск чи завершення процесів, модифікації файлів (FIM), зміну прав та власників, нетипові мережеві з'єднання, виклики API у хмарі. Технічною відправною точкою є чіткий перелік полів, які декодер має коректно витягувати з кожного джерела: timestamp, agent.id, host.name, src.ip, dst.ip, user.name, process.name, process.path, process.hash, pid, ppid, tty, event.type, file.path, file.hash.old/new, registry.key, registry.value, session.id, cloud.account, and severity. Формалізація полів — це необхідна умова для того, щоб однакові правила могли застосовуватися до гетерогенних джерел (Windows Event, Syslog, CloudTrail, Firewall logs) без необхідності дублювання логіки під кожен формат.

Розробка кореляційних правил починається з вибору моделі кореляції: сигнатурна кореляція (комбінації відомих подій), поведінкова кореляція (відхилення від профілю користувача/пристрою) та гібридні моделі, що поєднують обидва підходи. У практичній інженерії для Wazuh це означає створення кількох рівнів правил: низькорівневих «сигнатурних» правил, що ловлять одиночні високодостовірні індикатори (наприклад, виконання файлу з відомим зловмисним

хешем), середнього рівня правил, що агрегують повторювані події (наприклад, N невдалих автентифікацій з одного IP за T секунд), і кореляційних правил вищого рівня, що поєднують події різних типів у часових вікнах (наприклад, невдалий вхід → успішний вхід з тієї ж адреси → модифікація FIM → спроба підключення до зовнішнього сервера). Технічно в Wazuh такі правила формують XML-файли з параметрами `frequency`, `timeframe`, `same_source`, `group`, `aggregate` і `correlate`; ці параметри дозволяють задавати ключі кореляції (наприклад user.name або src.ip), часові вікна і агрегаційні пороги. При побудові правил критичною інженерною практикою є використання груп і тегів: кожне правило має бути прив'язане до тематичної групи (SSH, RDP, Cloud, Endpoint, FIM) і тендеризоване тегами комплаєнсу і ступеня ризику, що дозволяє аналітику відфільтровувати та систематизувати інциденти.

Коли йдеться про виявлення аномального доступу, корисна стратегія — комбінаторне накладання часових вікон: короткі (секунди–хвилини) для миттєвих сплесків, середні (десятки хвилин) для послідовних дій та довгі (години–дні) для повільних, таргетованих атак. Технічно це реалізується шляхом створення паралельних правил з різними `timeframe` і `frequency`. Наприклад, для виявлення brute-force атак доцільно мати правило з `frequency=5` і `timeframe=60s` для виявлення burst-атак та правилом з `frequency=50` і `timeframe=12h` для slow-and-low атак. Такий підхід дозволяє не пропустити ні швидку, ні затяжну кампанію. Проте при цьому необхідно враховувати накопичення контексту: для довгих часових вікон потрібна політика збереження кореляційних контекстів (correlation context TTL) у пам'яті менеджера або у тимчасових індексах у indexer; вибір TTL має балансувати між виявленням «повільних» ланцюжків і контролем використання пам'яті.

Важливим технічним аспектом є правильний вибір кореляційних ключів. Один лише src.ip часто буває ненадійним через динамічність адрес, NAT і проксі; тому слід комбінувати ключі: src.ip + user.name, src.ip + agent.id, process.hash + path, session.id. Комбінації допомагають будувати «корпус» подій, що належать до однієї кампанії. Наприклад, lateral movement часто проявляється як серія успішних

автентифікацій під різними обліковками, але з однаковим pattern процесів запуску (PSExec-like) або з однаковими цілями доступу до конкретного шару файлів; у такому випадку кореляція по agent.id і поцільових ресурсах (target.host, target.service) дає більш надійний індикатор. З технічної точки зору це означає, що правила повинні підтримувати множинний ключ кореляції і вміти об’єднувати події навіть при частковому збігу полів.

Ефективність кореляції значно зростає при застосуванні enrichment-процесів: asset tagging (точне відображення, що за хост — сервер бази даних, що — робоча станція), vulnerability scoring (CVSS або внутрішній score, підтягується з Vulnerability Detection), гео-локація IP, біли-листування відомих системного трафіку та threat-intel фідинг (IOC). Технічно enrichment виконується або на початковому етапі ingest (через decoders/transformers) або як окремий pipeline у indexer, де до документу додаються поля `asset.owner`, `asset.criticality`, `vuln.score`, `ioc_match`. Для інтеграції threat-intel важливо реалізувати кешування запитів, механізм TTL і пріоритезацію фідів, щоб запити не викликали перевантаження зовнішніх API при пікових навантаженнях. Практичний патерн — використовувати локальний proxy/feed-aggregator, який щогодини оновлює локальну базу IOC, а правила звертаються вже до внутрішнього кешу.

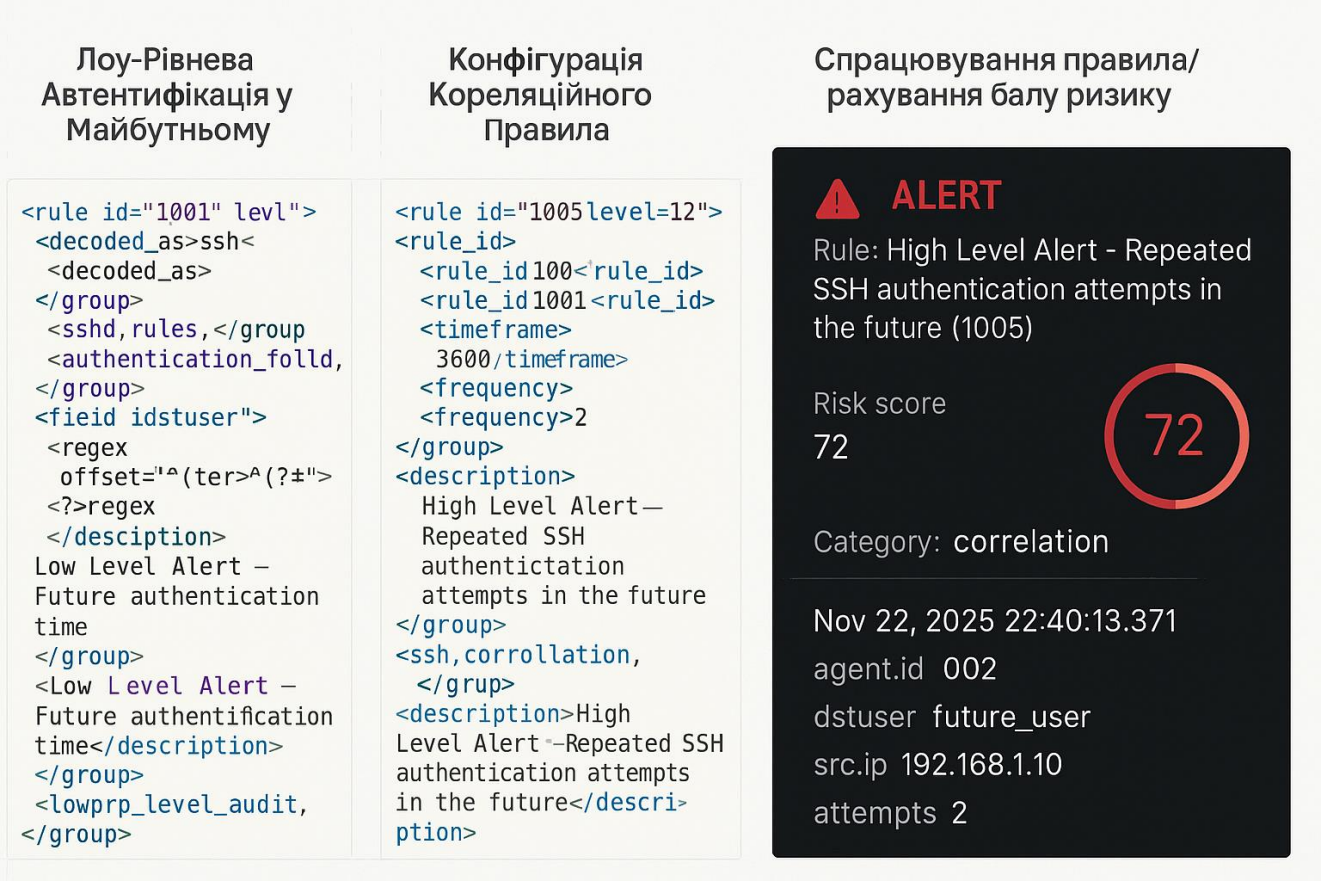

Рис 2.8 Кореляційне правило на аномальну автентифікацію

Поведінковий аналіз (UEBA) — окрема, але органічно інтегрована частина детекції. У базовому технічному вигляді UEBA реалізується як набір метрік, що описують нормальну поведінку сутностей за часовий проміжок: середній час активності, середня кількість сесій на добу, частота доступів до конкретних ресурсів, середня кількість файлів, що прочитуються, і так далі. Технічно це потребує збереження тимчасових серій у аналітичному шарі indexer з високим retention для усталених патернів (щонайменше кілька тижнів), а також механізмів обчислення агрегатів (rolling window aggregations) для швидкого порівняння. Простими прикладами UEBA-детекторів можуть бути правила з порівнянням поточного показника до історичного середнього із врахуванням стандартного відхилення (z-score), або детектори ентропії поведінки команд у shell. Для просунутих випадків доцільне використання зовнішніх ML-моделей, які на основі набору ознак (features) видають risk-score; з технічної точки зору такі моделі

живлять рішення threshold-based у Wazuh: якщо risk-score > X → створити medium/high alert і подати додатковий контекст.

Розробка правил вимагає ретельного тестування та валідації. На етапі розробки слід застосовувати декілька рівнів тестування: unit-тести на синтетичних логах (використовуючи `wazuh-logtest` для перевірки декодерів і тригерів правил), replay історичних логів (backfill) з підрахунком TPR/FPR для емпіричної оцінки якості, канареєчне розгортання (canary) у відокремленому сегменті для спостереження в умовах production, і, нарешті, rollout з incremental activation та моніторингом ключових метрик (alert rate, average analyst time per alert, MTTR). Технічно для автоматизації цього процесу рекомендується версіонування правил у Git, побудова CI pipeline, який запускає автоматичні тести (лог-тести і replay на історії) при кожному PR, і автоматичне деплоювання тільки при успішному проходженні тестів. Вміщення правил у CI/CD також дає можливість контролювати зміни, відстежувати авторів і дозволяти швидкий rollback у разі раптового збою.

Продуктивність і масштабування є вирішальними факторами при проектуванні кореляційного шару. Дорогі правила, що виконують складні regex-операції або довгі часові joins, слід переписувати у більш продуктивні конструкції: розбивати на проміжні агреговані події, виконувати попередню агрегацію на боці менеджера або indexer, або ж застосовувати stream-processing підхід, де важкі обчислення виконуються оффлайн батчами, а результати використовуються для кореляції легкими правилами у real-time. Практичний приклад: замість прямого підрахунку змін файлів і входів за 24 години у кожному новому правилі, зберігати щогодинні агреговані документи `failed_logins_summary` і `fim_changes_summary`, які містять counts і top-entities, а потім корелювати вже ці агреговані події. Така індексація значно зменшує latency і дозволяє виконувати складні аналітичні запити лише на aggregate-івентах.

Кореляція повинна тісно інтегруватися з FIM та Active Response. Наприклад, правило типу «Unauthorized Privilege Escalation» може поєднувати подію FIM про зміну `/etc/sudoers`, подію про створення облікового запису (useradd), і мережеву активність автономного з’єднання до зовнішнього сервера.

Якщо правило виявляє такий ланцюжок, воно має не лише створити high-severity інцидент, а й викликати серію дій через Active Response: зняти snapshot диска, ізолювати хост у VLAN, відкотити конфігурацію з golden-copy, і підняти ticket у ITSM. Технічно це означає, що правила повинні мати чітке поле `action` або `response` з посиланням на playbook, який виконує і робить idempotent операції з timeout і rollback можливістю. Інтеграція із SOAR-платформою (через API) дозволяє формалізувати і складніші сценарії з участю людей і автоматизації.

Важливий аспект — управління false positives та шумом. Для цього потрібні метрики і governance: для кожного правила визначається мета (що воно має виявити), очікувані показники (TP/week, acceptable FP rate), період тестування і власник правила. Після раннього запуску потрібно вести щотижневий перегляд правил, причому аналітик повинен мати простий спосіб перевести правило в «monitor only» режим (логування без створення інцидентів), скоригувати пороги або додати виключення. Технічно це реалізується через флаги в XML-правилі і через REST API для централізованого управління. Крім того, практикою є введення «тіньового» режиму: деякі правила працюють у продакшні, але не створюють алерти — система лише рахує, скільки балів вони генерували, щоб оцінити реальний FP до того, як активувати правило.

З точки зору відповідності й доказовості, кожне правило має містити маппінг на контролі (NIST, PCI, ISO), а інцидент повинен зберігати посилання на всі evidences (логи, snapshots, hashes). Технічно це вимагає, щоб при генерації інциденту система додавала поле `evidence_refs` зі списком ID документів у indexer, а також автоматично формувала пакет для аудитів з описом «що було виявлено», «які контролі порушено», «які кроки виконані». Це істотно спрощує пізні аудити і розслідування.

У контексті виявлення аномального доступу також доцільно впроваджувати adaptive thresholds і risk-scoring. Adaptive thresholds змінюють пороги правил залежно від контексту: наприклад, якщо хост має високий vulnerability score, то поріг для спрацьовування може бути знижений; якщо користувач відомий як адміністратор, то деякі дії можуть сприйматися як

нормальні. risk-scoring акумулює очки з різних джерел (failed_logins, unusual_hour_access, fim_change, ioc_match) і при перевищенні порога ініціює інцидент. Технічно це означає, що правила не просто створюють boolean тригер, а записують contribution у полі `risk_score` події, яке акумулюється і зберігається у correlation context.

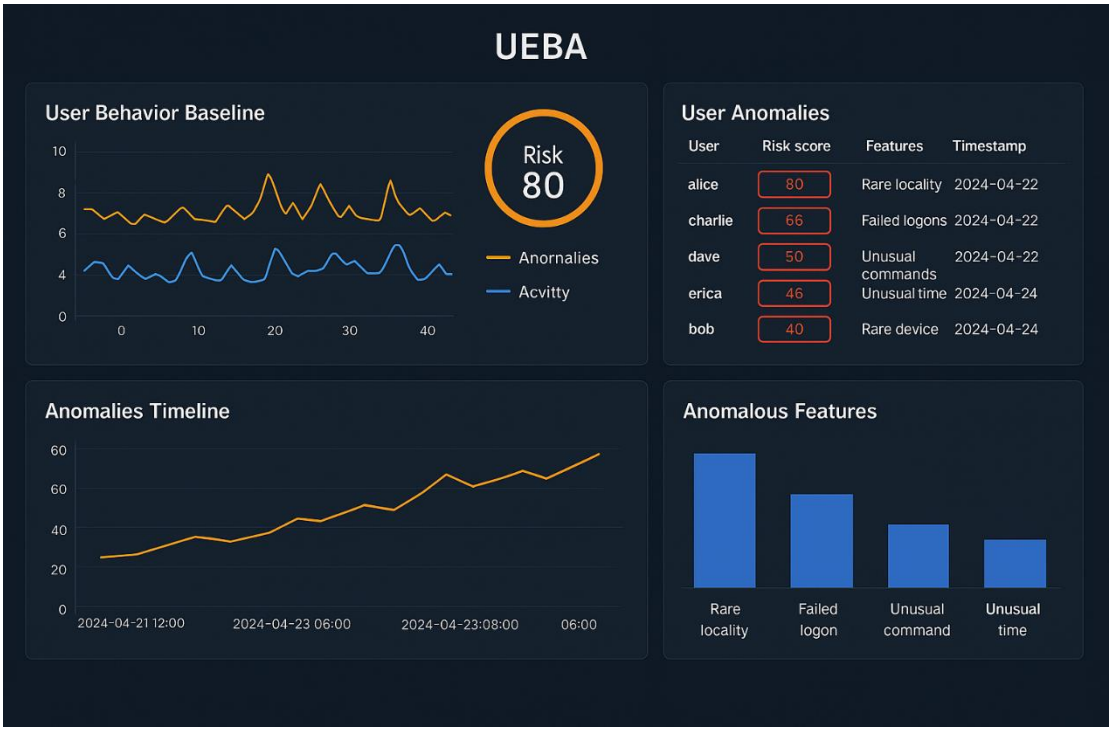

Рис 2.9 Налаштована панель UEBA

Що стосується практичних прикладів правил, допустимо описати схему для виявлення компрометації адміністративного аккаунту: на першому рівні ловляться одиночні події — успішний вхід адмін-аккаунта з непривченого гео/часу; другий рівень агрегує кількість невдалих входів на акаунт у попередні 24 години; третій рівень корелює це з FIM-подіями, що змінюють sudoers або shadow file; четвертий рівень перевіряє, чи відбувалися зовнішні мережеві з’єднання після цих змін; якщо усі умови виконані — правило генерує high-priority incident, запускає Active Response (ізоляція хоста) і викликає SOAR-плейбук для внутрішнього оповіщення і rotate credentials. Технічно це реалізується як ланцюг правил із посиланнями між ними через `parent_rule` або через збереження кореляційних контекстів з ключами `user.name` і `agent.id`.

Окремо варто приділити увагу детекції lateral movement. Ключові індикатори включають: підвищений обсяг мережевих з'єднань з нетипових портів між внутрішніми хостами, повторні використання administrative tools (psexec, winrm, smbclient), асинхронні хвилі автентифікацій з різних джерел, і збір схожих процесних подій на кількох хостах. Технічно для цього застосовуються правила з cross-agent correlation, де ключем кореляції виступає `target.host` або `target.user`. Для масштабних середовищ доцільно використовувати stream-processing у indexer або окремий correlation-engine, який агрегує події по цих ключах і заносить зведені індикатори у Wazuh як окремі документи для подальшої кореляції.

Не менш важлива автоматизація lifecycle управління правилами: реєстр правил з метаданими (автор, дата, опис, coveered use-cases, metrics) повинен бути доступний для SOC і audit команд. Правила мають розроблятися у середовищі з контролем версій, автоматичним тестуванням і чіткою процедурою релізу. Технічно це означає інтеграцію Git → CI → staging → canary → production pipeline, де кожен крок супроводжується тестуванням на історичних та синтетичних даних і телеметрією продуктивності.

Нарешті, процес повинен бути організаційно інтегрований у цикл Threat Detection & Response: правила створюються на основі threat modelling і результатів інцидентів, тестуються у контролованому середовищі, запускаються і постійно тюнінгуються на основі метрик. Регулярні red-team вправи і table-top імітації допомагають перевіряти як правила працюють у реальних сценаріях і коригувати їх за наслідками. Коли це все поєднане, Wazuh перестає бути лише реєстратором логів і стає проактивною платформою, яка не лише виявляє аномальний доступ, а й автоматично ініціює перші кроки реагування, готуючи SOC до швидкої і доказової реконструкції інциденту.

3 ТЕХНОЛОГІЯ ЗАХИСТУ КІНЦЕВИХ ТОЧОК ВІД НЕСАНКЦІОНОВАНОГО ДОСТУПУ НА БАЗІ SIEM WAZUH

3.1. Побудова технології моніторингу звернень до файлових ресурсів

Технологія моніторингу звернень до файлових ресурсів на базі SIEM Wazuh повинна одночасно забезпечити спостереження за операціями читання/запису/видалення/переміщення, фіксацію змін (контроль цілісності), відстеження суб’єкта (хто ініціював дію), процесного контексту (який саме процес/команда), сесійного контексту (з якої сесії/консолі), мережевого контексту (звідки прийшов доступ, куди пішла передача), а також мати механізми автоматизованої реакції та доказової бази для розслідування. В інженерному плані це означає, що система повинна з’єднати декілька «шарів» телеметрії: ядрові події ОС (Linux audit/auditd, Windows ETW/Sysmon), прикладні журнали (веб-сервіси, проксі, СУБД), протокольні події доступу до мережевих файлових шар (SMB/NFS/FTP/SFTP), події з хмарних сховищ (S3/Azure Blob/GCS), та власне дані FIM (File Integrity Monitoring) з whodata-контекстом. Wazuh надає центральну «шину» кореляції для всіх цих потоків: агент на хості збирає локальні події (включно з FIM), додаткові колектори приймають syslog із мережевих пристроїв і NAS, хмарні інтеграції передають «data events» файлових сервісів, а менеджер нормалізує, збагачує, корелює і зберігає їх в індексі для ретроспективи та аналітики. Коректна побудова технології починається з моделі загроз: що саме вважається «несанкціонованим доступом» у конкретному середовищі. Для офісних середовищ — це масове читання або архівація великої кількості конфіденційних документів поза робочим графіком, доступ до каталогів інших підрозділів, зміни прав доступу (ACL), видалення або шифрування файлів (початок атаки шифрувальника). Для серверних середовищ — зміни в конфігураціях служби, підміна бінарів, підвантаження бібліотек, створення «автозапусків», модифікації у webroot, масові операції на файлових шар для публічних сервісів. У виробничих/державних середовищах пріоритетом стає ЦОД-рівень контроль системних каталогів, файлів аутентифікації (shadow, SAM), секретів (ключі,

сертифікати), файлів конфігурацій безпеки. На основі цієї моделі створюється «карта ризику» файлових ресурсів: каталоги, файли, маски, реєстрові гілки (Windows), мережеві шари, хмарні бакети, які підлягають моніторингу з різним профілем глибини.

Ядро рішення на хості — FIM із режимом whodata, який, на відміну від чистого «диф-сканування», фіксує не лише факт зміни (змінився хеш/метадані), а й суб'єкта: ім'я користувача/UID, процес (ім'я/шлях/PPID), TTY/сесію, час, inode/USN. На Linux whodata спирається на комбінації inotify/auditd: auditd генерує SYSCALL-події open/creat/unlink/rename/setxattr/chmod/chown, що дозволяє отримувати «картинку» не лише модифікацій, а й небезпечних читань (наприклад, масове читання *.xlsx у каталозі з контрактами).

```
# /etc/audit/rules.d/10-file-access.rules
# Моніторинг читань/змін у критичних шляхах із whodata-контекстом
-D
-b 16384
-f 1
-w /etc/ -p wa -k etc_changes
-w /etc/ssh/ -p rwa -k ssh_cfg
-w /usr/local/bin/ -p wa -k local_bin_changes
-a always,exit -F dir=/srv/finance -F perm=r -F auid>=1000 -F auid!=unset -S open,openat -k finance_reads
-a always,exit -F arch=b64 -S unlink,unlinkat,rename,renameat -F auid>=1000 -F auid!=unset -k mass_delete
# augenrules --load
```

Рис 3.1 Правило моніторингу доступу до файлів

На Windows найбагатший контекст дає пара «Windows Security Auditing + Sysmon»: перший увімкне події доступу до об'єктів (Object Access, 4663 тощо) та змін прав (4670/4719), другий фіксує процесний граф (ProcessCreate/NetworkConnect/FileCreateTime/RegistryEvent). Агент Wazuh знімає ці події, нормалізує, зшиває з FIM та відправляє у менеджер для правил кореляції. Власне FIM треба конфігурувати шарово. «Ядро» — найкритичніші шляхи/реєстр: `/etc`, `/etc/ssh`, `/usr/local/bin`, `/var/www`, каталоги ключів/сертифікатів; на Windows — `%SystemRoot%`, `%ProgramData%`, каталоги служб, *Autostart Extensibility Points*, реєстрові гілки Run/Services/LSA/Winlogon. Для цих шляхів вмикається whodata, повний набір метаданих (права, власник, SELinux/AppArmor мітки), подвійні хеші (SHA-256/512), report_changes для конфіг-файлів (щоб

зберігався diff). Другий шар — великі і «гарячі» каталоги бізнес-даних, де доцільне періодичне сканування з розумними винятками (logs/temp/cache) та лімітами глибини. Третій шар — мережеві шари (SMB/NFS): їх краще контролювати не лише з боку клієнтів, а й на рівні сервера файлів (Samba vfs_full_audit для SMB, NFS audit через ядрові гачки), щоб у разі інциденту мати «об'єктивну» версію доступів до спільних ресурсів. Важливий момент — аудит читань. Багато «класичних» FIM-налаштувань відстежують тільки модифікації, хоча ексфільтрація починається з читань. На Linux це вирішується правилами auditd на `open(O_RDONLY)/read`, орієнтованими на шляхи з чутливими даними; на Windows — через Object Access Auditing на цільових каталогах/файлах та Sysmon Event ID 11 (FileCreate) у частині створення копій/тимчасових файлів архіваторів. З технічного боку, щоб не «вбити» продуктивність, події читання треба робити селективними та пороговими (кореляційні правила «N читань із каталогу *X* за T секунд»).

```
<!-- sysmon-config.xml -->
<Sysmon schemaversion="4.90">
  <EventFiltering>
    <FileCreate onmatch="include">
      <TargetFilename condition="begin with">C:\Windows\System32\</TargetFilename>
      <TargetFilename condition="begin with">C:\ProgramData\</TargetFilename>
    </FileCreate>
    <ProcessCreate onmatch="include">
      <Image condition="is">C:\Program Files\7-Zip\7z.exe</Image>
      <Image condition="is">C:\Program Files\WinRAR\rar.exe</Image>
      <Image condition="end with">powershell.exe</Image>
    </ProcessCreate>
    <HashAlgorithms>sha256,imphash</HashAlgorithms>
  </EventFiltering>
</Sysmon>
```

Рис 3.2 Sysmon та Object Access Auditing для файлового моніторингу

Далі — кореляція. Валідаційні правила базового рівня: одноразова зміна критичного файлу → high-severity; створення нового виконуваного файлу у системних шляхах → high; зміна ACL на каталозі з конфіденційними даними → high; масове перейменування/видалення → підозра на шифрувальник. Поведінкові правила: «масове читання» (наприклад, >100 файлів PDF/CSV у каталозі фінзвітів

за 5 хв), «нестандартний тип доступів» (архівація `.7z/.zip` одразу після читань 50+ файлів), «зміна в позаробочий час» (від 23:00 до 05:00), «новий процес архіватора/експортера», «спроба читання закритих каталогів». Для мережевих шар — «аномальна частота віддалених читань з однієї станції», «сканування списку каталогів і дерев». Для webroot — «зміна файлу у каталозі тем/плагінів CMS», «поява вебшелл-сигнатур», «модифікація .htaccess». Для хмарних бакетів — «GetObject/ListBucket вище норми», «PutObject у незвичний час», «зміна політики бакета/ACL/версіонування». Суть — скласти ланцюжки: від «хто» і «який процес» до «який тип дії і над яким ресурсом» і «що далі відбулося в мережі». Wazuh дозволяє у правилах визначати ключі кореляції (user, src.ip, agent.id, process.hash) і часові вікна; для «повільних» інсайдерських сценаріїв це можуть бути 6–24 год, для шифрувальників — хвилини. Практично корисно робити проміжні агреговані події: раз на 1–5 хв формувати підсумки «file_reads_summary», «file_writes_summary», «ext_archive_ops_summary» і вже їх корелювати зі змінами мережевої поведінки або з виходними підключеннями на «свіжі» домени.

Щоб зменшити шум і зробити сповіщення дієвими, необхідно системно побудувати винятки. «Шумні» каталоги (лог-каталоги, кеш індексації, temp-директорії, робочі каталоги офісних пакетів) треба або повністю ігнорувати, або переводити на періодичний диф. Релізні вікна та оновлення ОС/ПЗ слід маркувати тегами «maintenance» (метадані в алерті), а правила — зменшувати рівень серйозності/переводити у monitor-only на час вікна. Для команд build/DevOps — виділені сервісні обліковки; їхній «шумний» патерн оголошується нормою лише в конкретних часових рамках і на конкретних вузлах (labels/групи агентів). На Windows дуже корисно налаштувати *Object Access Auditing* адресно — лише на чутливі каталоги, а не «усюди». На Linux — аудитні правила слід ставити точково, інакше SYSCALL-потоки створять вузькі місця.

Мережеві файлові сервіси вимагають окремої уваги, бо інсайдерські інциденти найчастіше починаються саме там. Для SMB на серверах Samba доцільно вмикати `vfs_full_audit` на шери, що містять чутливі дані, і логувати open/close/rename/unlink/chmod/chown — з префіксом інформації про

користувача/клієнтський IP. Ці логи відправляються syslog-ом у Wazuh і декодуються у нормалізовані події доступу. На Windows-файлових серверах можна використати File Server Resource Manager + Audit Object Access для вибраних шляхів, із маршрутизацією у Security Log; агент Wazuh зчитує їх через WinEvt API. Для NFS — комбінувати ядрові audit-правила на операціях VFS із логами NFS-сервера (rpc-layer). Ключовий патерн кореляції для NAS: «новий клієнт/користувач» → «масові читання» → «масове створення архівів» → «вихідні з'єднання». Для DFS/розподілених сховищ — відстежувати мітки реплікації та аномальні «стрибки» обсягів змін.

```
# /etc/samba/smb.conf
[finance-share]
   path = /srv/finance
   read only = no
   vfs objects = full_audit
   full_audit:prefix = %u|%I|%S
   full_audit:success = open,read,write,rename,unlink
   full_audit:facility = local5
   full_audit:priority = NOTICE
# /etc/rsyslog.d/30-samba.conf
local5.notice /var/log/samba/audit.log
*.* @@wazuh-syslog.local:514
```

Рис 3.3 Samba vfs_full_audit

Хмарні файлові сервіси (S3/Azure Blob/GCS) дають критично важливі «data events», які треба під'єднати: `GetObject`, `PutObject`, `DeleteObject`, `ListBucket`, `PutBucketPolicy`, зміни ACL/версіонування. Для S3 це вмикається як CloudTrail Data Events на бакетах; потік іде у Wazuh через хмарний конектор. Антирискові правила: «масові GetObject із одного принципала поза робочим часом», «зміна політики на public-read», «PutObject з нестандартного VPC», «вимкнення версіонування перед хвилею змін». Ці події треба зшивати з локальними: хто саме на своєму ноутбуці запустив клієнт синхронізації, які файли він «торкав» перед тим, чи був зовнішній тунель/VPN. Таке наскрізне бачення часто є єдиним способом відрізнити легітимний ETL-процес від тіньового вивантаження.

```
{
 "TrailName": "corp-file-monitoring",
 "EventSelectors": [{
   "DataResources": [{
     "Type": "AWS::S3::Object",
     "Values": ["arn:aws:s3:::corp-sensitive-bucket/"]
   }]
 }]
}
# Приклад події GetObject
{
 "eventName": "GetObject",
 "bucket": "corp-sensitive-bucket",
 "key": "HR/Payroll_2025_Q4.xlsx",
 "sourceIPAddress": "198.51.100.24"
}
```

Рис 3.4 AWS CloudTrail Data Events

Істотну роль у технології відіграє процесний контур: «яким процесом» виконано доступ. Sysmon на Windows (Event ID 1/11/12/13/22) та auditd на Linux (символи EXE/COMM у SYSCALL) дозволяють зрозуміти, чи це штатний офісний пакет, чи «rar.exe/7z.exe», «powershell.exe» з параметрами стиснення, «scp/rsync» у нічний час, «curl»/«wget» на зовнішні адреси. Відповідні правила: «архіватор + масові читання → підозра на ексфільтрацію», «скриптові оболонки + зміни конфігів → персистентність/бекдоринг», «довгі ланцюжки процесів із незвичних батьків» (наприклад, офісний пакет породжує шелл).

Технологія моніторингу повинна включати активну реакцію. На критичні події FIM/доступу до файлів необхідно запускати playbook-дії: відкотити конфіг із «золотої» копії, припинити процес архіватора, тимчасово відключити користувача або перевести агента у «ізоляцію мережі», створити snapshot диска/тома для форензіки, автоматично підняти тікет у ITSM із вкладеними доказами (алерт, хеш-дифи, логи процесів). Реакції мають бути ідемпотентними, з тайм-аутами і контролем результату («успішно/помилка з кодом»). У продакшні корисно мати «жовтий» рівень — без блокування, лише сповіщення SOC і ескалація до ручного підтвердження, щоб не зірвати критичні бізнес-процеси.

Окрема практична техніка — «приманки» (canary/honey-files). Невеликі файли з привабливими назвами (наприклад, «Payroll_2025_Q4.xlsx») у захищених каталогах, доступ до яких генерує високопріоритетну подію. На серверних вузлах — спеціальні honey-конфіги, зміна яких у нормі не відбувається ніколи. Це значно скорочує час виявлення інсайдерів і автоматичних «пилососів» файлів. Разом із цим треба ретельно контролювати права на такі файли і не допустити випадкового використання легітимним процесом.

```
<group name="exfiltration,files,high_risk">
  <rule id="91001" level="5">
    <if_group>auditd,file_access</if_group>
    <field name="file.path">@/srv/finance/@</field>
    <timeframe>300</timeframe>
    <frequency>100</frequency>
  </rule>
  <rule id="91002" level="7">
    <if_group>sysmon,process_creation</if_group>
    <match>7z.exe|rar.exe|tar</match>
    <if_matched_group>mass_reads</if_matched_group>
  </rule>
  <rule id="91003" level="12">
    <if_group>sysmon,network_connection</if_group>
    <if_matched_group>archive_after_reads</if_matched_group>
    <description>Exfiltration chain</description>
  </rule>
</group>
```

Рис 3.5 Correlation rule

З погляду експлуатації критичною є якість часу (NTP), шифрування каналів (TLS), контроль ILM-політик для індексів (гарячі/теплі/холодні), розумне збереження метаданих (diff текстових конфігів, хеші двійкових), і план тестування. Перед запуском у продакшн рекомендовано пройти «пілот»: обрати 10–20 вузлів/шарів, увімкнути whodata на «ядерних» шляхах, перевірити середній час від події до алерта, оцінити FP-рівень, спланувати винятки. Після розширення — тижневий тюнінг «шумних» правил і джерел, оновлення винятків після релізів, перевірки Active Response у «сухому» режимі (monitor-only). Регулярні table-top вправи (інсценування інциденту) і «червоні» тести з імітацією шифрувальника/експорту допомагають калібрувати пороги і сценарії.

Метрики технології: покриття (відсоток критичних шляхів під FIM/доступом), середній час виявлення (MTTD) по класах подій (модифікація, масове читання, архівація), середній час реакції (MTTR) автоматичної/ручної, «шумність» (alerts per endpoint/day), частка підтверджених інцидентів, ризик-скор за підрозділами. На рівні відповідності — мапінг на контролі (PCI DSS 10/11, NIST 800-53 SI-7/CM-6/AC-2, ISO/IEC 27001 A.8/A.12), готові звіти по відхиленнях і історії виправлень (хто і коли повернув/змінив).

```
#!/usr/bin/env bash
FILE="$1"
AGENT="$2"
cp -f "/var/ossec/backups${FILE}.golden" "$FILE"
iptables -I OUTPUT -d 0.0.0.0/0 -j DROP
iptables -I INPUT -s 0.0.0.0/0 -j DROP
curl -X POST -H 'Content-Type: application/json'   -d '{"agent":"'$AGENT'","file":"'$FILE'"}'
https://soc.local/hooks/active-response
```

Рис 3.6 Active response script

Важливо розуміти, що ця технологія — не статична. Нові бізнес-системи постійно змінюють «норму» файлового життя: ETL-процеси з великими обсягами читань, індексатори/аналітика, автоматичні бек-апи. Тому модель звичайної поведінки треба регулярно переглядати, адаптувати правила до сезонних піків (звітні періоди), а винятки робити часово та контекстно обмеженими (лише на конкретний реліз/вікно, на конкретну групу агентів). Тільки так моніторинг звернень до файлових ресурсів на базі Wazuh залишатиметься чутливим до реальних атак і при цьому керованим для SOC.

3.2. Моделювання сценаріїв несанкціонованого доступу: зовнішнє вторгнення та дії інсайдера

Моделювання сценаріїв несанкціонованого доступу — це методологічно суворий процес, який поєднує threat modelling, розкладку можливих TTP (tactics, techniques, procedures), карту артефактів у телеметрії та набір детекційних гіпотез, що перетворюються на правила, playbook-и й forensic play. У межах захисту кінцевих точок на базі SIEM (Wazuh) корисно розглядати два фундаментально різні класи сценаріїв: зовнішнє вторгнення (external intrusion) — атаки, що ініціюються з-поза периметра організації, і дії інсайдера (insider threat) — зловмисні або випадкові дії суб’єктів, які мають легітимний доступ. Хоча технічно багато артефактів перетинаються, модель ризиків, поведінка атакувальників і набори контрзаходів відрізняються, тому кожен клас треба моделювати окремо, з відразу закладеним механізмом кореляції сигналів між ними.

У моделях зовнішнього вторгнення зазвичай виділяють етапи: розвідування (reconnaissance), початкова компрометація (initial access), розширення присутності (establish foothold / persistence), ескалація привілеїв (privilege escalation), рух по мережі (lateral movement), досягнення цілей (action on objective — data collection/exfiltration, disruption), та очищення слідів (anti-forensics).

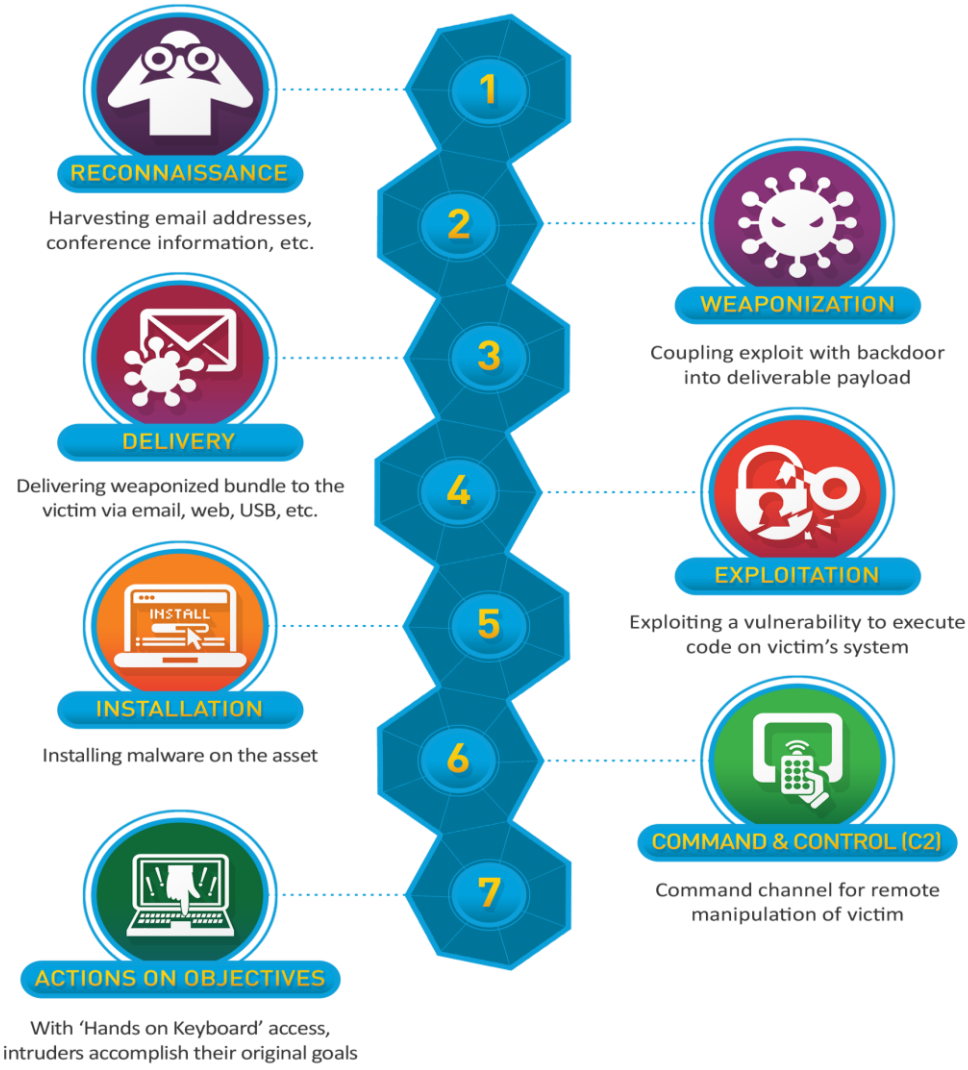

Рис 3.7 Cyber Kill Chain

Для кожного етапу необхідно формалізувати набір очікуваних логів/телеметрії та артефактів, які доступні в середовищі: мережеві connection logs (firewall, proxy, IDS/IDS), автентифікаційні події (AD/Azure/IdP, VPN), процесна телеметрія (Sysmon, auditd), FIM-події, поведінкові метрики користувачів і пристроїв (UEBA), події з хмарних логів (CloudTrail, Azure Activity) та індикатори на кінцях (vulnerability detection, SCA) [31]. Класична інженерна практика — розкласти кожен TTP на *логічні індикатори*, наприклад: фішингова початкова компрометація часто дає поєднання: зовнішній SMTP/Proxy лог із завантаженням вкладення → запуск Office macro (ProcessCreate для `winword.exe` з `powershell` у параметрі) → новий виконуваний файл у тимчасовому каталозі → DNS-запит до DGA-домену → установки з'єднання на C2. Технічно: для кожного індикатора

вказується джерело (Agent/Syslog/Cloud), формат поля (user, srcip, process.name, command_line, file.path, file.hash), очікуваний рівень достовірності та пріоритет кореляції.

Дії інсайдера моделюються інакше: інсайдер має легітимні облікові дані й знання організації, тому відмінність між нормальними і зловмисними діями може бути дуже тонкою. Сценарії включають зловмисне викрадення даних (steal & exfiltrate), зловживання привілеями (privilege misuse), випадкове порушення процедур (misconfiguration), або саботаж. Тут виявлення опирається насамперед на контексті: а) нетиповий доступ (час/гео/endpoint); б) дивні комбінації дій (наприклад, доступ до payroll у позаробочий час + масове копіювання в мережеву папку + ініціація zip), в) аномалії в обсягах і швидкостях операцій (переважно читання, не зміни). Технічно для інсайдерських сценаріїв UEBA і довгі часові вікна є первинними інструментами — агрегування поведінкових ознак за тижні/місяці, порівняння поточної сесії зі «звичним» профілем, і використання ансамблів ознак (out-of-hours access, access to atypical resources, escalation of privileges, use of removable media, bulk file reads) [31]. Важливий прийом — побудова «білого списку нормальних» для певних ролей та автоматичне підвищення тривожності при відхиленні від профілю. Практично це означає: зберігати гісторію дій користувача (команди shell, відвідані каталоги, середнє число файлів прочитаних за сесію) і вираховувати статистичні метрики (rolling averages, z-score) — якщо z-score перевищує поріг, тригерити investigative alert [32].

У моделюванні обох типів треба чітко прописувати *сценарії кореляції*, тобто які події і в якому порядку й з яким TTL формують інцидент. Приклад для зовнішнього вторгнення: `failed_auth x 50 (1h) from many IPs` → `successful_auth from one of those IPs` → `process_create of suspicious loader` → `FIM change to /etc/ssh/sshd_config` → `network_connect to new domain` = high-confidence incident. Для інсайдера: `user A` (звичний час 09:00–18:00) здійснив `mass_read` у каталозі HR за 03:00–04:00 (позаразовий), потім `archive_create` і `outbound_connection` через нестандартний порт → medium/high [31]. У Wazuh такі ланцюжки будуються із

застосуванням груп правил (group/if_matched_group/aggregate) та correlation context; реалізація ефективна лише за умови, що джерела даних синхронізовані по часу, мають контекст asset-tagging і що декодери повертають однакові неймовірні поля.

```
<group atttack_type="insider">>
<rule id="101000">
  <decoded_as> wazuh-correlation>
  <if_sid>140</ifsid>
  <if_sid>205</ifsid>
  <if_sid>301</ifsid>
  <level>10
  <description>Інсайдер: масове ч:
  читання + створення архіву отвконнін
  + outbound connection
  <options><alert_by_email>
< rule>                                    [32]
```

Рис 3.8 Правило детектування архівування та передачу назовні

Детекційна інженерія повинна охоплювати чотири шари: preventive (hardening, least privilege, SCA), detection (FIM, process telemetry, network logs, UEBA), response (Active Response, SOAR playbooks) і forensic readiness (retention, chain-of-custody, snapshots)[31]. На практичному рівні це означає: закріпити SCA-політики, забезпечити чутливі шляхи під FIM (whodata) з подвійними хешами і diff-логами; підключити Sysmon/Security Audit (Windows) і auditd (Linux) з вибірковим but sufficient coverage логуванням відкриттів/читань/записів; підключити мережеві логи (firewall, proxy, IDS) і хмарні data events; організувати централізований indexer з ILM та retention, де зберігаються сирі події і агрегати для пошуку. Саме на основі цих шарів будується можливість швидкої реконструкції ланцюжка: від першого

сету DNS-запитів до доказів ексфільтрації (логів S3 GetObject з sourceIPAddress і bytesTransferred).

Для кожного сценарію необхідно визначити індикатори Compromise (IoC) і індикатори поведінки (IoB). IoC — це конкретні хеші файлів, домени, IP, C2-шаблони. IoB — патерни дій: незвична комбінація подій, частотні аномалії, незвичний процесний граф. Моделювання підказує, які саме IOC має сенс автоматично перевіряти і як поєднувати їх у risk-score. Наприклад, ексфільтрація конфіденційних файлів отримує додаткові очки ризику, якщо: a) файли були прочитані масово; b) був запущений архіватор; c) був встановлений зовнішній з'єднаний тунель; d) незвичний акаунт ініціював дію — сукупність цих ознак підвищує risk-score і ініціює containment playbook.

Технічна модель також має описувати *шумні зони* і механізми їхнього придушення: регулярні бекапи, індексація, індексаційні завдання, аналітичні ETL-процеси — усе це породжує велику кількість легітимних подій (наприклад, масові `Get` з бекап-агентів). Для уникнення false positives треба вводити asset tagging (backup agents, analytics agents), time-bound maintenance tags, exception lists для конкретних processes/users, а також проміжну агрегацію (summary events) для важких операцій. Практичний патерн: для кожного rule, що бачить масові операції, перевіряти чи виконавець не є у whitelist (backup-service) і чи не знаходиться у maintenance window, і лише за відсутності цих умов переходити далі в ланцюжок кореляції.

У моделюванні важливо передбачити форензічну підготовку: collection playbook, що описує конкретні команди і артефакти, які треба зняти при виявленні (лог файли, /proc/$pid/ cmdline, network captures, memory dumps, disk snapshots), та механізм збереження їх у tamper-evident сховище. Для зовнішніх вторгнень це важливо для attribution і відновлення; для інсайдерів — для доказування мотиву і масштабу впливу. Технічно Wazuh може ініціювати Active Response скрипти для миттєвого збирання snapshot-ів і виклику зовнішніх форензик-інструментів — але політика таких дій має бути ретельно затверджена (щоб уникнути пошкодження доказів або бізнес-процесів) [32].

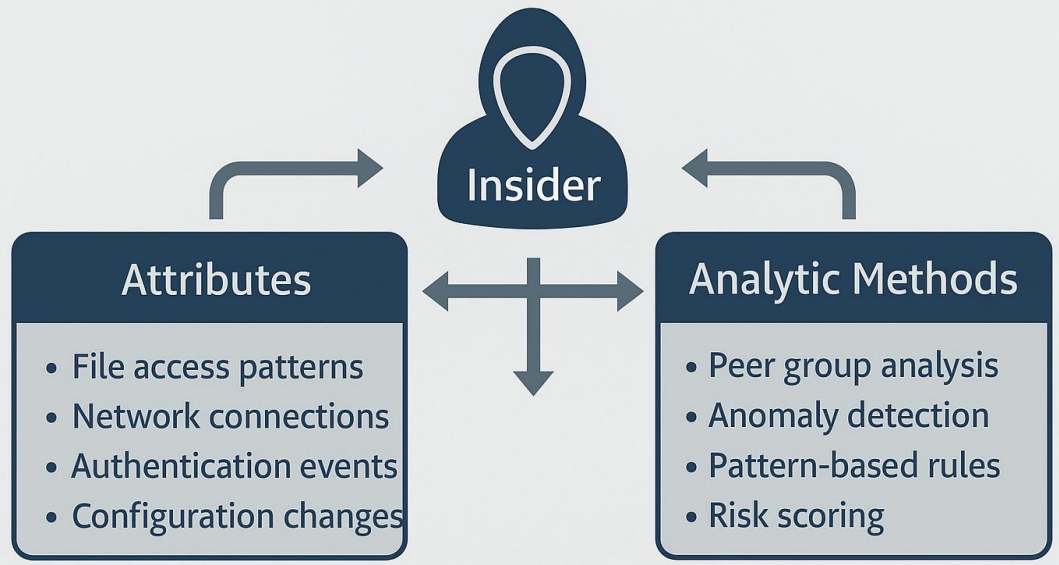

Рис 3.9 Побудова ланцюжка кореляції подій у Wazuh для формування єдиного інциденту безпеки

Реалізація playbook-ів containment & remediation повинна бути артикулярована: для low/medium/high severity мають бути чітко визначені кроки, наприклад: low — оповіщення SOC + логування; medium — автоматична ізоляція інтерфейсу/віртуальної мережі + створення тикету; high — snapshot диска + rotate credentials + відкат конфігів + повідомлення керівництва [31]. Кожен крок має бути idempotent, логований і мати rollback-стратегію. У технологічному впровадженні це вимагає інтеграції Wazuh з ITSM, orchestration/SOAR платформами, рішеннями резервного копіювання та мережею (SDN/VLAN orchestration).

Для тестування моделі — постійні red-team / purple-team вправи з чітко визначеними сценаріями (phishing→ransomware, compromised admin→lateral movement, data theft by insider), а також replay історичних інцидентів у sandbox (backfill) для оцінки TPR/FPR. Важливо автоматизувати метрики: MTTD (mean time to detect), MTTR (mean time to respond), percent automated containment, false positives per analyst per day [31]. Успішний цикл — виявлення вражень → корекція правил → оновлення playbook → нові тести. Моделювання має також включати threat intelligence feed testing: чи виявляє наша конфігурація відомі IOC; чи є ефект від підключення різних фідів; який latency enrichment дає.

Юридичний і етичний контекст теж повинен бути передбачений у сценаріях: збір персональних даних у логах, політика зберігання (retention), доступ до доказів, інформація про повідомлення постраждалих/регуляторів у разі витоку — всі ці вимоги впливають на design (наприклад, які логи дотримуються довшого retention, як шифруються архіви evidences). Для інсайдерів важлива процедура chain-of-custody і обмеження доступу до доказів, щоб уникнути витоку під час розслідування [31].

На рівні детекційної інженерії потрібно також формалізувати «контрзаходи» для ускладнення життя атакуючих: жорстке управління привілеями, MFA, обмеження відкриття ресурсів (just-in-time access), network micro-segmentation, DLP для чутливих файлів, використання EDR з поведінковою блокіровкою. Ці контрзаходи скорочують площину атаки і одночасно породжують нові телеметричні події, які повинні бути враховані у моделях (наприклад, JIT доступи створюють пік подій автентифікації, який треба не плутати з атакою) [32].

Підсумовуючи, ефективне моделювання сценаріїв несанкціонованого доступу — це поєднання: формалізованих TTP → mapping на джерела телеметрії → створення кореляційних ланцюжків (rules + timeframes) → enrichment (asset, vuln, IOC) → детекційні гіпотези (UEBA, thresholds, agg) → playbook-и реагування та forensics → тестування (red-team, replay) → метрики і governance. Wazuh в цьому циклі є центральним елементом: збирає, нормалізує, корелює, збагачує й ініціює реакції. Але успіх залежить від якості декодерів, охоплення телеметрією, ретельного тюнінгу правил та інтеграції з orchestration/SOAR і ITSM — інакше навіть найкраще змодельований сценарій не запрацює на практиці.

3.3 Встановлення та налаштування правил кореляції та FIM в WAZUH для виявлення атак

Технологічна сутність побудови системи виявлення атак у Wazuh базується на тісній інтеграції двох підсистем — File Integrity Monitoring (FIM) та Rules and Correlation Engine, які працюють спільно на рівнях агента, менеджера та індексера. Обидва компоненти функціонують як частина аналітичного конвеєра, де FIM відповідає за генерацію низькорівневих подій зміни стану файлової системи, а механізм правил (rules engine) виконує контекстну обробку, багаторівневе зіставлення та логічну кореляцію сигналів, формуючи кінцеві алерти безпеки.

Коли агент Wazuh запускає підсистему `ossec-syscheckd`, він створює локальну базу хешів для визначених директорій. Для кожного файлу система зберігає набір атрибутів:

- `sha256`, `sha1`, `md5` контрольні суми;
- `uid`, `gid`, права доступу (mode);
- розмір, час модифікації, тип;
- при активованому режимі `whodata` — PID, процес, користувача і термінал.

Ці дані формують «базову лінію» стану системи. При наступних скануваннях або у режимі реального часу (через `inotify` у Linux або `USN Journal` у Windows) Wazuh порівнює поточні атрибути файлів із еталонними. Якщо хеш змінився або файл був створений/видалений, агент генерує подію типу `syscheck` і відправляє її на Wazuh Manager через зашифрований канал (TLS/1514 TCP). Формат події — JSON із полями:

```
{
  "agent": {"id": "002", "name": "srv-finance"},
  "rule": {"groups": ["syscheck"], "level": 7},
  "syscheck": {
```

```
      "path": "/srv/finance/payroll_2025.xlsx",
      "event": "modified",
      "old_sha256": "A9C...",
      "new_sha256": "F2B...",
      "user": "john.doe",
      "process": "excel.exe"
    },
    "timestamp": "2025-10-06T18:20:35Z"
  }
```

На стороні менеджера ця подія обробляється Decoder Framework — XML-декодерами, які перетворюють «сирі» поля логів у нормалізовані ключі (наприклад `file.path`, `srcuser`, `program`, `event`, `srcip`). FIM-події належать до групи `syscheck`. Менеджер послідовно проходить через pipeline декодерів, де для кожного визначеного шаблону (`if_sid`, `match`, `regex`) застосовуються правила з `ruleset/rules/`.

Алгоритм обробки виглядає так:

1. Вхідна подія надходить у чергу менеджера.
2. Менеджер визначає її тип і застосовує відповідний декодер (`syscheck_decoder`).
3. Поля JSON мапляться на стандартні змінні (`file`, `event`, `sha256`, `srcuser`, `process`).
4. Rules Engine шукає збіги серед XML-правил.
5. Якщо кілька подій задовольняють умови кореляції (спільний агент, часовий інтервал, тип подій), система генерує агрегований алерт.

Типове правило кореляції у `local_rules.xml` може виглядати так:

```
<rule id="94005" level="12">
  <if_group>syscheck</if_group>
  <field name="event">modified</field>
  <same_source>true</same_source>
  <frequency>50</frequency>
```

```
        <timeframe>300</timeframe>
        <description>High number of file modifications detected - possible ransomware</description>
    </rule>
```

Це правило означає, що якщо протягом 300 секунд на одному агенті зафіксовано понад 50 змін файлів, Wazuh сформує алерт рівня 12. Саме `frequency` і `timeframe` утворюють часовий буфер — ключовий механізм кореляції. Менеджер веде кеш подій у пам'яті, і якщо кількість збігів перевищує поріг, відправляє нову подію типу `alert`.

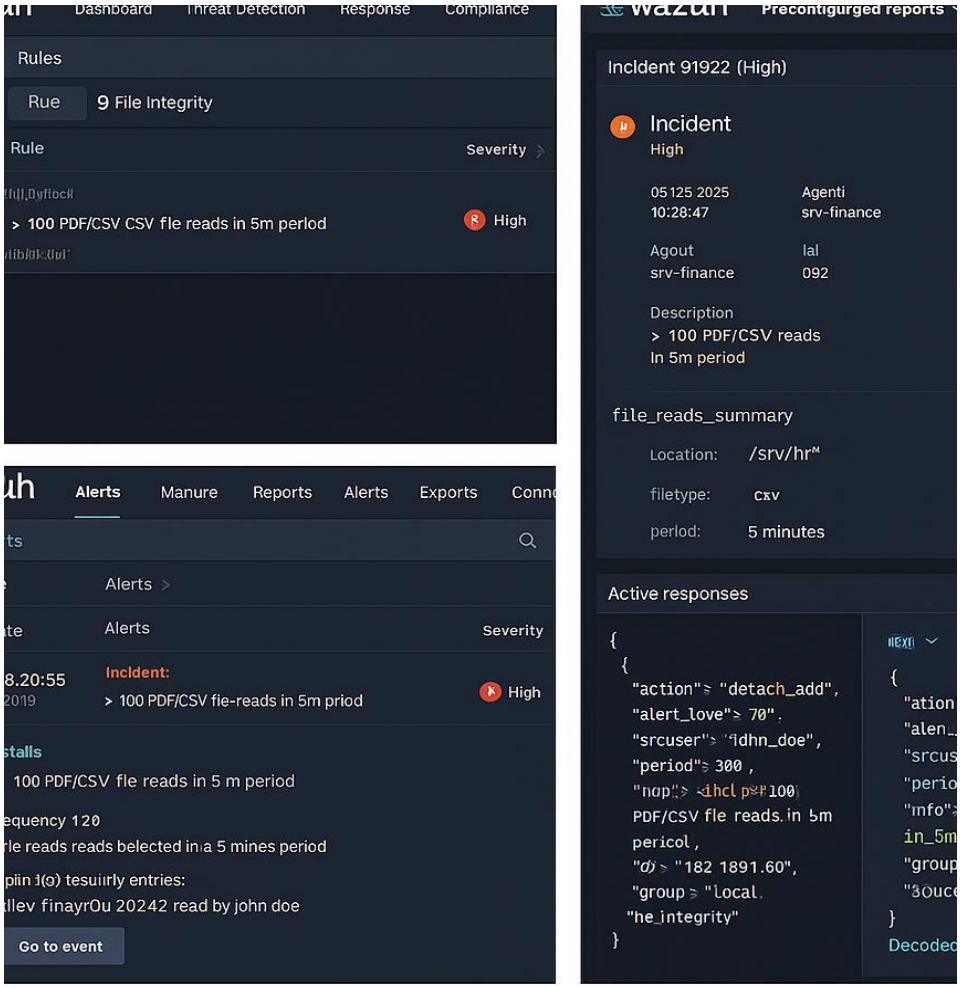

Рис 3.10 - Детектування масового читання файлів та автоматизована відповідь у консолі Wazuh

Для більш складних сценаріїв, таких як виявлення ексфільтрації або ransomware, застосовуються зв'язні правила. Наприклад, Wazuh може поєднати

кілька груп: FIM (syscheck), процесну (sysmon), мережеву (network_connection), щоб виявити причинно-наслідковий ланцюг:

1. syscheck:modified — зміни у файлах.
2. sysmon:process_creation — запуск архіватора або шифрувальника.
3. sysmon:network_connection — outbound-з'єднання.

Таке правило виглядає наступним чином:

```
<rule id="94010" level="14">
  <if_group>sysmon,network_connection</if_group>
  <if_matched_group>syscheck</if_matched_group>
  <timeframe>600</timeframe>
  <same_source>true</same_source>
  <description>File modifications followed by outbound connection - possible exfiltration</description>
</rule>
```

Менеджер підтримує кореляційний кеш — це хеш-таблиця, де події індексуються за агентом і часовим вікном. Коли нова подія надходить, система перевіряє, чи існує активний контекст для цього агента, і якщо так — розширює його або формує інцидент. Саме такий принцип дозволяє Wazuh створювати складні, багаторівневі зв'язки без необхідності окремого correlation server.

Модуль FIM також підтримує реальний час через auditd. У режимі whodata="yes" агент реєструє всі системні виклики open(), write(), chmod(),unlink(),rename(). Це дозволяє відновити точну послідовність дій користувача, включно з PID процесу і командою, яка спричинила зміну файлу. Наприклад, FIM може зафіксувати, що процес vim змінив /etc/ssh/sshd_config, після чого система автоматично спрацює на правило "Config Tampering".

Щоб зменшити хибні спрацьовування, у конфігурації варто застосовувати фільтри:

```
<ignore>/var/log</ignore>
```

```
<ignore>/tmp</ignore>
<ignore type="sregex">^/home/.*\.cache/</ignore>
```

Це дозволяє виключати нестабільні шляхи або службові файли, які постійно оновлюються.

Для продуктивних середовищ із великою кількістю агентів доцільно застосовувати кластеризацію правил — кожен Wazuh Manager (worker node) має копію `ruleset`, синхронізовану з master, а результати обробки надсилаються до централізованого Indexer. Це забезпечує горизонтальне масштабування та відмовостійкість.

Після створення або зміни правил обов'язково виконується перевірка за допомогою інструменту:

```
/var/ossec/bin/wazuh-logtest -r /var/ossec/etc/rules/local_rules.xml
```

Ця команда дозволяє протестувати логіку спрацьовування без фактичного надходження подій із агентів.

Wazuh також підтримує Active Response, що безпосередньо прив'язується до ідентифікаторів правил. Якщо правило з ID `94010` спрацьовує, система може автоматично виконати команду блокування IP, завершення процесу чи відкат файлу. Приклад конфігурації:

```
<active-response>
  <command>host-deny</command>
  <location>local</location>
  <rules_id>94010</rules_id>
</active-response>
```

Таким чином формується замкнене коло: FIM → Rules → Alert → Active Response → Audit.

У результаті цього технічного ланцюга Wazuh може не лише фіксувати зміни, але й розуміти їхній контекст: чи це звичайне оновлення, чи аномальна активність, яка відповідає технікам MITRE ATT&CK, таким як T1059 (Command and Scripting Interpreter) або T1486 (Data Encrypted for Impact). Система створює

корельовану подію, що описує причину, наслідок і джерело — формуючи повну картину атаки у масштабі всієї інфраструктури.

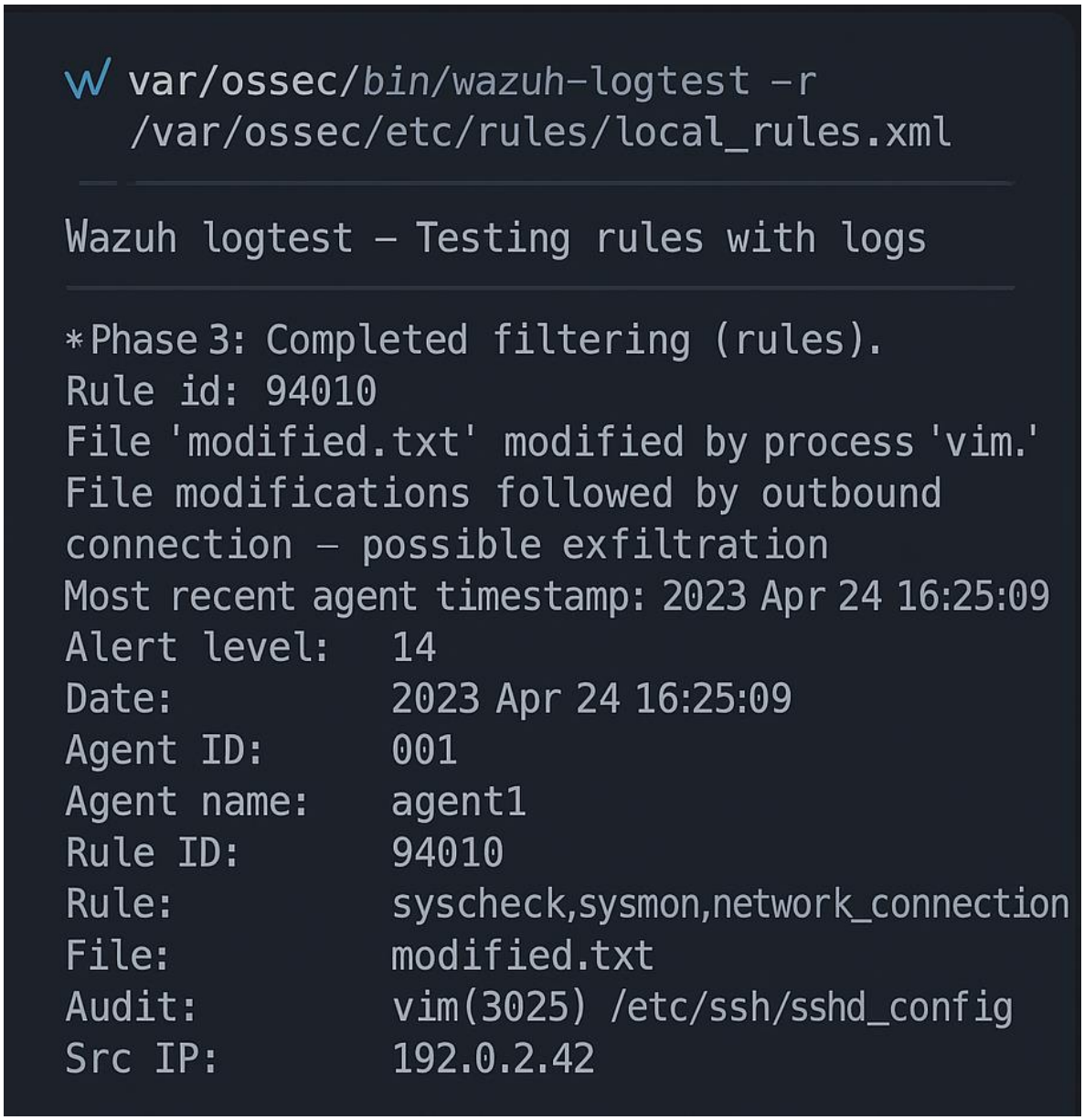

Рис 3.11 Демонстрація роботи кореляційного механізму Wazuh для виявлення ексфільтрації

Налаштований таким чином FIM у поєднанні з кореляційними правилами забезпечує рівень спостережності, необхідний для SOC середовищ, де важливо не просто бачити події, а й розуміти логіку атакувальних дій. У разі правильної конфігурації Wazuh переходить від моделі простого журналювання до моделі поведінкової аналітики, що здатна самостійно розпізнавати аномалії та ініціювати дії у відповідь із мінімальною участю оператора.

3.4 Рекомендації щодо застосування технології захисту кінцевих точок від НСД на базі

Ефективне застосування SIEM-платформи Wazuh для захисту кінцевих точок від несанкціонованого доступу вимагає комплексного підходу, який поєднує технічну конфігурацію, правильну архітектуру, процедури адміністрування та політики реагування. Основна мета використання Wazuh — не просто збір журналів чи сповіщень, а побудова єдиної аналітичної екосистеми, яка виявляє, аналізує та нейтралізує інциденти безпеки у режимі, наближеному до реального часу.

Передусім варто правильно спроєктувати архітектуру системи. Для середніх і великих інфраструктур оптимально використовувати кластерну схему Wazuh, що складається з кількох серверів-менеджерів, індексера для збереження даних і дашборду для візуалізації. Такий підхід забезпечує високу відмовостійкість і масштабованість: при збільшенні кількості агентів навантаження рівномірно розподіляється між вузлами. Важливо також забезпечити синхронізацію часу на всіх вузлах через NTP, оскільки від правильного таймстампу залежить якість кореляції подій. Для з'єднання між агентами і менеджером рекомендується використовувати TLS 1.2 або вище, з обов'язковою перевіркою сертифікатів.

На рівні агентів слід визначити пріоритетні області моніторингу залежно від типу системи. Для серверів баз даних ключовими є каталоги `/etc/`, `/var/lib/mysql` або `/var/lib/postgresql`; для вебсерверів — `/var/www`, `/etc/nginx`, `/etc/apache2`, а для робочих станцій — домашні каталоги користувачів, директорії документів, завантажень і робочих файлів. Активований модуль File Integrity Monitoring (FIM) з параметрами `realtime="yes"` та `whodata="yes"` повинен відстежувати всі критичні шляхи, щоб фіксувати не тільки факт зміни файлів, а й користувача, який її виконав, і процес, через який вона відбулася.

Окрім FIM, агенти мають працювати із такими додатковими модулями:

- Rootcheck для виявлення прихованих процесів, змінених бібліотек або руткітів;
- Syscollector для побудови інвентаризаційного профілю кожного вузла (версії ПЗ, відкриті порти, інтерфейси, пакети);
- Vulnerability Detector, який порівнює версії встановлених програм із базами CVE;
- Security Configuration Assessment (SCA), що перевіряє конфігурації відповідно до політик CIS, NIST або ISO.

Налаштування модулів повинно бути централізованим: через шаблони конфігурацій та API-запити адміністратор може оновлювати конфігурації одразу для сотень агентів. Важливо реалізувати стандартизацію профілів — сервери баз даних, вебсервери, робочі станції, хмарні вузли повинні мати різні політики FIM та різну глибину моніторингу.

Окрему увагу слід приділити розробці кореляційних правил, які перетворюють “сирі” події на осмислені інциденти. Кожне правило має включати часовий інтервал, кількість подій і логічну залежність між ними. Наприклад, правило для виявлення шифрувальників може фіксувати ситуацію, коли протягом 300 секунд спостерігається понад 50 змін файлів, а процес, який це ініціював, містить у назві `7z`, `winrar` або `encrypt`. Якщо після таких дій фіксується мережеве з’єднання на зовнішню IP-адресу, система одразу створює алерт високої критичності.

Застосування Active Response є ключовим елементом автоматизації. Wazuh дозволяє не лише повідомляти аналітиків про подію, але й миттєво виконувати контрдії: блокування IP-адреси, завершення процесу, відключення користувача або ізоляцію хосту від мережі. Проте застосування активних дій потребує обережності — автоматизацію варто використовувати лише для подій з високим рівнем достовірності, щоб уникнути хибних блокувань легітимної активності.

Важливим етапом є оптимізація FIM-політик. Не слід моніторити всі файли системи — це призведе до перевантаження і зниження продуктивності. Варто виділяти три категорії:

- Критичні — системні файли, конфігурації, службові скрипти;
- Важливі — документи, конфігурації застосунків;
- Нейтральні — тимчасові файли, кеші, логи, які можна ігнорувати.

Систему потрібно регулярно перевіряти — тестування кореляційних правил, перевірка цілісності бази FIM, емуляція сценаріїв атак. Такі перевірки дозволяють виявляти “сліпі зони” моніторингу. Рекомендується впровадити практику purple teaming — коли команда безпеки і тестувальники моделюють реальні атаки, перевіряючи реакцію системи у реальному часі.

Організаційна частина також відіграє ключову роль. Необхідно забезпечити інтеграцію Wazuh із системами управління інцидентами (наприклад, ServiceNow або Jira) для автоматичного створення тикетів і відстеження статусу розслідування. Ролі користувачів у Wazuh Dashboard мають бути чітко розмежовані: аналітики — лише для перегляду та створення звітів, адміністратори — для керування агентами та політиками. Регулярне навчання персоналу SOC з аналізу FIM-подій та створення правил детекції є обов’язковою частиною процесу безперервного вдосконалення.

Таблиця 3.1.

Рекомендації щодо застосування технології

Категорія	Рекомендація	Очікуваний результат
Архітектура системи	Використання кластерів Manager + Indexer + Dashboard	Забезпечення масштабованості, відмовостійкості та швидкої обробки подій

Категорія	Рекомендація	Очікуваний результат
Конфігурація агентів	Реалізація централізованих шаблонів конфігурацій та політик FIM	Зменшення помилок конфігурації, спрощення адміністрування
Моніторинг FIM	Увімкнення режиму realtime та whodata для критичних файлів	Повна видимість змін і контроль над ініціаторами дій
Правила кореляції	Побудова сценаріїв виявлення складних атак (ransomware, insider)	Виявлення поведінкових аномалій і рання ідентифікація інцидентів
Active Response	Використання автоматичних дій (блокування, ізоляція, відновлення)	Зниження часу реакції на інцидент, мінімізація людського фактора
Інтеграція та звітність	Зв’язок із ITSM, створення дашбордів комплаєнсу (NIST, ISO, PCI DSS)	Підвищення прозорості й відповідності стандартам безпеки
Навчання персоналу	Проведення регулярних симуляцій і аналіз помилкових спрацювань	Безперервне вдосконалення якості аналітики та правил

Узагальнюючи, Wazuh не є просто інструментом моніторингу, а комплексною технологією, що об’єднує елементи SIEM, XDR і поведінкової аналітики. Для досягнення максимальної ефективності система має бути інтегрована у весь цикл реагування: збір → аналіз → кореляція → автоматична дія → перевірка → вдосконалення. Впровадження таких рекомендацій дозволяє створити середовище, де кожна кінцева точка — сервер, робоча станція чи контейнер — перебуває під постійним контролем, а будь-яка спроба

несанкціонованого доступу одразу фіксується, аналізується і блокується з мінімальною затримкою.

ВИСНОВКИ

У межах практичної частини кваліфікаційної роботи було здійснено поетапне впровадження та тестування технології захисту кінцевих точок організації на базі платформи Wazuh. На першому етапі було розроблено прототип системи, який включає ключові компоненти Wazuh — зокрема серверну частину, агенти для кінцевих точок та механізми кореляції подій безпеки. Прототип був адаптований до умов корпоративного середовища з урахуванням типових загроз, що виникають у процесі експлуатації кінцевих пристроїв.

Агенти Wazuh було встановлено на тестових кінцевих точках, що дозволило забезпечити збір логів, моніторинг активності пристроїв та передачу даних до центрального вузла обробки. Наступним кроком стало налаштування правил виявлення загроз, реагування та ізоляції, які базуються на сигнатурному та поведінковому аналізі, а також на інтеграції з базою знань MITRE ATT&CK для точнішої класифікації інцидентів.

Система була протестована на основі типових сценаріїв атак, таких як фішингові повідомлення, запуск шкідливого програмного забезпечення та спроби несанкціонованого доступу до ресурсів. У ході тестування було зафіксовано здатність Wazuh своєчасно виявляти загрози, блокувати шкідливу активність та ізолювати уражені пристрої. Ефективність реагування оцінювалася за ключовими метриками — середнім часом виявлення загрози (MTTD) та середнім часом реагування/відновлення (MTTR). Отримані результати свідчать про здатність системи забезпечити оперативне реагування на інциденти та знизити ризики поширення атаки в межах корпоративної мережі.

Узагальнюючи результати дослідження, можна зробити висновок, що поставлена мета роботи — розробка та впровадження технології захисту кінцевих точок на базі Wazuh — була досягнута. У процесі виконання роботи було вирішено всі наукові завдання: проведено аналіз актуальних загроз, досліджено архітектуру Wazuh, реалізовано практичне впровадження системи захисту, протестовано її

функціональність та сформульовано рекомендації щодо інтеграції в корпоративну IT-інфраструктуру.

Практичне значення роботи полягає в тому, що запропонована технологія може бути адаптована до потреб різних організацій, незалежно від їх масштабу чи галузі діяльності. Вона забезпечує гнучкий, масштабований та ефективний захист кінцевих точок, враховуючи сучасні виклики кіберпростору, зокрема зростання кількості гібридних працівників, мобільних пристроїв та складних атак. Результати дослідження можуть бути використані для подальшого вдосконалення систем кіберзахисту, розробки нових політик безпеки та інтеграції Wazuh з іншими платформами, такими як SOAR, CTI та засоби автоматизації реагування.

Таким чином, кваліфікаційна робота має як теоретичну, так і прикладну цінність, а її результати можуть бути використані в освітньому процесі, наукових дослідженнях та практичній діяльності фахівців з кібербезпеки.

ПЕРЕЛІК ПОСИЛАНЬ

1. ENISA. Threat Landscape 2024. URL: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024 (дата звернення: 03.10.2025).

2. NIST. SP 800-53 Rev.5: Security and Privacy Controls for Information Systems and Organizations. URL: https://csrc.nist.gov/pubs/sp/800/53/r5/final (дата звернення: 03.10.2025).

3. Center for Internet Security (CIS). Critical Security Controls v8 (та v8.1). URL: https://www.cisecurity.org/controls/v8 (дата звернення: 03.10.2025).

4. Wazuh Documentation. File Integrity Monitoring та дотичні матеріали (FIM PoC, HIST-відповідність). URL: https://documentation.wazuh.com/current/user-manual/capabilities/file-integrity/ (дата звернення: 03.10.2025).

5. ENISA. Threat Landscape 2024. URL: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024 (дата звернення: 04.10.2025).

6. Verizon. 2025 Data Breach Investigations Report (DBIR). URL: https://www.verizon.com/business/resources/T163/reports/2025-dbir-data-breach-investigations-report.pdf (дата звернення: 04.10.2025).

7. Mandiant (Google Cloud). M-Trends 2025. URL: https://services.google.com/fh/files/misc/m-trends-2025-en.pdf (дата звернення: 04.10.2025).

8. MITRE ATT&CK Enterprise Matrix. URL: https://attack.mitre.org/matrices/enterprise/ (дата звернення: 04.10.2025).

9. NIST. SP 800-53 Rev.5: Security and Privacy Controls. URL: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final (дата звернення: 04.10.2025).

10. Sandhu R., Ferraiolo D., Kuhn R. Role-Based Access Control Models. IEEE Computer, 1996.

11. NIST. Zero Trust Architecture. Special Publication 800-207. URL: https://csrc.nist.gov/publications/detail/sp/800-207/final (дата звернення: 05.10.2025).

12. CIS. Critical Security Controls v8. URL: https://www.cisecurity.org/controls/cis-controls (дата звернення: 05.10.2025).

13. Tripwire. File Integrity Monitoring: Best Practices. URL: https://www.tripwire.com/solutions/file-integrity-monitoring (дата звернення: 05.10.2025).

14. Qualys. File Integrity Monitoring Datasheet. URL: https://www.qualys.com (дата звернення: 05.10.2025).

15. Wazuh Documentation. *Architecture — Getting started with Wazuh*. URL: https://documentation.wazuh.com/current/getting-started/architecture.html (дата звернення: 06.10.2025).

16. Wazuh Documentation. *File Integrity Monitoring (FIM) — Capabilities / PoC*. URL: https://documentation.wazuh.com/current/user-manual/capabilities/file-integrity/index.html (дата звернення: 06.10.2025).

17. Wazuh Documentation. *Active Response — Capabilities / PCI DSS mapping*. URL: https://documentation.wazuh.com/current/user-manual/capabilities/active-response/index.html (дата звернення: 06.10.2025).

18. Wazuh Documentation. *Log data collection — journald / logcollector*. URL: https://documentation.wazuh.com/current/user-manual/capabilities/log-data-collection/journald.html (дата звернення: 06.10.2025).

19. Wazuh Documentation. *Decoders & Rules — syntax and testing (wazuh-logtest)*. URL: https://documentation.wazuh.com/current/user-manual/ruleset/ruleset-xml-syntax/decoders.html (дата звернення: 06.10.2025).

20. Wazuh Documentation. *Components & Indexer (OpenSearch-based) — clustering & operations*. URL: https://documentation.wazuh.com/current/user-manual/wazuh-indexer/index.html (дата звернення: 06.10.2025).

21. Wazuh Documentation. Ruleset, Decoders, and Correlation Mechanisms. URL: https://documentation.wazuh.com/current/user-manual/ruleset/ (дата звернення: 06.10.2025).

22. Wazuh Documentation. Active Response and SOAR Automation. URL: https://documentation.wazuh.com/current/user-manual/capabilities/active-response.html (дата звернення: 06.10.2025).

23. OpenSearch / Elasticsearch Security Analytics Guide. Event Correlation and Aggregation Techniques. URL: https://opensearch.org/docs/latest/security-analytics/ (дата звернення: 06.10.2025).

24. MITRE ATT&CK Framework. Detection and Correlation for Privilege Escalation and Lateral Movement. URL: https://attack.mitre.org/ (дата звернення: 06.10.2025).

25. SANS Institute. Practical Log Management and Use of Threat Intelligence in SIEM Tuning. URL: https://www.sans.org/ (дата звернення: 06.10.2025).

26. NIST. Computer Security Incident Handling Guide (SP 800-61 Rev. 2). URL: https://csrc.nist.gov/publications/detail/sp/800-61/rev-2/final (дата звернення: 06.10.2025).

27. Red Hat / Linux Audit Documentation. *Configuring Linux Audit and auditd for file access monitoring (syscalls, rules, performance)*. URL: https://access.redhat.com/documentation (дата звернення: 06.10.2025).

28. Microsoft Sysinternals. *Sysmon Documentation — Process, File and Registry Event IDs; Windows Security Object Access Auditing*. URL: https://learn.microsoft.com/sysinternals/downloads/sysmon (дата звернення: 06.10.2025)

29. Samba Project. *vfs_full_audit: Auditing SMB file operations on Samba shares*. URL: https://www.samba.org/samba/docs/ (дата звернення: 06.10.2025).

30. AWS CloudTrail. *Logging Amazon S3 Data Events (GetObject/PutObject/DeleteObject) for detection and investigation*. URL: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-data-events.html (дата звернення: 06.10.2025).

31. CISA. *Ransomware & Threat Actor Tactics* — рекомендації з виявлення та реагування. URL: https://www.cisa.gov (дата звернення: 06.10.2025).

32. SANS Institute. *Practical Log Management / Detection Engineering materials* — поради щодо побудови детекцій і тестування (збірка whitepapers). URL: https://www.sans.org/ (дата звернення: 06.10.2025).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)