План лекції.

Тема 0Р. Опис циклу лабораторних робіт та методики їх виконання.

- Тематика лабораторних робіт та порядок виконання.
- Рекомендації до побудови навчального стенду (лабораторна робота №1)
- Nagios vs Icinga.
- Сучасна архітектура Icinga2.
- Розгортання платформи безпекового моніторингу Wazuh.
- Основи налаштування та експлуатації Wazuh
- Розгортання Splunk на сервері Ubuntu
- Побудова та налаштування конвеєра Wazuh → Splunk

Тематика лабораторних робіт та порядок виконання.

Сьогодні ми розпочинаємо новий практичний курс, присвячений системному та мережевому моніторингу. Більшість із вас уже має базові навички та досвід роботи з моніторингом, адже на попередніх курсах ви вивчали фундаментальні речі: принципи збору метрик, роботу агентів, основи SNMP, а також знайомилися з такими системами як Nagios і Zabbix.

Наш нинішній курс — це логічне продовження та розширення попередньої підготовки. Тепер ми переходимо на більш професійний рівень практики, орієнтований на сучасні інструменти моніторингу, які широко застосовуються у бізнесі, DevOps-підходах та інфраструктурній безпеці.

Ціль наших лабораторних робіт — не просто повторити теорію, а побудувати робочі системи моніторингу, навчитися налаштовувати їх відповідно до реальних сценаріїв і зрозуміти, як такі інструменти застосовуються в сучасній IT-інфраструктурі. У цьому курсі ми торкнемося двох ключових напрямів:

- Системний моніторинг — спостереження за сервісами, хостами, ресурсами, логами та процесами.
- Моніторинг безпеки — виявлення загроз, аналіз подій безпеки, обробка логів, правил і кореляцій.

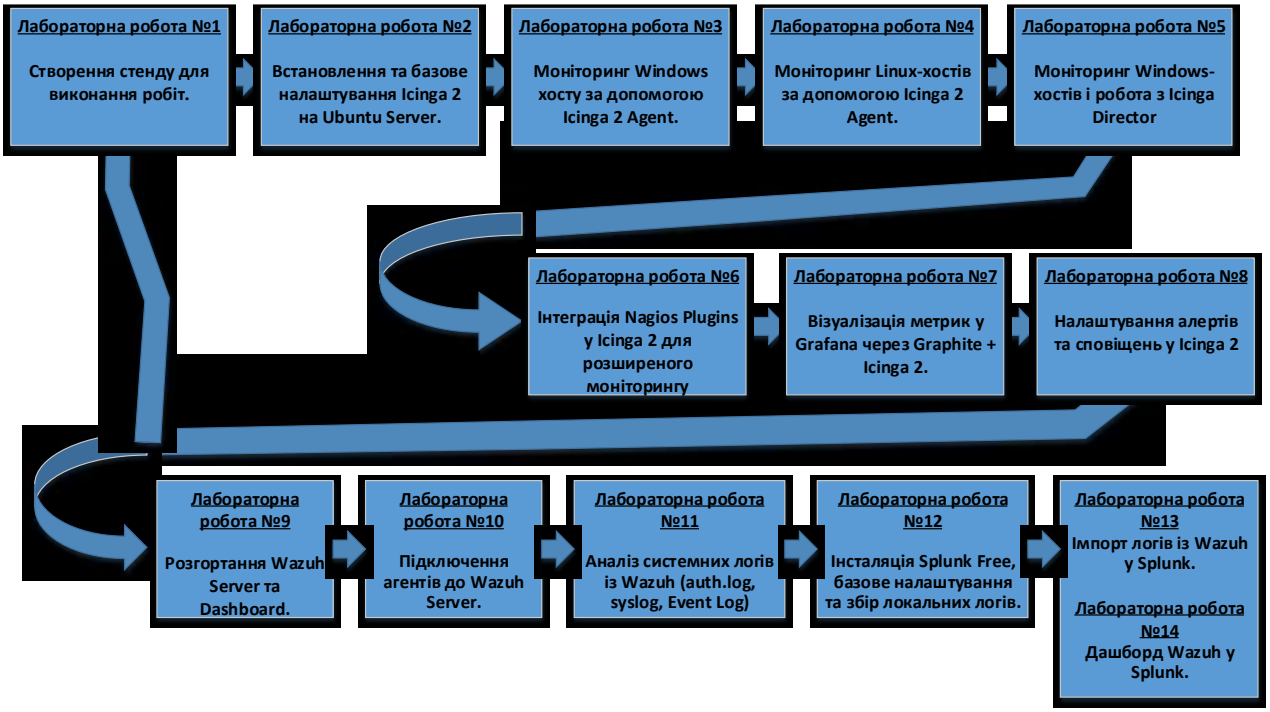

Рис. Р.1. Лабораторні роботи курсу та порядок їх виконання

Стрілочками показаний можливий порядок виконання робіт. Лабораторні завдання поділені на дві великі частини.

Частина 1 – системний моніторинг, де вивчається Icinga + Grafana та дотичні до них речі. Це лабораторні роботи №2-8

Частина 2 – безпековий моніторинг, де працюємо з Wazuh та Splunk і їх взаємна інтеграція та доповнення. Це лабораторні роботи №9-13

Зоозуміло, що для початку роботи необхідно розгорнути навчальні об’єкти моніторингу. Цьому присвячена обов’язкова робота №1

Таким чином, прослідкувавши за напрямком стрілочок на рис.Р.1 робимо висновок, що є три основних можливих стратегії виконання курсу лабораторних робіт. Перша – всі роботи підряд з першої до останньої. Друга – проходження гілки системного моніторингу. При такому виконанні ми зупиняємось на 8 лабораторній роботі. Третя стратегія – налаштовуємо стенд-мережу у першій роботі та одразу переходимо до виконання роботи №9 і рухаємось до завершення курсу.

Звичайно є безліч варіацій цих трьох стратегій, бо вимоги до кількості виконаних робіт у цьому курсі не ставиться.

Структура всіх методичних посібників однакова: теоретична частина, де приведено опис дій, завдання, та інформаційні джерела. Після завдань можуть бути наведені додатки.

Рекомендації до побудови навчального стенду (лабораторна робота №1)

Стенд будується у VirtualBox версії не нижче 7.10.0. У якості мережі використовується віртуальна мережева інфраструктура типу NAT Network, яка забезпечує взаємодію між віртуальними машинами в межах єдиної адресної підмережі. Спочатку визначається схема адресації відповідно до умов варіанту. Всі умови приведені у завданнях.

Під час розподілу IP-адрес враховується резервування перших двох адрес діапазону для службових потреб або потенційного розширення. Адреса шлюза NAT-мережі фіксується окремо (зазвичай перша доступна адреса підмережі). Статичні IP-адреси для серверів обираються з вільного діапазону, починаючи з третьої адреси, а робоча станція отримує адресу динамічно через DHCP-сервер, який розгортається на одному з серверів.

Для мережі вмикаємо DHCP-сервер, що забезпечить автоматичну адресацію на початковому етапі. Далі виконуємо імпорт трьох віртуальних машин з аплайнсів, що доступні по лінку https://drive.google.com/drive/folders/1kfdx0bx93UAzs12PxTZHiVPWEduzwN2L

Serv-G-N-1 – сервер на базі Windows Server 2022,
Serv-G-N-3 – сервер на базі Ubuntu 24.04 LTS,
WS-G-N-1 – робоча станція з Windows 10.

Результатом цих дій має стати мережа, схема якої наведена на рис.Р.2. Приклад розподілу адрес наведений у таблиці Р.1.

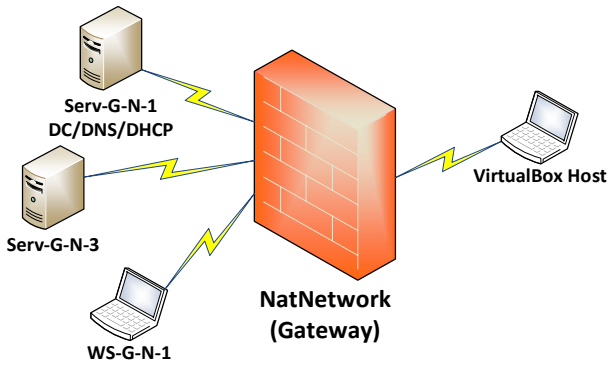

Рис. Р.2. Топологія мережі

Таблиця Р.1

Мережа / Пристрій	**Інтерфейс / Мережний адаптер / Шлюз**	**IP-адреса**	**Маска**
Мережа		192.168.50.0/27	255.255.255.224
Зарезервовані адреси VBox		192.168.50.1, 192.168.50.2	255.255.255.224
Windows сервер Serv-22-50-1	Мережний адаптер	192.168.50.3	255.255.255.224
	Шлюз за замовчуванням/ Public DNS	192.168.50.1	-
Ubuntu 24.04 or later server Serv-22-50-3	Мережний адаптер	192.168.50.5	255.255.255.224
	Шлюз за замовчуванням/ Public DNS	192.168.50.1	-
Робоча станція Windows 10 WS-22-50-1	Мережний адаптер	DHCP	255.255.255.224
	Шлюз за замовчуванням/ Public DNS	192.168.50.1	-

Кілька слів про аплайнси віртуальних машин, що використовуються у цьому курсі.

Serv-G-N-1.ova побудований на базі Windows Server 2022 Standard. Фактично це базовий шаблон Windows Server 2022 Standard (Desktop Experience) у форматі .OVA, створений для навчальних цілей або розгортання стендів. У шаблоні не встановлено ролі чи служби. Неактивована система (GVLK встановлено, але slmgr /ato не виконувався) Даний шаблон придатний для довготривалого зберігання та багаторазового використання. Для ліцензування встановлено офіційний GVLK для Windows Server 2022 Standard

VDYBN-27WPP-V4HQT-9VMD4-VMK7H

Активація не виконана - відлік 180-денного KMS-терміну не почався. Ви можете імпортувати шаблон коли завгодно, і активувати пізніше. Після виконання slmgr /ato активація триватиме 180 днів

Serv-G-N-3.ova — Ubuntu Server 24.04.2 LTS. Мінімалістичний шаблон віртуальної машини на базі Ubuntu Server 24.04.2 LTS (Jammy Jellyfish). Містить мінімалістичну інсталяцію системи з LVM-розбиттям диска. Логін локального адміністратора Student, пароль – 111111.

Чому є у шаблоні Serv-G-N-3 використовується LVM замість класичного статичного розбиття диска. LVM (Logical Volume Manager) — це рівень абстракції між фізичними дисками та файловими системами, який дозволяє гнучко керувати дисковим простором уже після встановлення операційної системи. Основні переваги LVM:

- Гнучкість керування дисковим простором
  Розміри розділів не є фіксованими. Їх можна збільшувати без перевстановлення системи.
- Масштабованість
  До існуючої системи можна додавати нові віртуальні диски або розширювати наявні.
- Безперервність роботи
  У більшості випадків розширення логічних томів виконується без перезавантаження.

Під час виконання лабораторних робіт з Icinga 2, баз даних, Redis, логування та збереження метрик можливі ситуації, коли швидко зростає обсяг логів, або накопичуються історичні дані моніторингу чи зберігаються дампи або резервні копії. Результатом цих явищ буде нестача місця у /var або кореневому розділі. У таких випадках застосовується методика розширення дискового простору за допомогою LVM, яка є стандартною процедурою в адмініструванні Linux-серверів.

WS-G-N-1.ova побудовано на базі Windows 10 Professional. Базовий шаблон Windows 10 Professional (x64) у форматі .OVA. Логін локального адміністратора Student, пароль - 111111

Таким чином виконавець лабораторних робіт отримує у своє розпорядження аплайнси трьох різних операційних систем. Системні вимоги до віртуальних машин, що створюються на базі цих шаблонів описані у відповідних методичних рекомендаціях.

По лінку https://drive.google.com/drive/folders/1kfdx0bx93UAzs12PxTZHiVPWEduzwN2L де розміщені описані аплайнси також доступні інсталяційні образи Windows Server 2022 Standard та Ubuntu Server 24.04.2 LTS.

Nagios vs Icinga

Перш ніж перейти до рекомендацій щодо виконання наступних лабораторних робіт, хочу розповісти одну цікаву історію — вона допоможе краще зрозуміти, чому сьогодні ми звертаємо увагу не лише на Nagios, а й на його головну альтернативу, Icinga 2. У статті, лінк на яку я наведу у конспекті, цю тему розглянуто докладніше, але зараз — коротко про головне.

Історія починається з Nagios — перша версія була випущена 14 березня 1999 року. Автор Nagios, Ітан Галстад (Ethan Galstad), спочатку створив систему як хобі — він вважав, що наявні на той момент рішення для моніторингу були надто негнучкі. Nagios задумувався як гнучке рішення, яке надалі можна буде використовувати і комерційно.

Архітектура Nagios передбачала ядро + плагіни: сама “логіка” в ядрі, а перевірки, доповнення й розширення — як окремі плагіни. Це давало гнучкість при розширенні функціоналу.

Чому ж саме виник форк — Icinga? Через 10 років після появи Nagios, в травні 2009-го, група розробників оголосила про створення Icinga — форку Nagios. Головною причиною розколу стало незадоволення станом розвитку Nagios: деякі учасники спільноти вважали, що проект “застряг”, й хотіли більш відкритого, активного та спільнотного шляху розвитку. При цьому, хоча Icinga “відокремився”, частина розробників продовжували надсилати патчі для Nagios — тобто розрив не був категоричним, але новий проєкт мав інші цілі.

Icinga запровадив новий підхід до розвитку: від ідеї “окреме ядро + плагіни” намагалися перейти до більш активної участі спільноти, відкрітого розвитку та швидшої еволюції. Крім технічних функціональних переваг, Icinga прагнув бути “спільнотним” — не лише набором патчів, але і живим проєктом, яким могли б опікуватися багато людей.

Після появи форку Icinga розвивалася дуже швидко: вже в перший рік з’явилися окремі релізи ядра, API і веб-інтерфейсу. Згодом Icinga перетворився на повноцінну систему моніторингу — з підтримкою баз даних, розподілених інфраструктур, новим UI, гнучкішими можливостями, які переглядають і доповнюють архітектуру класичного Nagios. Але й Nagios не “завмер”: розробник визнає, що існували “затримки” в розвитку, але сам факт форку засловами Ітана Галстеда (Ethan Galstad) став сигналом для модернізації. Він припускає, що це могло стимулювати новий поштовх для розвитку Nagios.

Форк Icinga — це приклад, як у світі open-source конфлікт інтересів чи незадоволення розвитком може породити новий, конкуретний проєкт, який — за умови активної спільноти — може еволюціонувати. Icinga демонструє, що базова архітектура (як у Nagios) може бути збережена, але додана сучасна функціональність, гнучкість і підхід до розвитку — що робить його привабливим варіантом для тих, хто хоче більш сучасного, масштабованого та дружнього до спільноти рішення. З іншого боку — історія показує: навіть проєкти з багаторічним досвідом і великою базою користувачів можуть вимагати перезапуску чи оновлення; і іноді “форк” — це не кидок, а шанс для розвитку.

Рис. Р.3. Автор Nagios, Ітан Галстад (Ethan Galstad)

Рис. Р.4.Один з засновників команди Icinga та розробник Nagios-plugins Майкл Люббен (Michael Lübben)

Порівняння Nagios з сучасною архітектурою Icinga2

Перш ніж перейти до практичної частини курсу, нам важливо зрозуміти, чому в сучасних інфраструктурах більшість організацій відходять від класичного **Nagios Core** та переходять на більш гнучкі системи, зокрема **Icinga 2**. Логічним кроком є розібратися у відмінностях між цими продуктами. Щоб оцінити сильні та слабкі сторони, ми будемо порівнювати саме **некомерційні, відкриті версії** Nagios Core та Icinga 2 — тобто ті продукти, які доступні вільно й які зазвичай використовують для навчання, досліджень або в невеликих інфраструктурах.

Icinga 2 історично виникла як форк Nagios, але за останні роки переросла свого “пращура” та значно розширила можливості, зберігаючи сумісність із плагінами. Саме тому нам важливо побачити, **що змінилося, які проблеми Nagios були перенесені або вирішені**, і чому сучасний моніторинг дедалі частіше вибирає Icinga 2.

1. Архітектура та продуктивність

Nagios Core побудовано на класичній моделі: центральний сервер, текстові конфігураційні файли, перевірки через плагіни, лінійний цикл опитування. Через це система погано масштабується та часто вимагає складних оптимізацій.	Icinga 2 створена з урахуванням сучасних вимог — це повністю перероблене ядро зі швидким механізмом обробки подій та можливістю горизонтального масштабування. Воно підтримує розподілені топології “мастер–сателіт–агент”, що дозволяє легко моніторити великі географічно розподілені мережі.

Icinga 2 набагато швидша, гнучкіша та легше масштабується.

2. Конфігурація та гнучкість

Nagios Core використовує простий, але обмежений синтаксис конфігів, заснований на декларативних об'єктах. Структура зрозуміла, проте погано адаптується до складних сценаріїв.	Icinga 2 має власну DSL-мову для конфігурацій — потужну, об'єктно-орієнтовану та модульну. Вона дозволяє використовувати змінні, наслідування, шаблони, логічні умови, групи, залежності — те, чого Nagios робити не вміє або робить через костилі.

У плані конфігурацій Icinga 2 — значно сучасніша та зручніша.

3. Управління через API

Nagios Core у своїй базовій версії дійсно не містить сучасного вбудованого REST API. Проте завдяки великій екосистемі та довгій історії проєкту існує цілий набір сторонніх рішень, які частково компенсують цю прогалину. ❖ NagiosQL — веб-інструмент для централізованого керування конфігураціями; ❖ інші модулі та API-подібні надбудови, що дозволяють керувати об'єктами, статусами та імпортувати конфігурації. Хоча це розширює функціональність Nagios, але слід зазначити, що такі рішення: ❖ не є частиною ядра, ❖ не охоплюють усю можливу функціональність, ❖ часто мають окремі цикли розробки та підтримки.	Icinga 2, на відміну від цього, має повноцінний REST API прямо з коробки, інтегрований у саму архітектуру системи. API охоплює весь спектр завдань і дозволяє: ❖ створювати/видаляти об'єкти, ❖ керувати хостами, ❖ отримувати статуси та метрики, ❖ інтегрувати інші системи автоматично.

Nagios Core можна доповнити API через сторонні продукти, але Icinga 2 має сучасний, потужний та рідний REST API, що робить її зручнішою для автоматизації та інтеграцій.

4. Моделі перевірок та агенти

У Nagios Core традиційно застосовується pull-модель, де сервер опитує хости через плагіни. Проте екосистема Nagios включає багато клієнтів та агентських рішень (NRPE, NCPA, NSClient++, NRDP та інші), що дозволяють: ❖ виконувати перевірки локально на вузлі, ❖ встановлювати захищене з'єднання (SSL/TLS), ❖ використовувати сертифікати, ❖ передавати дані в обидва боки. Варто підкреслити: можливості Nagios у агентській моделі не такі вже й обмежені, хоча вони реалізовані через зовнішні компоненти, які не завжди однаково підтримуються та мають різну сумісність.	Icinga 2 же має вбудовану нативну агентську інфраструктуру, інтегровану в ядро системи. Вона: ❖ використовує єдину модель сертифікації та шифрування, ❖ нативно працює у розподіленій архітектурі (майстер–сателіт–агент), ❖ легко масштабується, ❖ не потребує зовнішніх надбудов для базових агентських функцій.

Nagios Core підтримує безпечні агенти через зовнішні клієнти, але Icinga 2 пропонує єдиний, стандартизований та сучасний агентський механізм, інтегрований у саму систему.

5. Веб-інтерфейс та візуалізація

Базовий веб-інтерфейс Nagios Core мінімалістичний і застарілий, але існує широка екосистема готових рішень, що забезпечують сучасну візуалізацію: ❖ Thruk ❖ PNP4Nagios ❖ Grafana (через InfluxDB/Graphite/pnp4nagios datasource) ❖ MRTG Великий вибір сторонніх UI дозволяє довести Nagios до візуального рівня сучасних систем.	Icinga Web 2 — сучасна, модульна й інтерактивна панель із коробки. Менше залежить від сторонніх UI, ніж Nagios. Проте також інтегрується з Grafana без будь-яких проблем.

Nagios має слабкий базовий веб-інтерфейс, але багату екосистему для розширення. Icinga 2 має сильний веб-інтерфейс «із коробки», менше потребує зовнішніх інструментів.

6. Сумісність із плагінами

Обидві системи використовують сумісні плагіни, бо Icinga 2 зберегла підтримку стандарту Nagios Plugins.

7. Логи та аналітика

Nagios Core працює з простим текстовим логуванням без структурованого аналізу.	Icinga 2 підтримує структуровану модель логів, передачу в ELK або інші системи, перегляд через веб-інтерфейс.

Icinga 2 краще інтегрується з сучасними аналітичними системами.

8. Підтримка агентів Nagios в Icinga 2

Агент Icinga 2 (icinga2-agent) не можна напряму використовувати в Nagios, оскільки він використовує власний протокол шифрування і конфігураційної синхронізації, несумісний із Nagios Core.	Icinga 2 повністю сумісна з Nagios-плагінами (NRPE, check_*), оскільки використовує ту саму архітектурну модель плагінів. Багато середовищ після міграції на Icinga продовжують використовувати існуючі Nagios-агенти. Це робить перехід з Nagios на Icinga легким і безболісним.

Але Icinga 2 може виконувати Nagios-плагіни, тому вектор сумісності односторонній:
Nagios → Icinga: так

Icinga → Nagios: ні

9. Порівняльна таблиця Nagios Core vs Icinga 2

Характеристика	Nagios Core	Icinga 2
Ліцензія	Open-source (GPL)	Open-source (GPL)
Архітектура	Монолітна, одноканальна	Розподілена, агентська, масштабована
API	Відсутній «із коробки», але є сторонні рішення (NagiosQL, NDOUtils, NRDP)	Потужний REST API «із коробки»
Плагіни	Широка екосистема	Повністю сумісна з Nagios-плагінами
Агенти	NRPE, NSClient++, багато сторонніх	Власний агент + сумісність з NRPE/NSClient++
Веб-інтерфейс	Базовий, але багато сторонніх UI (Thruk, Centreon, Grafana, PNP4Nagios)	Сучасний Icinga Web 2 «із коробки»
Візуалізація	Через PNP4Nagios, MRTG, Grafana тощо	Через Icinga Web 2, Grafana
Масштабування	Можливе, але складніше	Закладено в архітектуру
Конфігурація	Текстові файли, багато сторонніх конфігураторів	DSL + конфігурація через API та Director

Сучасна архітектура Icinga2

Після того як ми порівняли Nagios Core та Icinga 2 і побачили їхні архітектурні, конфігураційні й функціональні відмінності, логічним кроком є розгляд актуальної конфігурації Icinga 2 та її функціональних компонентів. Сучасна система Icinga 2 — не просто моніторинг-сервер із перевірками. Це модульна платформа, що складається з низки окремих функцій (features), кожна з яких реалізовує певний функціональний блок системи: від журналювання й API до інтеграцій з базами даних чи системами метрик. Саме така модульна структура дозволяє гнучко налаштовувати моніторинг відповідно до потреб інфраструктури. Розглянемо найважливіші компоненти, які формують актуальну конфігурацію Icinga 2, і пояснимо яку роль і яку функцію виконує кожна з частин. Icinga

- **Logging — журналювання**

Ця частина відповідає за запис подій, станів і внутрішніх повідомлень системи. Icinga 2 підтримує кілька типів логів:

- mainlog — основний журнал роботи системи,
- debuglog — детальна інформація для налагодження,
- syslog/journald — інтеграція з системними журналами ОС,
- windowseventlog — для Windows-середовища.

Можна вмикати або вимикати різні логгери через команду icinga2 feature enable/disable. Icinga. Це важливо для моніторингу стану самого Icinga 2, діагностики проблем і аудиту виконання перевірок.

- **Core Backends — основні бекенди**

Це набір компонентів, що забезпечують внутрішню інфраструктуру даних і API-сервіс:

- REST API. Це сучасний API, який працює «із коробки» і дозволяє отримувати статуси об'єктів, виконувати команди, керувати конфігураціями та об'єктами моніторингу та автоматизувати задачі з зовнішніх систем. API є одним із ключових механізмів інтеграції з DevOps-інструментами та автоматизацією. Icinga
- Icinga DB. Цей бекенд забезпечує публікацію, зберігання і синхронізацію даних між Redis, базою даних і фронтендом (через Icinga Web або інші інструменти) поточні стани хостів і сервісів, результати перевірок, події та зміни станів, списки запланованих робіт та їх результатів. Icinga DB дозволяє створити «живу» картину стану інфраструктури, синхронізовану між усіма компонентами. Icinga.

  Якщо з базами даних Ви всі знайомі, то про таку річ, як Redis (від англ. remote dictionary server), можливо не чули. Це резидентна система управління базами даних, що працює зі структурами даних типу "ключ - значення". Використовується як для баз даних , так і для реалізації кешів, брокерів повідомлень. Спочатку випускалася під вільною ліцензією , з 2024 року - під ліцензією SSPL (Server Side Public License). Орієнтова на досягнення максимальної продуктивності на атомарних операціях (заявляється приблизно 100 тис. SET- і GET -запитів на секунду на Linux -сервері початкового рівня).

  У даній конфігурації Redis використовується як високопродуктивний транспортний шар (message broker), який передає внутрішні дані моніторингу від Icinga 2 до Icinga DB Writer, а звідти — у реляційну базу даних.

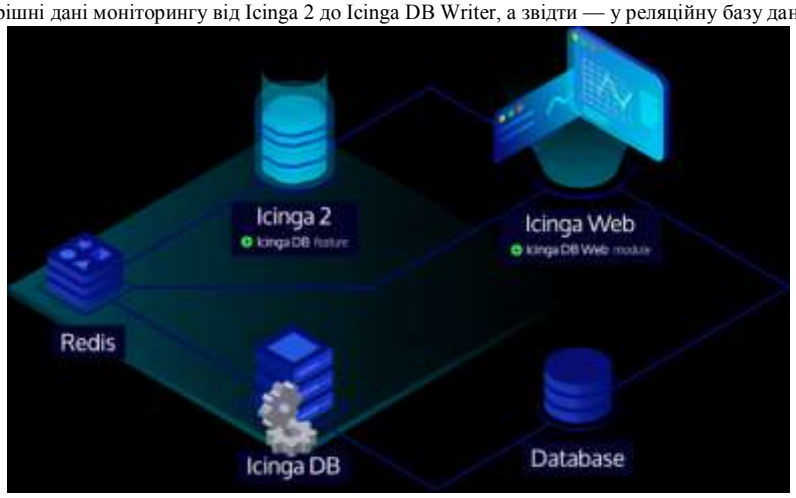

Рис. Р.5. Структура Icinga DB

- **Metrics — робота з метриками**

Ця частина обробляє перформанс-дані, що повертають плагіни перевірок (CPU, Load, пам'ять, затримки тощо), та передає їх у зовнішні системи зберігання. У межах лабораторних робіт підсистема метрик використовується в обмеженій конфігурації. Ми розглядаємо Grafana як інструмент візуалізації, не заглиблюючись у налаштування зовнішніх time-series баз (InfluxDB, Graphite), які підтримуються Icinga 2, але не входять до базового лабораторного стенду.

- **External Integrations — інтеграція з іншими системами**

Ці функції дозволяють передавати події й результати перевірок у зовнішні сервіси логування та аналітики:

- Elastic Stack Integration. Інтеграція з Elasticsearch/Logstash (через логгер або Beats), що дозволяє централізовано шукати та аналізувати події моніторингу разом із іншими логами інфраструктури. Icinga
- Graylog Integration. Підтримка GELF-протоколу для передачі подій у Graylog. Icinga

Для спрощення функціоналу та зменшення навантаження на наше «залізо» інтеграція не використовується в лабораторних роботах

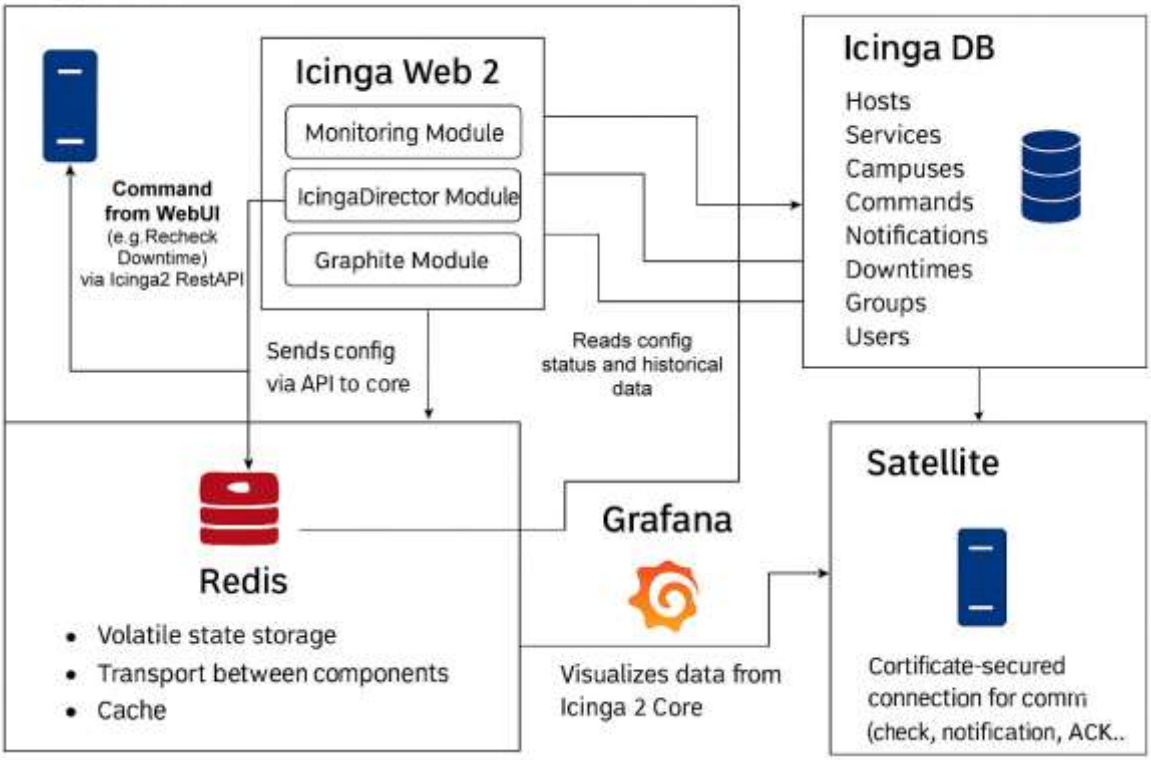

Рис. Р.6. Актуальну архітектуру системи моніторингу на базі Icinga 2

Наведена на рис. Р.6 схема відображає актуальну архітектуру системи моніторингу на базі Icinga 2, яка буде використовуватися в межах лабораторних робіт курсу. Вона демонструє логічний поділ компонентів системи, напрямки потоків керування та даних, а також роль кожного елемента у забезпеченні повноцінного моніторингу інфраструктури. Архітектура побудована за модульним принципом, що дозволяє масштабувати систему, адаптувати її до різних сценаріїв використання та чітко розмежовувати функціональні обов'язки між компонентами.

Рівень керування та взаємодії з користувачем: Icinga Web 2

Центральною точкою взаємодії адміністратора з системою є Icinga Web 2. Саме цей компонент забезпечує веб-інтерфейс для перегляду станів, керування конфігураціями та виконання оперативних дій. У складі Icinga Web 2 використовуються такі ключові модулі:

- ✓ Monitoring Module — відповідає за відображення поточного стану хостів і сервісів, історії подій, повідомлень та аварій.
- ✓ Icinga Director Module — забезпечує централізоване керування конфігурацією: опис хостів, сервісів, шаблонів, груп, правил застосування та об'єктів моніторингу.
- ✓ Graphite Module — інтегрує метрики продуктивності та забезпечує їхню візуалізацію у вигляді графіків.

Важливо підкреслити, що Icinga Web 2 не виконує перевірок самостійно. Усі керуючі дії, ініційовані з веб-інтерфейсу (наприклад, примусовий повтор перевірки, встановлення downtime або підтвердження інциденту), передаються до ядра системи через REST API Icinga 2.

Ядро моніторингу: Icinga 2 Core

У наведеній схемі логічно Icinga 2 Core розташований у межах Master-вузла, на якому також розміщені інші ключові компоненти системи.:

Master node
├── Icinga 2 Core ← (виконує перевірки, логіку моніторингу)
├── Icinga Web 2
├── Redis
├── Icinga DB
└── Grafana

Тобто Core — це не окремий сервер і не окремий сервіс UI, а фоновий демон, який працює на тому ж вузлі, що й інші компоненти Master-інстансу

Icinga 2 Core є центральним елементом усієї системи моніторингу. Саме тут виконуються перевірки хостів і сервісів, обробляються результати перевірок, приймаються рішення щодо зміни станів, формуються сповіщення та події.

Ядро отримує конфігурацію як з локальних файлів, так і через API (зокрема від модуля Director), що дозволяє реалізувати сучасну модель керування конфігураціями без ручного редагування файлів.

Рівень зберігання даних: Icinga DB

Для збереження та обробки станів, історичних даних і конфігурацій використовується Icinga DB. Цей компонент виступає проміжною ланкою між Icinga 2 Core та веб-інтерфейсом. Рівень детально наводився на рис. Р.5. У базі даних зберігається інформація про хости та сервіси, команди перевірок, нотифікації та downtime, користувачів і групи, поточні стани та історичні дані.

Icinga Web 2 звертається саме до Icinga DB для отримання актуальної інформації, що забезпечує відокремлення веб-рівня від безпосередньої роботи ядра та підвищує стабільність системи.

Транспортний і кешуючий рівень: Redis

Redis у цій архітектурі використовується як високопродуктивний транспортний шар між компонентами системи. Опис рівня давався раніше у цій лекції.

Візуалізація метрик: Grafana

Для візуалізації даних продуктивності в архітектурі використовується Grafana. Вона забезпечує побудову гнучких дашбордів, графіків та аналітичних панелей на основі даних, зібраних системою моніторингу. У рамках лабораторних робіт Grafana використовується як основний інструмент візуального аналізу метрик.

Розподілений моніторинг: Satellite

На схемі показано наявність Satellite-вузлів як елемент повної архітектури Icinga 2. У межах лабораторних робіт цей компонент не розгортається, однак його роль розглядається концептуально для розуміння принципів розподіленого моніторингу.

Ми розглянули структуру, призначення та взаємодію основних компонентів Icinga 2, що формують сучасну архітектуру системного моніторингу. Такий огляд є необхідним, оскільки Icinga 2 — це не просто система виконання перевірок, а комплексна модульна платформа, яка об'єднує механізми журналювання, API-доступ, передачу подій, обробку метрик, взаємодію з базами даних і спеціалізовані транспортні шари.

Ми звернули увагу на те, що кожна функціональна частина Icinga 2 виконує чітко визначену роль у загальному процесі моніторингу:

- механізми журналювання забезпечують прозорість роботи та діагностику системи;
- REST API відкриває можливості автоматизації й інтеграції з іншими сервісами;
- Icinga DB та SQL-сервер відповідають за довготривале зберігання даних і історію подій;
- Redis використовується як високопродуктивний транспортний шар для передачі даних між компонентами системи, що дозволяє досягати стабільності, масштабованості та високої швидкодії;
- підсистема метрик інтегрує Icinga з InfluxDB, Graphite та іншими time-series базами для побудови сучасних дашбордів у Grafana;
- механізми зовнішніх інтеграцій забезпечують передачу подій у системи логування та аналітики.

Такий модульний підхід дозволяє гнучко керувати конфігурацією: кожна функція може бути увімкнена або вимкнена залежно від завдань конкретної інсталяції.

Усі ці компоненти, які ми розглянули, будуть послідовно задіяні в процесі розгортання системи моніторингу, що виконуватиметься на сервері Ubuntu, підготовленому з лабораторного аплайнсу. Під час виконання інсталяції ви побачите, як ці функції вмикаються, налаштовуються та взаємодіють між собою в реальному середовищі.

Детальний алгоритм розгортання, налаштування та перевірки роботи кожного з компонентів наведено у відповідних методичних рекомендаціях до лабораторних робіт. У цих документах надано покрокові інструкції, приклади конфігурацій, пояснення структури файлів та рекомендації щодо усунення можливих помилок. Метою цієї лекції є саме систематизоване розуміння того, з яких елементів складається Icinga 2 та чому кожен із них є важливим у комплексному рішенні моніторингу.

Розгортання платформи безпекового моніторингу Wazuh

Після завершення вивчення системного та мережевого моніторингу на базі Icinga 2 ми переходимо до наступного логічного етапу курсу — безпекового моніторингу. Його метою є не лише спостереження за станом систем, а аналіз подій безпеки, журналів, змін у конфігураціях та ознак компрометації. Для цього у межах курсу використовується платформа Wazuh, яка поєднує функції HIDS, SIEM та Security Analytics.

Платформа Wazuh виникла як еволюційна відповідь на практичні проблеми централізованого аналізу подій безпеки в сучасних інфраструктурах. Її історія тісно пов'язана як із розвитком класичних HIDS-систем, так і з популяризацією стеку ELK (Elasticsearch, Logstash, Kibana). Історично Wazuh виріс із проєкту OSSEC (Open Source Host-based Intrusion Detection System) — однієї з перших відкритих HIDS-систем, розробленої на початку 2000-х років. OSSEC забезпечував збір логів, контроль цілісності файлів та базову кореляцію подій, однак мав обмежені можливості масштабування, зберігання історичних даних і візуалізації.

З появою та швидким розвитком ELK-стеку стало очевидно, що поєднання класичного HIDS із сучасною платформою індексації та пошуку подій може суттєво розширити аналітичні можливості системи безпеки. Саме в цьому контексті у 2015 році з'явився проєкт Wazuh — спочатку як форк OSSEC, орієнтований на глибоку інтеграцію з Elasticsearch і Kibana.

На початкових етапах Wazuh фактично використовував ELK як сховище подій безпеки, механізм швидкого пошуку та платформа візуалізації та аналітики. З часом Wazuh переріс модель «OSSEC + ELK» і став самостійною платформою безпекового моніторингу, що включає власний менеджер подій і правил кореляції, розширену агентну модель, контроль цілісності файлів та інтеграції з зовнішніми сервісами загроз.

У подальшому, з урахуванням змін ліцензійної політики Elasticsearch, Wazuh адаптував свою архітектуру до використання OpenSearch, зберігши при цьому концепцію індексованого сховища подій та потужної аналітики.

Таким чином, Wazuh можна розглядати як приклад еволюції класичної HIDS-системи в сучасну SIEM/XDR-платформу, де поєднані агентний збір подій, централізована кореляція, масштабоване сховище і аналітика та візуалізація.

Саме ця еволюція робить Wazuh показовим прикладом сучасного підходу до безпекового моніторингу, особливо у навчальному контексті.

У межах цього курсу ми свідомо обираємо аплайнс-підхід до розгортання серверів. Щоб спростити інсталяцію та зосередитися на принципах роботи, архітектурі та логіці аналізу подій, використовується готовий Wazuh Appliance, який постачається розробниками у вигляді попередньо зібраного програмно-апаратного комплексу. У даному випадку аплайнс реалізований у форматі віртуальної машини, однак концептуально він являє собою завершене рішення "з коробки", що вже містить основні серверні компоненти платформи (рис. Р.7):

- **Wazuh Manager** — центральний компонент, що приймає події від агентів, виконує кореляцію та аналіз;
- **Wazuh API** — сервіс керування та взаємодії з платформою;
- **Wazuh Indexer (OpenSearch)** — підсистема зберігання та індексації подій;

- **Filebeat** — компонент транспорту та доставки подій до індексатора;
- **Wazuh Dashboard** — веб-інтерфейс для аналізу подій, візуалізації та керування.

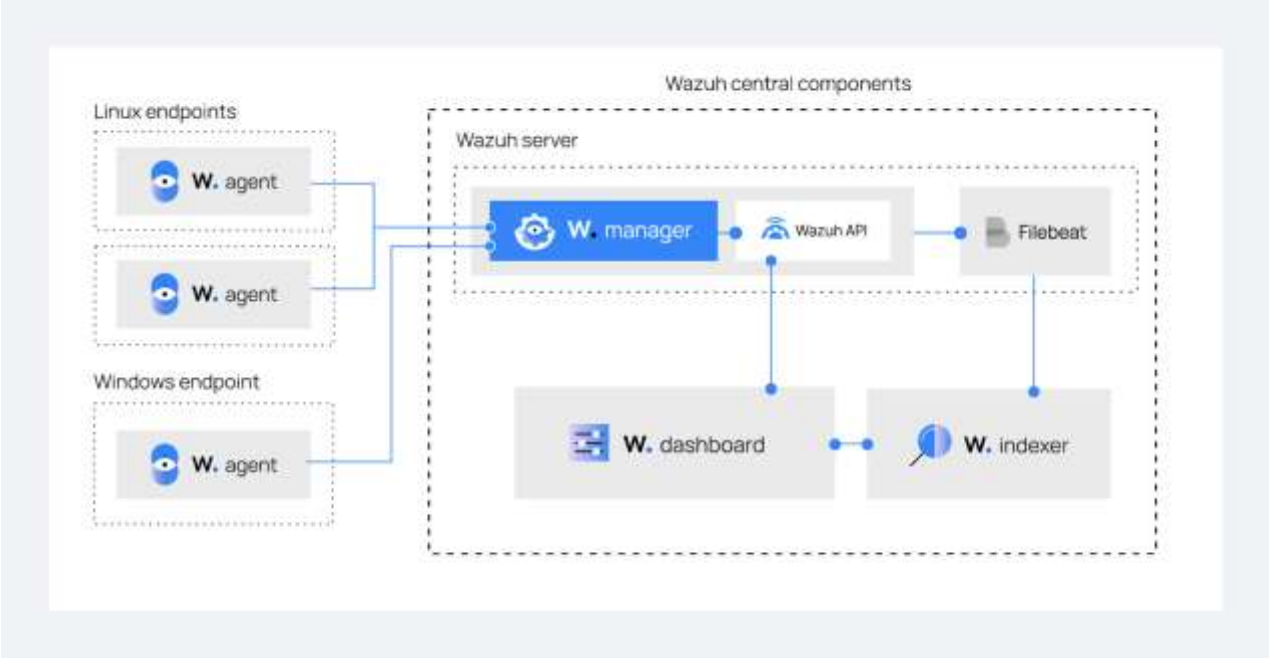

Рис. Р.7. Мінімальна архітектура Wazuh, що використовується у appliance

Таким чином, Appliance реалізує повноцінну серверну частину Wazuh "з коробки", без необхідності окремо встановлювати базу даних, пошуковий рушій або веб-інтерфейс. Водночас важливо розуміти, що у цьому середовищі відсутні клієнтські агенти, не налаштовані зовнішні інтеграції (Splunk, SIEM, Syslog тощо), не виконано адаптацію мережевих параметрів під конкретний стенд.

Після імпорту віртуальної машини їй надається місце у вже існуючому стенді, де вона логічно замінює сервер системного моніторингу Icinga 2, але вже в контексті безпекового моніторингу.

Для оптимізації використання ресурсів попередні ВМ, що більше не використовуються на цьому етапі курсу, вимикаються.

На відміну від попередніх лабораторних робіт, де використовувалась Ubuntu Server, Wazuh Appliance побудований на Amazon Linux 2023. Це дистрибутив, що належить до Red Hat-сімейства і має низку особливостей, а саме – використання менеджера пакетів dnf, іншу структуру системних сервісів та застосування механізму cloud-init для початкової ініціалізації системи.

Після першого запуску Wazuh Appliance сервер автоматично отримує IP-адресу через DHCP. Проте для серверних систем, особливо у сфері моніторингу та безпеки, динамічна адресація є небажаною. Тому в межах лабораторної роботи виконується відключення мережевої частини cloud-init, передача повного контролю над мережею системі systemd-networkd та налаштування статичної IP-адреси, що забезпечує стабільність доступу до сервера.

Після коректного налаштування мережі виконується перевірка стану основних компонентів Wazuh. Незважаючи на те, що Appliance є готовим середовищем, деякі сервіси можуть бути не активовані автоматично. У рамках роботи перевіряється та забезпечується коректна робота основних складових, про які мивже згадували: Wazuh Manager, Indexer та Dashboard.

Після запуску сервісів перевіряється, що сервер слухає відповідні мережеві порти, зокрема HTTPS-порт веб-інтерфейсу.

Веб-інтерфейс Wazuh Dashboard є основним інструментом роботи адміністратора з платформою. Після успішного входу до нього користувач бачить зведену інформацію про стан серверних компонентів, повідомлення про відсутність підключених агентів, заготовлені розділи для аналізу подій безпеки.

Повідомлення про відсутність агентів є очікуваним і коректним станом для цього етапу курсу. Воно підтверджує, що серверна частина Wazuh працює правильно і готова до подальшого розширення.

Основи налаштування та експлуатації Wazuh

Підсумком етапу розгортання Wazuh має стати розгорнутий сервер безпекового моніторингу Wazuh з працездатним веб-інтерфейсом Dashboard, стабільною мережевою конфігурацією. Система має бути готова до підключення агентів і зовнішніх інтеграцій.

Wazuh є комплексною платформою безпекового моніторингу, що поєднує функції HIDS / SIEM / XDR. На відміну від класичних систем моніторингу доступності (Nagios, Icinga), Wazuh орієнтований на аналіз подій безпеки, журналів операційних систем, контроль цілісності файлів, виявлення шкідливої активності та реагування на інциденти.

У межах навчального стенду Wazuh використовується як центральний елемент безпекового моніторингу, який збирає події з різних операційних систем, корелює їх та надає аналітичний інтерфейс для дослідження інцидентів.

Агентна модель є ключовою для роботи Wazuh. Саме агенти забезпечують збір даних безпеки безпосередньо на вузлах інфраструктури. Розгортання агентів виконується централізовано через Wazuh Dashboard, що є важливою концепцією. Адміністратор не підбирає пакети вручну, система автоматично формує команди інсталяції, під час встановлення агент одразу реєструється на сервері.

Агент Wazuh виконує збір системних логів (Linux: auth.log, syslog; Windows: Event Log), інвентаризацію системи (процеси, пакети, мережеві інтерфейси), контроль цілісності файлів (FIM), виконання політик безпеки (SCA) та передачу подій на сервер по захищеному каналу.

Після успішної інсталяції агент переходить у стан Active, що означає повноцінну взаємодію з сервером Wazuh.

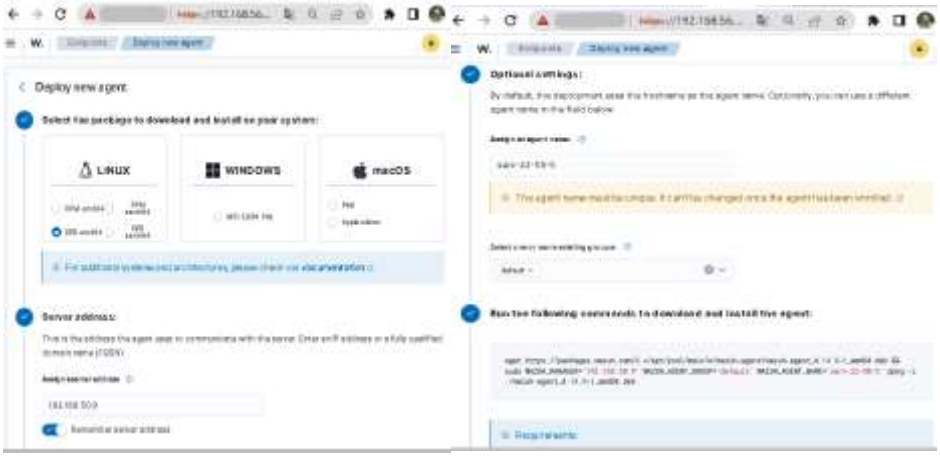

Рис. Р.8. Налаштування та експлуатація Wazuh

Однією з ключових задач платформи Wazuh є **централізований аналіз системних журналів** з різних операційних систем. У Linux-середовищі агент працює з типовими журналами автентифікації та системних подій, зокрема з /var/log/auth.log, де фіксуються спроби входу, та з /var/log/syslog або сервісом journald, який акумулює загальносистемні повідомлення. У Windows агент аналізує стандартні журнали Security, System та Application, що дозволяє отримати повну картину подій на вузлі. Зібрані події проходять послідовний ланцюг обробки: спочатку вони збираються агентом, далі передаються на Wazuh Manager, де декодуються, зіставляються з правилами кореляції, після чого формується алерт і він відображається у веб-інтерфейсі Dashboard. Такий підхід дає змогу виявляти невдалі спроби автентифікації, ескалацію привілеїв, підозрілу активність користувачів та різноманітні системні помилки ще на ранніх етапах.

Важливою складовою безпекового моніторингу є **контроль цілісності файлів, або File Integrity Monitoring**. Цей механізм дозволяє відстежувати будь-які зміни у визначених каталогах і файлах — створення, модифікацію або видалення. FIM є критично важливим для контролю системних конфігурацій, виявлення несанкціонованих змін і перевірки відповідності політикам безпеки. У межах лабораторних робіт на практиці демонструється, як додавати каталоги до моніторингу, як система реагує на зміну файлів і як відповідні FIM-алерти з'являються у Dashboard, формуючи чіткий слід інциденту.

Окрему увагу в курсі приділено **виявленню шкідливої активності та механізмам Active Response**. У цій моделі Wazuh поєднує контроль цілісності файлів із зовнішніми сервісами аналізу загроз, зокрема VirusTotal. Коли FIM фіксує появу або зміну файлу, цей файл може бути автоматично перевірений через VirusTotal API. У разі виявлення загрози система формує алерт, може виконати автоматичну дію — наприклад, видалити файл — і зафіксувати результат реагування. Для навчальних цілей використовується тестовий файл EICAR, який є безпечним стандартом для перевірки антивірусних механізмів і дозволяє наочно продемонструвати повний цикл роботи системи: від події через аналіз і прийняття рішення до активного реагування.

Завершальним етапом роботи з Wazuh є **аналітика та Threat Hunting**. Wazuh Dashboard надає інструменти, які дозволяють фільтрувати події за агентами, шукати конкретні індикатори компрометації, аналізувати історію інцидентів та корелювати події з різних джерел. Саме на цьому рівні Wazuh перестає бути просто системою збору логів і проявляється як повноцінна SIEM-платформа, орієнтована на глибокий аналіз подій безпеки та підтримку процесів реагування на інциденти.

Розгортання Splunk на сервері Ubuntu

Splunk — це універсальна платформа для збору, індексації, пошуку та візуалізації машинних даних у режимі реального часу. Вона широко застосовується для аналізу системних і прикладних логів, моніторингу інфраструктури, а також для задач інформаційної безпеки (SIEM-подібні сценарії).

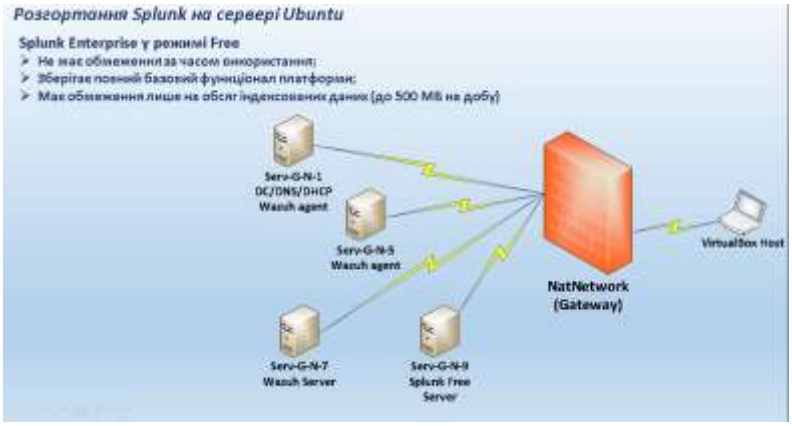

Рис. Р.9. Розгортання Splunk на сервері Ubuntu

В основі Splunk лежить концепція централізованого збирання подій з різних джерел, їх нормалізації та збереження у власному індексі з можливістю подальшого швидкого пошуку й аналітики через веб-інтерфейс Splunk Web

У межах навчального курсу використовується Splunk Enterprise у режимі Free, який:

- не має обмеження за часом використання;
- зберігає повний базовий функціонал платформи;
- має обмеження лише на обсяг індексованих даних (до 500 МБ на добу), що є достатнім для лабораторних стендів

У лабораторних роботах Splunk розгортається шляхом локальної інсталяції на сервері Ubuntu Server 24.04, який імпортується з готового аплайнсу (OVA-образу) та інтегрується у віртуалізоване стендове середовище VirtualBox. Такий підхід дозволяє отримати повний контроль над конфігурацією Splunk, збирати локальні журнали Linux (/var/log), забезпечити подальшу інтеграцію з системою безпекового моніторингу Wazuh та відпрацювати практичні навички встановлення, первинного налаштування та експлуатації SIEM-подібної платформи у реальному серверному середовищі.

У рамках курсу Splunk використовується як система централізованого збору та аналізу логів, інструмент кореляції подій, платформа візуалізації та аналітики і нарешті, як кінцева точка у конвеєрі Wazuh → Splunk, що демонструє практичну інтеграцію системного та безпекового моніторингу.

Побудова та налаштування конвеєра Wazuh → Splunk

Завершальним етапом безпекового моніторингу в нашому курсі є побудова та налаштування конвеєра інтеграції між Wazuh та Splunk. На цьому етапі ми фактично поєднуємо дві системи в єдиний ланцюг обробки подій безпеки: Wazuh виступає джерелом і первинним аналітиком подій, а Splunk — платформою для централізованої індексації, кореляції та подальшого аналізу.

Логіка цього конвеєра доволі проста, але показова. Події безпеки спочатку збираються агентами Wazuh на кінцевих вузлах, обробляються менеджером, проходять етапи декодування та застосування правил, після чого передаються у вигляді структурованих повідомлень. Далі ці події транспортуються через Filebeat та pipeline обробки, де їм надається узгоджений формат і вони спрямовуються до Splunk для індексації. Таким чином ми отримуємо чітке розмежування ролей: Wazuh відповідає за виявлення та первинну інтерпретацію подій, Splunk — за їх централізований аналіз і візуалізацію.

У процесі налаштування цього конвеєра особлива увага приділяється конфігураційним файлам, які визначають логіку передачі та обробки даних. Зокрема, саме у файлі pipeline ***/etc/logstash/conf.d/wazuh-to-splunk.conf*** описується, які події приймаються, як вони трансформуються та куди саме передаються. Файл ***/etc/filebeat/filebeat.yml***, у свою чергу, відповідає за коректну доставку подій, визначає джерела логів і параметри з'єднання між компонентами. Хоча детальні лістинги цих файлів наведені в методичних рекомендаціях, на лекції важливо розуміти їхню роль у загальній архітектурі, а не лише синтаксис.

Окремо варто наголосити на мережевих аспектах лабораторного стенду. У методичних рекомендаціях детально описано налаштування Port Forwarding, і ці кроки не є формальністю. Використання пробросу портів дозволяє працювати з веб-інтерфейсами Wazuh та Splunk з хостової системи, не перевантажуючи віртуальні машини відображенням важких веб-сторінок. Крім того, це значно знижує ризик помилок під час роботи в командному рядку та редагування конфігураційних файлів безпосередньо у віртуальному середовищі.

Таким чином, побудова конвеєра Wazuh → Splunk є не просто технічною інтеграцією двох продуктів, а логічним завершенням курсу безпекового моніторингу. Вона демонструє, як окремі інструменти поєднуються у єдину систему, де події безпеки проходять повний життєвий цикл — від збору та аналізу до централізованої обробки та аналітики. Саме цей підхід відображає реальні практики побудови сучасних SOC-інфраструктур.

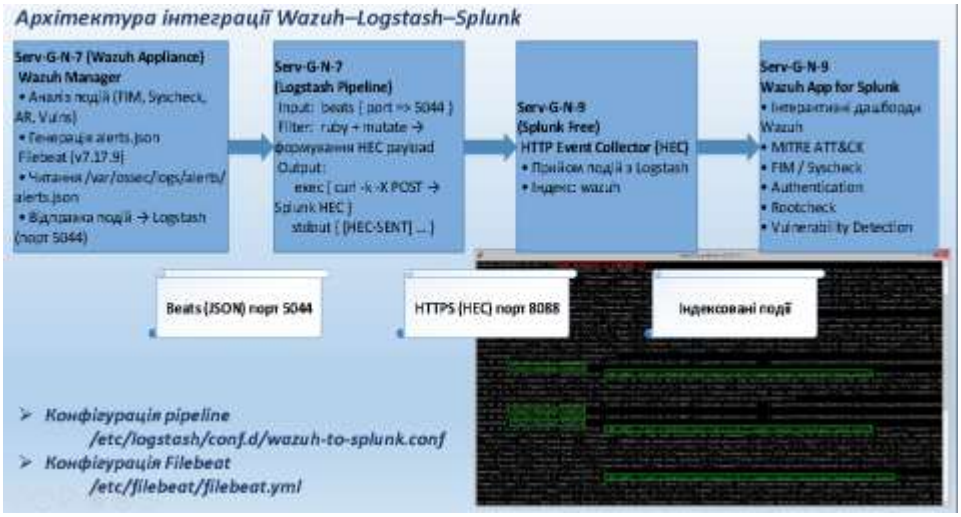

Рис. Р.10. Архітектура інтеграції Wazuh–Logstash–Splunk