Лабораторна робота №11

Аналіз системних логів із Wazuh (auth.log, syslog, Event Log).

Мета: Набути практичних навичок аналізу системних логів за допомогою платформи Wazuh, навчитися виявляти події безпеки на основі журналів auth.log, syslog та Event Log. Ознайомитися з механізмами обробки, фільтрації та класифікації подій у Wazuh Dashboard, виявити невдалі спроби автентифікації, ескалації привілеїв та інші критичні інциденти, що впливають на безпеку інформаційних систем.

Інструменти: гіпервізор VirtualBox, модель комп'ютерної мережі.

Теоретичні відомості

У попередніх лабораторних роботах було створено віртуалізоване стендове середовище у VirtualBox, що складається з трьох хостів:

Serv-G-N-1 (Windows Server 2022) – контролер домену з ролями AD DS, DNS і DHCP. Налаштовано Wazuh Agent для локального моніторингу ресурсів.

Serv-G-N-5 (Ubuntu Server 24.04) – сервер на налаштовано Wazuh Agent для локального моніторингу ресурсів.

Serv-G-N-7 (Amazon Linux 2023) – сервер Wazuh Appliance, що містить компоненти Wazuh Server (Manager, API) та Elasticsearch + Kibana (Dashboard)

Мережеве середовище забезпечує взаємодію між вузлами з доменною інфраструктурою для подальшого моніторингу її елементів.

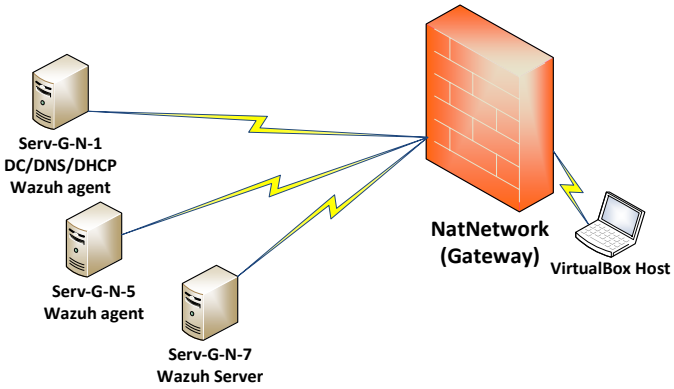


Рис. 11.1. Топологія мережі

Перевірка конфігурації збору логів (auth.log, syslog, Windows Event Log)

Мета: упевнитися, що агенти збирають необхідні джерела логів (на Ubuntu — /var/log/auth.log, /var/log/syslog; на Windows — журнали System / Security / Application) і що ці події надходять на сервер Wazuh і відображаються у Dashboard.

- **Перевірка на хості Ubuntu (Serv-G-N-5)**

Підключаємося до Ubuntu через SSH та переконуємося, що присутні системні журнали:

```
ls -l /var/log/auth.log /var/log/syslog
tail -n 30 /var/log/auth.log
tail -n 30 /var/log/syslog
```

Перевіряємо стан агента Wazuh на хості:

```
sudo systemctl status wazuh-agent
```

Переглядаємо лог агента, що відстежує відправку подій до сервера (якщо файл логів у іншому місці — знайдіть ossec.log всередині /var/ossec/ або /var/log/). Переглядаємо та аналізуємо помилки, якщо вони присутні. Це можуть бути повідомлення про відмову підключення, сертифікати, права доступу.

```
sudo tail -n 200 /var/ossec/logs/ossec.log
```

Та перевіряємо з'єднання до менеджера:

```
nc -vz 192.168.50.9 1514
```

На рис.11.2 приведено фрагменти логів, що демонструють нормальну поведінку агента Wazuh під час роботи різних модулів моніторингу та взаємодії з сервером.

2025/10/29 06:31:33 Повідомлення модуля wazuh-syscheckd: INFO: netstat not available. Skipping port check. Під час сканування модуль намагався виконати додаткову перевірку відкритих портів, але не знайшов утиліту netstat. Через це етап “port check” пропущено.

2025/10/29 09:41:18 Модуль: wazuh-agentd. Агент на короткий час втратив з’єднання з сервером Wazuh (порт 1514/tcp — канал передачі логів). Причиною може бути коротке перезавантаження сервера, рестарт служби wazuh-manager, або короткий збій мережі у VirtualBox NAT.

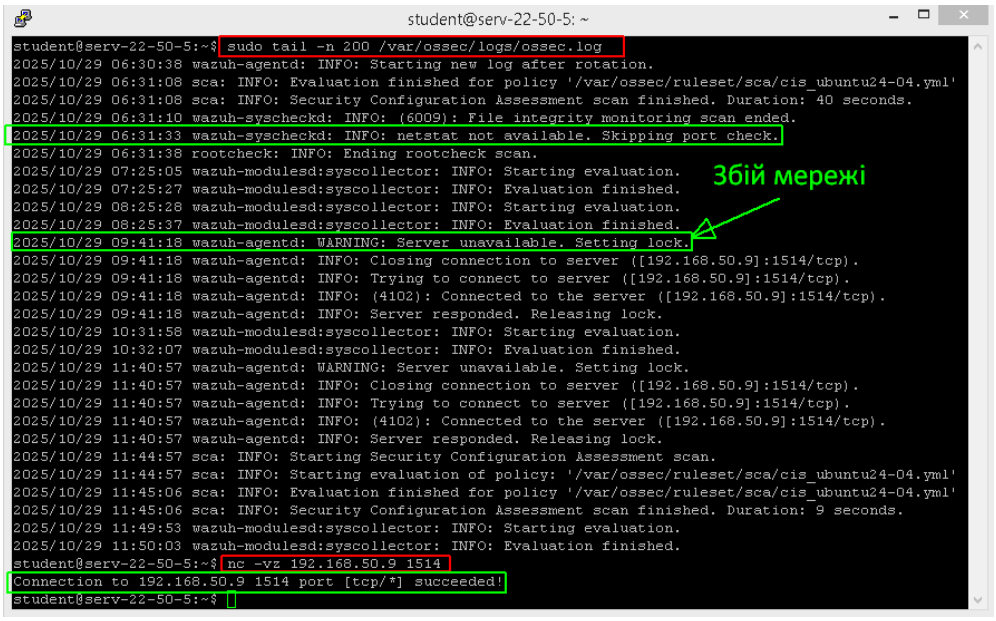


Рис. 11.2. Аналіз логу агента (файл /var/ossec/logs/ossec.log) та перевірка доступності серверу Wazuh

Вирішимо проблему, що викликала повідомлення модуля wazuh-syscheckd: INFO: netstat not available. Skipping port check. Ця проблема пов’язана з відсутністю утиліти netstat, яку Wazuh використовує для додаткової перевірки відкритих портів під час сканування цілісності. У сучасних дистрибутивах Ubuntu netstat більше не входить у базову установку — вона міститься у пакеті net-tools. Встановлюємо пакет net-tools та перевіряємо доступність netstat (очікуваний результат /usr/bin/netstat):

sudo apt update
sudo apt install net-tools -y
which netstat

Перезапускаємо службу агента Wazuh

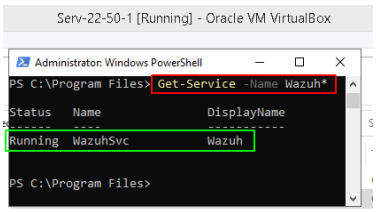
sudo systemctl restart wazuh-agent

Перезапуск потрібен, щоб модуль syscheckd при наступному циклі перевірки повторно ініціалізував усі залежності — включно з netstat. Через кілька хвилин після рестарту переглядаємо свіжі записи у логах:

sudo tail -n 50 /var/ossec/logs/ossec.log | grep netstat

Очікується, що повідомлення “netstat not available” більше не з’являтиметься.

❖ **Перевірка на Windows Server (Serv-G-N-1)**



Перевіряємо у Power Shell, що служба агента запущена:

Get-Service -Name Wazuh*

Якщо агент працює, у полі Status буде зазначено Running.

У старіших версіях служба може мати іншу назву, тому у разі потреби перевіряємо також:

Get-Service -Name Ossec*

Перевіряємо лог-файли агента. Поточний журнал роботи агента розташований за адресою ***C:\Program Files (x86)\ossec-agent\ossec.log***. Для перегляду останніх подій виконуємо команду (рис.11.3):

Get-Content "C:\Program Files (x86)\ossec-agent\ossec.log" -Tail 50 -Wait

Параметр -Wait дозволяє у реальному часі відслідковувати нові записи, що надходять до журналу.

Wazuh автоматично архівує старі журнали у стиснутому форматі .gz у каталозі
C:\Program Files (x86)\ossec-agent\logs\<рік>\<місяць>
Наприклад: ***C:\Program Files (x86)\ossec-agent\logs\2025\Oct\ossec-29.log.gz***

Такі файли можна переглядати за допомогою утиліт, що підтримують GZIP-архіви (7-Zip, WinRAR, або через PowerShell із використанням Expand-Archive).

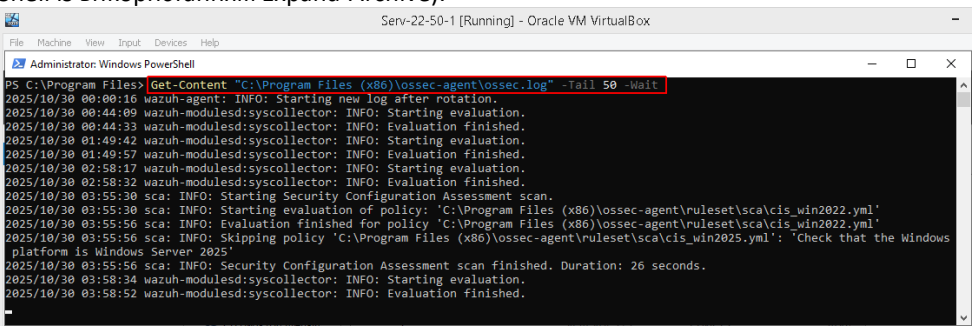


Рис. 11.3. Перегляд останніх подій у файлі ossec.log на Serv-22-50-1

На рис.11.3 ми бачимо, що ossec.log має лише записи:

wazuh-agent: INFO: Starting new log after rotation.
wazuh-modulesd:syscollector: INFO: Starting evaluation.
wazuh-modulesd:syscollector: INFO: Evaluation finished.

Це означає, що Wazuh Agent запущений і справно передає інформацію про систему (syscollector = збір інформації про ОС, пакети, інтерфейси, диски тощо), але модуль “logcollector” (збір журналів подій Windows) або не ввімкнено, або налаштований тільки частково (наприклад, не слухає “Security” лог).

Wazuh на Windows може збирати журнали через модуль logcollector, який читає події з

- Application
- System
- Security
- Microsoft-Windows-* (за потреби)

Щоб він почав це робити, у файлі ***C:\Program Files (x86)\ossec-agent\ossec.conf*** повинні бути розділи:

```
<localfile>
  <location>Security</location>
  <log_format>eventchannel</log_format>
</localfile>

<localfile>
  <location>System</location>
  <log_format>eventchannel</log_format>
</localfile>

<localfile>
  <location>Application</location>
  <log_format>eventchannel</log_format>
</localfile>
```

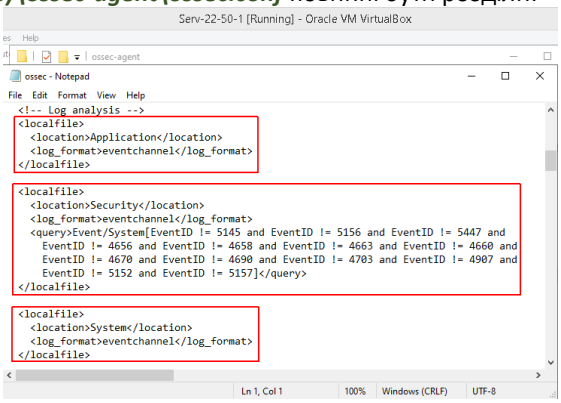


Рис. 11.4. Перегляд налаштувань у ossec.conf

Переглядаємо файл конфігурації (рис.11.4) та бачимо, що відповідні секції у налаштуваннях агента присутні. Якщо таких блоків немає — агент не читатиме події журналів подій Windows. Спробуємо викликати події на сервері, які спричинять перевірку політики безпеки. Найпростіша дія – зміна користувача на сервері.

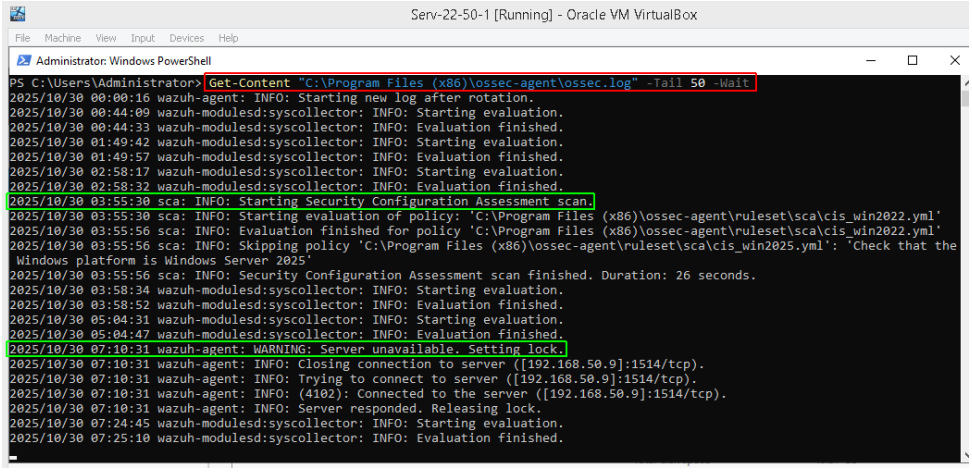


Рис. 11.5. Повторний перегляд останніх подій у файлі ossec.log на Serv-22-50-1

Після створення нового користувача, входу під ним та повернення до адміністратора у логах агента з'являються записи (рис.11.5) про запуск модулів syscollector і SCA (Security Configuration Assessment). Це очікувана реакція Wazuh — під час змін у системі (створення облікових записів, вхід користувачів, зміна політик) агент автоматично оновлює інвентаризацію та перевіряє відповідність безпековим політикам. Таким чином підтверджується, що збір подій з каналів Security і System працює коректно.

2025/10/30 03:55:30 sca: INFO: Starting Security Configuration Assessment scan. Тут все добре: політика запущена і завершилася успішно. ***Skipping policy ... cis_win2025.yml: 'Check that the Windows platform is Windows Server 2025'*** Це не помилка — просто політика для Windows Server 2025 пропущена, бо у нас Windows Server 2022.

2025/10/30 07:10:31 wazuh-agent: WARNING: Server unavailable. Setting lock. Короткочасна втрата зв'язку з сервером (можливо, був рестарт менеджера або проблема з мережею).

2025/10/30 07:10:31 wazuh-agent: INFO: (4102): Connected to the server ([192.168.50.9]:1514/tcp). Через кілька секунд агент успішно перепідключився — отже, все відновилося автоматично.

❖ **Перевірка у Wazuh Dashboard (сервер)**

Відкриваємо у Wazuh Dashboard меню Agents Management – Summary. Статус агентів має бути Active.

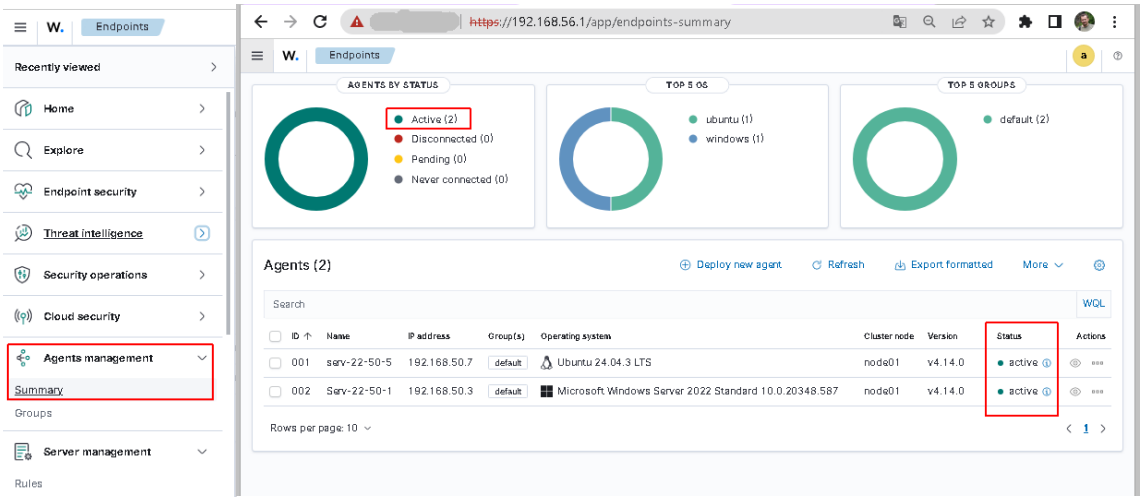


Рис. 11.6. Перевірка статусів агентів у Wazuh Dashboard

Переключаємось у меню Threat Intelligence – Threat Hunting та змінюємо представлення з Dashboard на Events. (serv-g-n-5, serv-g-n-1).

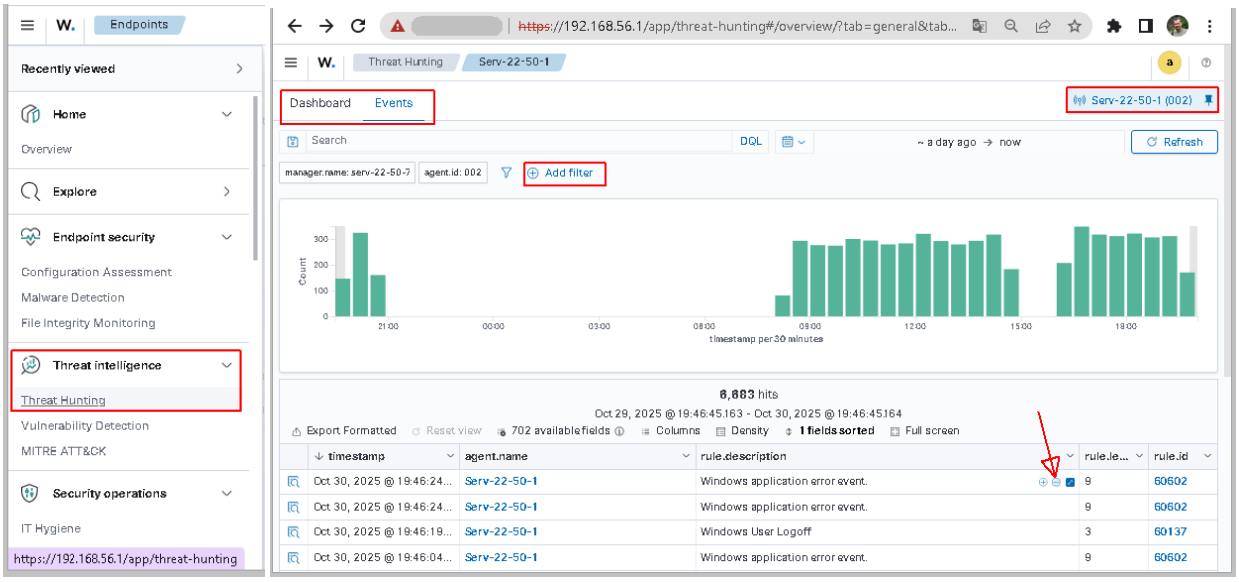


Рис. 11.7. Підменю Events для Serv-22-50-1 меню Threat Intelligence – Threat Hunting

Для кожного агента будуємо найпростіші фільтри для пошуку login/logoff подій (рис.11.7). Фільтр можна будувати через додавання умов Add filter, або вимиканням певних типів подій, що не задовольняють критеріям фільтрації. Зверніть увагу, що всі додані фільтри зберігаються у верхній частині цього ж вікна.

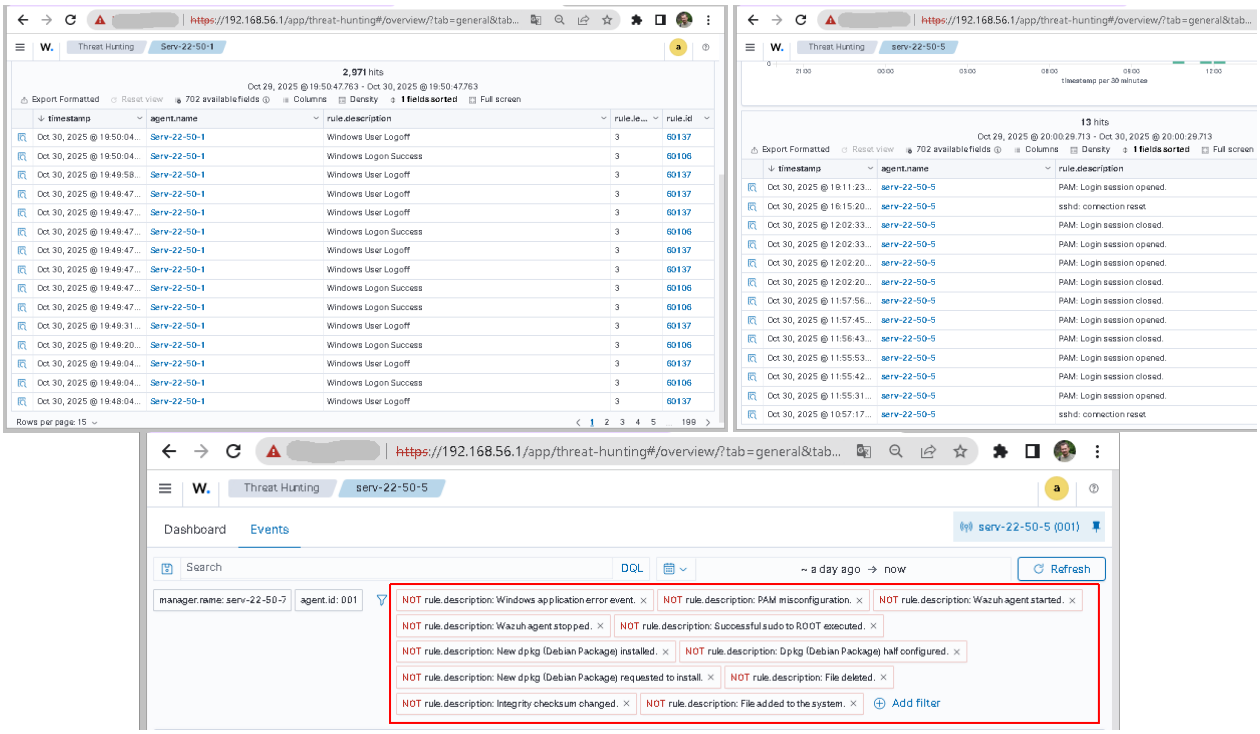


Рис. 11.8. Відфільтровані події для Serv-22-50-1 та Serv-22-50-5 та перелік доданих фільтрів.

Якщо агент показує Never connected / Disconnected — перевіряємо адресу менеджера, брандмауери, порти (1514/1515). Перевіряємо час системи на всіх вузлах (час/таймзона). Переглядаємо логи агента (/var/ossec/logs/ossec.log) та менеджера (/var/ossec/logs/ossec.log на сервері).

Симуляція невдалих входів (без шкідливих дій) — безпечно

Подібні симуляції виконуються тільки в ізольованих стендах VirtualBox. Не проводьте справжніх брутфорс-атак або тестів на продуктивних системах. Перед початком бажано створити снапшот VM.

Згенеруємо контрольовані записи про невдалі спроби автентифікації і перевіримо, що Wazuh виявляє їх як «failed login».

❖ **Ubuntu (Serv-G-N-5)**

У конфігурації агента Wazuh передбачено збір подій із системного журналу journald (див. секцію <log_format>journald</log_format>). Тому імітаційні події слід створювати командою logger, яка автоматично записує повідомлення у системний журнал.

```
for i in {1..5}; do \
  logger -p authpriv.warning "sshd[9999]: Failed password for invalid user testuser${i} from 192.0.2.1
port $((5550 + i)) ssh2"; \
  sleep 1; \
done
```

Перевіряємо, що події з’явилися у локальному журналі:

```
sudo journalctl | grep "Failed password for invalid user testuser"
```

Якщо події видно — агент зчитав їх із journald і має передати на Wazuh Server.

Перевіримо отримання подій на сервері Wazuh (serv-22-50-7 через консоль), де виконуємо команду для пошуку конкретного користувача:

```
sudo grep testuser /var/ossec/logs/alerts/alerts.log
```

Якщо передача працює, у логах з’являться події виду:

Рис. 11.9. Пошук у alerts.log користувача testuser.

Nov 03 18:20:48 serv-22-50-7 sudo[6220]: wazuh-user : TTY=pts/0 ; PWD=/home/wazuh-user ; USER=root ; COMMAND=/usr/bin/grep testuser /var/ossec/logs/alerts/alerts.log command: /usr/bin/grep testuser /var/ossec/logs/alerts/alerts.log

Вивід на рис.11.9 означає, що агент успішно надіслав подію, декодер sshd її розпізнав, а відповідне правило спрацювало.

Перевіримо отримання інформації про подію у Wazuh Dashboard. У вебінтерфейсі Wazuh переходимо до меню Threat Intelligence – Threat Hunting (Events). У полі Search вводимо testuser або "Failed password", а у полі Agent name обираємо serv-22-50-5. У результаті пошуку за ключовими словами або "Failed password" події не виявлено. Проте під час пошуку за ключовим словом testuser за агентом serv-22-50-5 відображається подія

Rule ID: 5402
Rule description: "Successful sudo to ROOT executed."
Rule level: 3

Агент успішно передає події з системного журналу на сервер Wazuh, однак тестові події типу "Failed password for invalid user …" не були інтерпретовані системою.

Якщо події не з'явились у Dashboard, але є в alerts.log, то проблема у фільтрах інтерфейсу Kibana/Elastic (варто перевірити діапазон часу або індекси). Якщо подій немає навіть у alerts.log, тоді перевірити з'єднання агента (/var/ossec/logs/ossec.log).

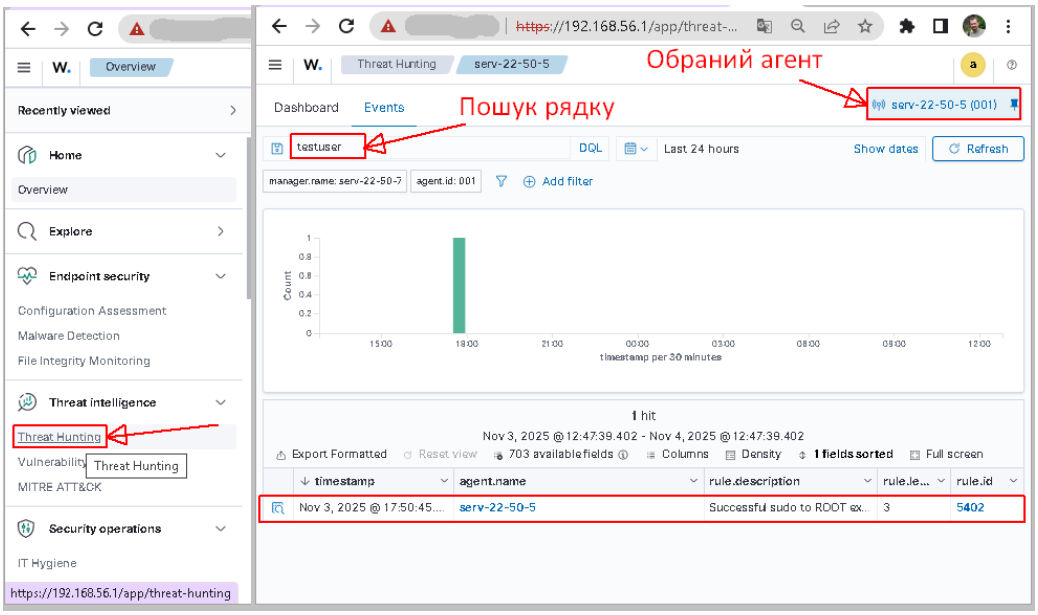


Рис. 11.10. Пошук подій за ключовим словом testuser за агентом serv-22-50-5.

❖ **Windows (Serv-G-N-1)**

Створимо тестову подію на сервері. Виконуємо у PowerShell (з правами адміністратора) додавання тестового запису у Application log:

eventcreate /T ERROR /ID 1000 /L APPLICATION /SO "WazuhTest" /D "Failed logon for user testuser"

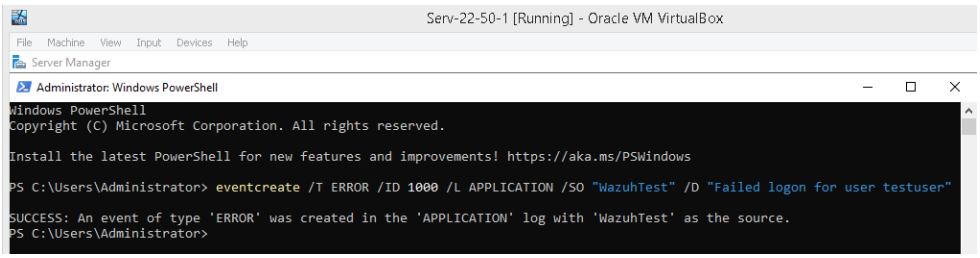


Рис. 11.11. Створення тестової події у PowerShell на Serv-22-50-1.

Після генерації подій на Serv-G-N-1 переглядаємо, чи з'явились алерти на сервері Wazuh:

sudo grep WazuhTest /var/ossec/logs/alerts/alerts.log

Рис. 11.12. Перегляд тестової події у консолі серверу Wazuh (Serv-22-50-7).

На сервері Wazuh виконуємо команду:

sudo grep WazuhTest /var/ossec/logs/alerts/alerts.log

Результат:

{"win":{"system":{"providerName":"WazuhTest","eventID":"1000","level":"2","channel":"Application","computer":"Serv-22-50-1.falkovsky.net","severityValue":"ERROR","message":"\"Failed logon for user testuser\""},"eventdata":{"data":"Failed logon for user testuser"}}}

win.system.providerName: WazuhTest

Подія успішно отримана сервером (рис.11.12). З журналу видно, що джерело події (ProviderName) ***WazuhTest***, рівень ***ERROR*** , подія (Event ID) ***1000***, хост: ***Serv-22-50-1.falkovsky.net*** та опис: ***"Failed logon for user testuser"***

Відкриваємо Wazuh Dashboard – Threat Hunting (Events). У фільтрах обираємо відповіднийAgent (serv-g-n-1 та у полі пошуку вводимо WazuhTest або EventID:1000. Перевіряємо підказки/alert cards, де має з'явитись відповідна подія/аларм.

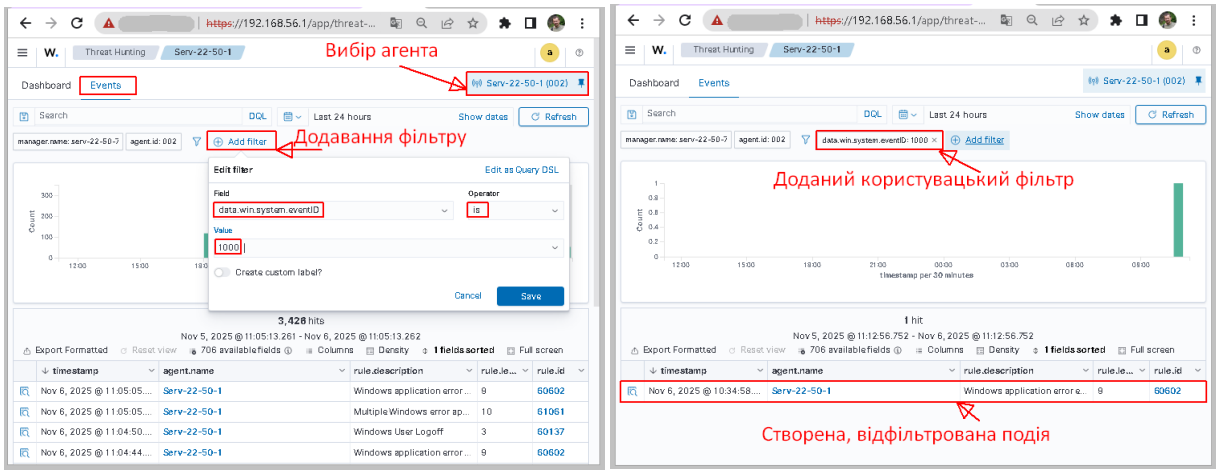


Рис. 11.13. Перегляд тестової події у Wazuh Dashboard.

Якщо події не з'явились, виконуємо швидку діагностику. Перевіряємо, чи агент Active у Dashboard. На агенті перевіряємо, що він читає відповідний журнал:

Ubuntu: ***/var/ossec/etc/ossec.conf має <localfile> для /var/log/auth.log / syslog.***

Windows: ***ossec.conf має <location>Security|System|Application</location> (eventchannel).***

Перевіряємо логи агента:

Ubuntu: ***sudo tail -n 200 /var/ossec/logs/ossec.log***

Windows: ***Get-Content "C:\Program Files (x86)\ossec-agent\ossec.log" -Tail 100***

У логах менеджера на сервері шукаємо помилки з'єднання, відхилені події або помилки розбору:

sudo tail -n 200 /var/ossec/logs/ossec.log

sudo tail -n 200 /var/ossec/logs/alerts/alerts.log

Симуляція інциденту FIM (File Integrity Monitoring)

Механізм FIM (File Integrity Monitoring) у Wazuh забезпечує контроль цілісності критичних файлів та каталогів, відстежуючи будь-які зміни — створення, редагування або видалення. Перевіримо працездатність FIM на тестовому файлі.

- **Ubuntu (Serv-G-N-5)**

Відкриваємо конфігураційний файл агента Wazuh ***/var/ossec/etc/ossec.conf***, знаходимо секцію ***<syscheck>*** та додаємо у ній шляхи до тестових каталогів, що скануються у реальному часі:

<directories realtime="yes" report_changes="yes">/tmp</directories>
<directories realtime="yes" report_changes="yes">/root</directories>

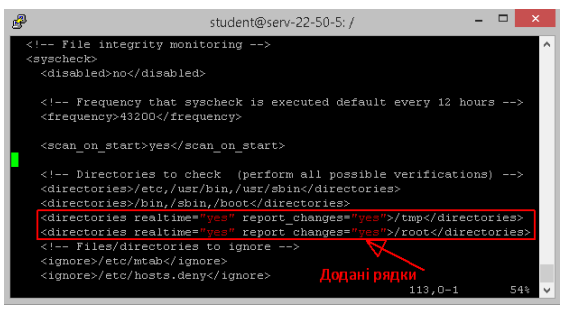


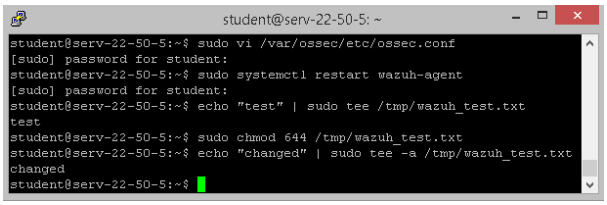
Рис. 11.14. Редагування на /var/ossec/etc/ossec.conf

realtime="yes" вмикає миттєвий контроль змін файлу — не потрібно період, який задано показником frequency у цій же секції. Зберігаємо зміни та перезапускаємо агента:

sudo systemctl restart wazuh-agent

Створимо і змінимо файл. Ця послідовність дій має викликати генерацію FIM-сповіщення:

echo "test" | sudo tee /tmp/wazuh_test.txt
sudo chmod 644 /tmp/wazuh_test.txt
echo "changed" | sudo tee -a /tmp/wazuh_test.txt



Перевіряємо чи отримано події на сервері Wazuh. Виконуємо команду:

sudo grep wazuh_test /var/ossec/logs/alerts/alerts.log

На рис.11.15 показаний результат перегляду уконсолі, де з'явилися записи FIM-сповіщення, що підтверджує виявлення змін агентом та передачу цієї інформації на сервер Wazuh.

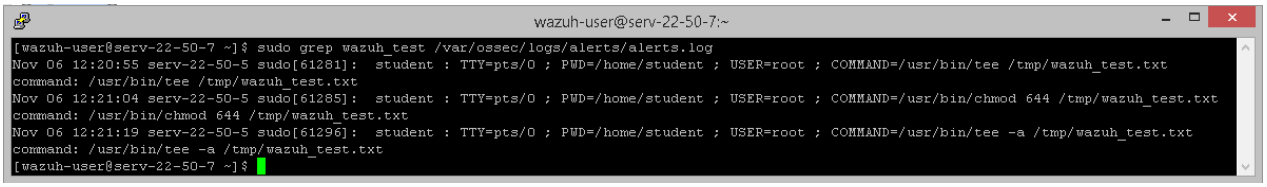


Рис. 11.15. Пошук записів FIM-сповіщення у alerts.log на сервері Wazuh

Перевірка у Wazuh Dashboard виконується у меню Endpoint security – File Integrity Monitoring. У фільтрах обираємо відповідний Agent та налаштовуємо відповідні користувацькі фільтри.

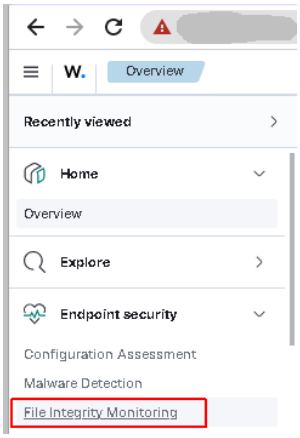


Рис. 11.16. Меню FIM

❖ **Windows (Serv-G-N-1)**

Для симуляції на Windows (для контролера домену) у директорії C:\ створюємо файл та вносимо у нього зміни:

echo "test" > C:\wazuh_test.txt
echo "changed" >> C:\wazuh_test.txt

Через 1–2 хвилини події буде передано агентом до сервера Wazuh. Перевірка у Dashboard виконується аналогічно перевірці для Ubuntu (Serv-G-N-5)

Таким чином ми впевнились, що система Wazuh успішно виявляє створення та зміну файлів, формує події контролю цілісності (FIM-Alerts) і передає їх на сервер для подальшого аналізу через Dashboard. Механізм FIM може бути використаний для моніторингу системних конфігураційних файлів, політик безпеки та важливих директорій користувачів.

Імітація індикатора компроментації (EICAR) — безпечно

EICAR-тест — це спеціально створений безпечний текстовий файл, який використовується для перевірки роботи антивірусних систем, модулів детекції шкідливого ПЗ та інших засобів безпеки, таких як Wazuh, без жодного ризику зараження. Абревіатура EICAR – European Institute for Computer Antivirus Research.

Цей файл був розроблений Європейським інститутом комп'ютерних антивірусних досліджень (EICAR) у співпраці з CARO (Computer Antivirus Research Organization). Його мета — надати універсальний, стандартизований спосіб перевірки реакції антивірусних і моніторингових систем без використання реального шкідливого коду.

Вміст EICAR-тесту виглядає так:

X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*

Цей рядок не містить жодних шкідливих інструкцій, але імітує сигнатуру вірусу, оскільки більшість антивірусів мають у своїх базах даних спеціальну сигнатуру для цього шаблону. Його можна розмістити у звичайному текстовому файлі (наприклад /tmp/eicar.txt), і система безпеки має відреагувати на нього як на потенційно небезпечний об'єкт.

Таким чином, цей файл дозволяє протестувати, чи агент виявлення шкідливих програм (наприклад, у Wazuh, Windows Defender, ClamAV тощо) активно працює. Дає змогу перевірити, чи система коректно формує повідомлення (alert) та переконатися, що події передаються до SIEM або аналітичної консолі (у нашому випадку — Wazuh Dashboard).

> EICAR — єдиний офіційно затверджений безпечний тестовий "вірус", який допускається для лабораторних і навчальних робіт. Заборонено використовувати або поширювати реальні шкідливі зразки навіть у навчальних цілях.

Відповідно до сценарію емуляції вірусної атаки з документації продукту, увімкнемо на сервері Wazuh FIM для моніторингу, інтеграцію VirusTotal (автоматична перевірка через API будь-яких нових або змінених файлів у директорії /root, надісланих агентом FIM), Active Response (автоматичне видалення файлу, якщо VirusTotal виявить його як шкідливий) та правила для створення сповіщень в Dashboard про успішне або неуспішне видалення.

Редагуємо файл правил Wazuh ***/var/ossec/etc/rules/local_rules.xml***, додаючи у кінець файлу секції:

```
<group name="syscheck,pci_dss_11.5,nist_800_53_SI.7,">
  <!-- Rules for Linux systems -->
  <rule id="100200" level="7">
    <if_sid>550</if_sid>
    <field name="file">/root</field>
    <description>File modified in /root directory.</description>
  </rule>

  <rule id="100201" level="7">
    <if_sid>554</if_sid>
    <field name="file">/root</field>
    <description>File added to /root directory.</description>
  </rule>
</group>

<group name="virustotal,">
 <rule id="100092" level="12">
  <if_sid>657</if_sid>
  <match>Successfully removed threat</match>
  <description>$(parameters.program) removed threat located at
$(parameters.alert.data.virustotal.source.file)</description>
 </rule>

 <rule id="100093" level="12">
  <if_sid>657</if_sid>
  <match>Error removing threat</match>
  <description>Error removing threat located at $(parameters.alert.data.virustotal.source.file)</description>
 </rule>
</group>
```

Для отримання безкоштовного Virus Total API ключа, переходимо на сайт https://www.virustotal.com/gui/join-us та створюємо безкоштовний акаунт. При створенні акаунту використовуємо власну електронну пошту, де підтверджуємо реєстрацію. Після входу в акаунт натискаємо на свій профіль (іконка у правому верхньому куті сайту) та обираємо пункт “API key”. Там можна переглянути, або скопіювати у буфер обміну свій персональний ключ. Ключ виглядає як довгий рядок латинських символів і цифр.

Копіюємо цей ключ. Він знадобиться для файлу ***/var/ossec/etc/ossec.conf***.

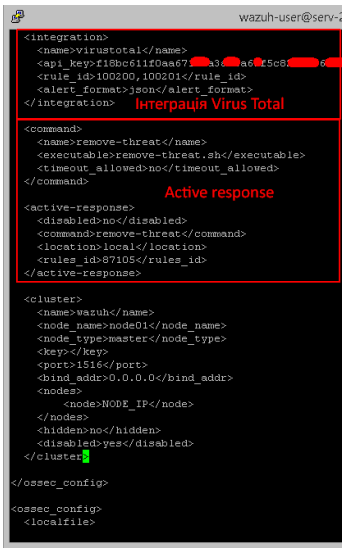
Безкоштовний ключ має обмеження: максимум 4 запити за хвилину. Для нашого тесту цього цілком достатньо. Якщо Wazuh часто сканує файли, можна зробити паузу між тестами.

У файлі конфігурації ***/var/ossec/etc/ossec.conf*** додаємо блоки у кінець загальної секції <ossec_config>, перед секцією <cluster> Обов’язково замінюємо ***YOUR_VIRUS_TOTAL_API_KEY*** на власний Virus Total API key.

```
 <integration>
  <name>virustotal</name>
  <api_key>YOUR_VIRUS_TOTAL_API_KEY</api_key>
  <rule_id>100200,100201</rule_id>
  <alert_format>json</alert_format>
 </integration>

<ossec_config>
 <command>
  <name>remove-threat</name>
  <executable>remove-threat.sh</executable>
  <timeout_allowed>no</timeout_allowed>
 </command>

 <active-response>
  <disabled>no</disabled>
  <command>remove-threat</command>
  <location>local</location>
  <rules_id>87105</rules_id>
 </active-response>
```



При додаванні інтеграційних блоків у ossec.conf важливо пам'ятати, що не потрібно обгортати блок другим <ossec_config> — його треба просто вставити всередину існуючого <ossec_config>, перед <cluster>. Інакше при перезапуску конфігурації wazuh-manager видасть помилку на дублікат тегу <ossec_config>

Перезапускаємо сервіс:

sudo systemctl restart wazuh-manager

Для імітації зараження на Ubuntu-агенті виконуємо команду завантаження безпечного тестового зразка антивірусної індустрії:

sudo curl -Lo /root/eicar.com https://secure.eicar.org/eicar.com && sudo ls -lah /root/eicar.com

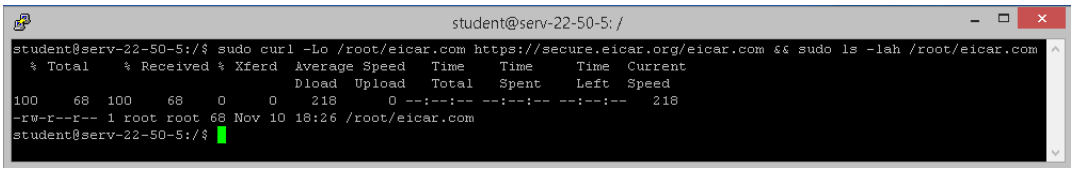


Рис. 11.17. Завантаження EICAR на сервер Ubuntu Serv-22-50-5.

Через 1–2 хвилини переходимо до Wazuh Dashboard – Endpoint Security – Malware Detection, обираємо відповідний Agent та можемо налаштувати відповідні користувацькі фільтри. Наприклад rule.id:(87105). В подіях має з'явитись щось типу «VirusTotal: Alert - /root/eicar.com», рис.11.18.

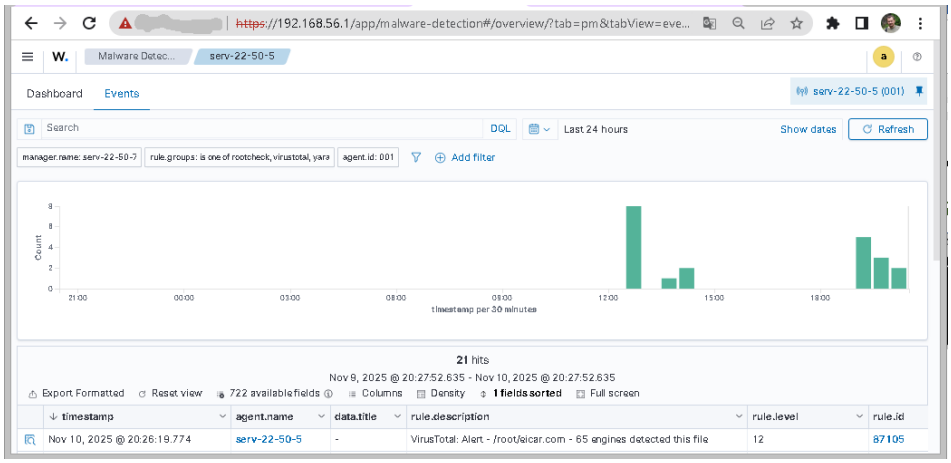


Рис. 11.18. Endpoint Security – Malware Detection. Подія «VirusTotal: Alert - /root/eicar.com»

❖ **Створення EICAR на Windows (агент)**

Під час виконання аналогічного тесту на Windows-агенті слід врахувати, що вбудовані антивірусні засоби (наприклад, Microsoft Defender) можуть автоматично видалити або помістити тестовий файл у карантин одразу після його створення. Це є нормальною поведінкою системи захисту і не вказує на помилку в роботі агента Wazuh.

Перевірка завантаження або створення EICAR-файлу на контролері домену не входить до обов'язкових завдань цієї лабораторної роботи, оскільки антивірусна політика Windows Server за замовчуванням блокує такі дії. Достатньо виконати тестування на Linux-агенті для підтвердження працездатності інтеграції з VirusTotal.

Завдання до лабораторної роботи

1. Перевірити роботу механізму File Integrity Monitoring (FIM) на агенті Ubuntu (Serv-G-N-5) шляхом створення та зміни тестового файлу.
2. Налаштувати інтеграцію Wazuh – VirusTotal на сервері (Serv-G-N-7) для перевірки нових або змінених файлів.
3. Створити правила для обробки подій FIM та перевірки файлів через VirusTotal.
4. Увімкнути Active Response для автоматичного видалення виявлених загроз.
5. Імітувати інцидент за допомогою тестового файлу EICAR і перевірити реакцію системи у Wazuh Dashboard.
6. Зробити висновок про взаємодію FIM, VirusTotal і Active Response у процесі виявлення та реагування на загрози.

Корисні посилання

- Navigating the Wazuh dashboard
https://documentation.wazuh.com/current/user-manual/wazuh-dashboard/navigating-the-wazuh-dashboard.html

- Виявлення аномалій за допомогою Wazuh
https://corewin.ua/blog/enhancing-it-security-with-anomaly-detection/

- Використання Wazuh для моделі безпеки Zero Trust
https://corewin.ua/blog/leveraging-wazuh-for-zero-trust-security/

- EICAR – European Institute for Computer Antivirus Research.
https://www.eicar.org/

- Wikipedia. EICAR-Test-File
https://en.wikipedia.org/wiki/EICAR-Test-File

- Detecting and removing malware using VirusTotal integration
https://documentation.wazuh.com/current/proof-of-concept-guide/detect-remove-malware-virustotal.html