DOI: https://doi.org/10.36910/6775-2524-0560-2022-49-03
УДК 004.94:656.02
Багнюк Наталія Володимирівна, к.т.н., доцент
https://orcid.org/0000-0002-7120-5455
Бортник Катерина Яківна, к.т.н., доцент
https://orcid.org/ 0000-0001-5282-099X
Варченко Леонід Леонідович, магістр
Луцький національний технічний університет, м.Луцьк, Україна

МОНІТОРИНГОВА СИСТЕМА ДЛЯ ОПЕРАЦІЙНОЇ СИСТЕМИ WINDOWS

Багнюк Н.В., Бортник К.Я., Варченко Л.Л. Моніторингова система для операційної системи Windows. У даній статті розглядається створення моніторингової системи для операційної системи Windows та специфіка її використання.
Ключові слова: Windows, Grafana, моніторингова система, операційна система, сервіс.

Bahniuk N.V., Bortnyk K.Y., Varchenko L.L. Monitoring system with Windows operating system. This article discusses about monitoring Windows orerating system and specific things to solve in process.
Keywords: Windows, Grafana, monitoring system, operating system, service.

Постановка проблеми та аналіз досліджень. В даній статті описано створений альтернативний програмний комплекс для відслідковування та аналізування ресурсів, які задіяні під час роботи операційної системи Windows. Всі антивіруси грунтуються суто на хеш-сумах виконуваних файлів, в той же як моніторингова система надасть можливість проаналізувати поведінку зловмисного програмного забезепечення або будь-якого іншого.

Так як Windows є найпоширенішою операційною системою, то питання її безпеки є критично важливим. Велика популярність даної системи пояснює кількість різних зловмисних програм, що орієнтуються не тільки на безпосередньо пряму атаку та викрадення, а й приховування своєї присутності для нанесення якнайбільшої шкоди, збитків для користувача. Тому превентивні методи визначення шкідливого ПЗ є актуальною темою на даний час та в майбутньому.

Виділення невирішених раніше частин загальної проблеми

Створюваний програмний комплекс орієнтований на клієнт-серверну архітектру, а саме - створення інтерфейсу для програм та кінцевих кристувачів. Причини вибору такого принципу є подальша можливість розробити масштабовану систему не тільки для одного юзера, а й цілої системи для моніторингу групи пов'язаних між собою систем. Для систем типу UNIX більшість шкідливих програм користуються принципом неавторизованого доступу до системи. Тобто, для них притаманне використання доступу через програми або "дірки" в коді, які поки що існують в системі. В Windows ситуація зводиться до комплексів, які не тільки намагаються отримати неавторизований доступ через програми, а намагаються використати безпосередньо користувача як "винуватця" зараження. Тим і ця тема є цікава для розгляду. Тому антивірусні програми часто безсилі – користувач самостійно надає програмі адміністративний доступ, що і є причиною його проникнення. [4]

Формулювання мети дослідження

Одним з найважливіших етапів у розробці ПЗ є, власне, опрацювання структури програми, визначення архітектурних рішень і т.д. Основною ідеєю під час етапу проектування є розробка системи, яка легко масштабується, легко тестується, зберігає можливість як серверного так і локального розташування.

Для операційної системи Windows не так легко встановити моніторингову систему через деякі особливості роботи, тому для розробки використовуються Prometheus, Node Exporter і Grafana.

Виклад основного матеріалу дослідження

Prometheus – головна частина. Prometheus збирає показники з кількох служб і отримує їх в одному місці.

Проста програма під назвою Node Exporter збирає показники операційної системи та пропонує до них HTTP-доступ. Prometheus збирає дані з одного або кількох екземплярів Node Exporter.

Grafana представляє інформаційні панелі з графіками та діаграмами, які містять дані з Prometheus.

Для забезпечення вищезазначених параметрів використовується віртуалізація програм в різні процеси. Для цього використовується різні докер-контейнери, які будуть запущені для кожних компонентів програми як унікальний контейнер (рис. 1).

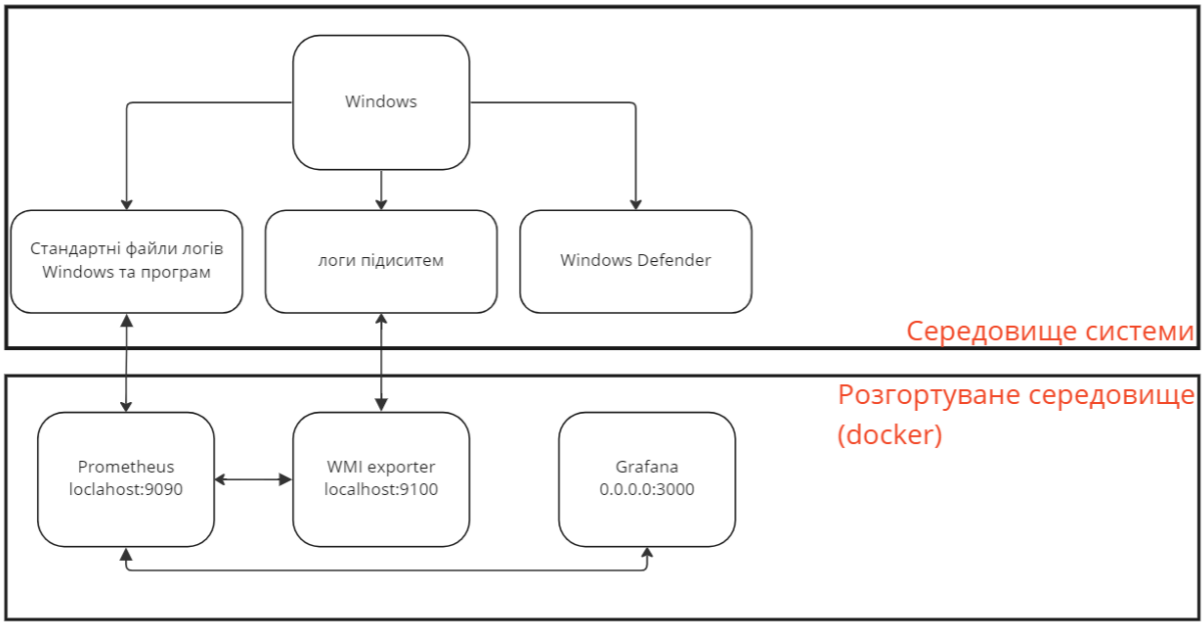

Рисунок 1 – Структурна схема програми

З рисунка 1 видно загальні налаштування сервісів. Кожен контейнер має свої налаштування, різні версії програм, "образи" та попередні команди у вигляді скрипт-файлів для підготовки контейнеру до роботи.

Prometheus. Образ "prom/prometheus" при установці "закидає" в контейнер файл prometheus.yml та використуває локальну папку prometheus_data спільно з хостом. При запуску використовує певні підготовчі команди та відкриває в мережі порт 9090.

Grafana. Завдяки команді depends_on, образ "grafana/grafana" під час встановлення чекає запуску контейнеру з prometheus. Після цього в контейнер надсилає файли налаштувань з папки grafana/provisioning/ та використовує локальну папку grafana_storage спільно з хостом. При запуску використовує певні підготовчі команди та відкриває на хості порт 3000.

Розглянему структурну схему програми (рис. 1), де зображено взаємозв'язки між сервісами.

Для зручної розробки ПЗ моніторингу, інструменти віртуалізації є необхідністю. У випадку розробки цим інструментом є docker. Docker – спеціальне середовище віртуалізації. "Середовищем" виступають невеликі або повноцінні операційні системи, які запускаються в середині віртуалізації, та можуть бути пов'язані між собою за допомогою внутрішньої, недоступної ззовні мережі. Такий підхід надає декілька переваг: відслідковування роботи програми, побудова клієнт-серверної системи, можливість швидкого запуску програм, копіювання усіх логів та дій програм, обмеження ресурсів споживання.

Prometheus. ПЗ, що записує показники в режимі реального часу в базу даних часових рядів (що забезпечує високу розмірність), створену з використанням моделі HTTP pull, із гнучкими запитами та сповіщеннями в реальному часі.

Проект написаний на Go та ліцензований за ліцензією Apache 2, з вихідним кодом, доступним на GitHub, і є випускним проектом Cloud Native Computing Foundation разом з Kubernetes і Envoy. Prometheus збирає та зберігає свої метрики як дані часових рядів, що означає, що інформація про показники зберігається разом із міткою часу, коли вона була отримана, і не обов'язковими парами ключ-значення, відомими як мітки. [1] Добре працює для запису будь-яких чисто числових часових рядів. Він підходить як для машинного моніторингу, так і для моніторингу високодинамічних сервіс-орієнтованих архітектур. У світі мікросервісів його підтримка збору багатовимірних даних і запитів є особливою перевагою. [5]

Одного інструменту тільки для збору даних мало, додатковим, але не обов'язковим, інструментом є візуалізатор даних. Для означених раніше цілей найкраще підходить інструмент візуалізації Grafana.

Програмним забезпечення з відкритим кодом Grafana можна запитувати, переглядати, сповіщати та аналізувати свої показники, журнали та трасування незалежно від того, де вони зберігаються.

Також можна використовувати інструменти, надані Grafana OSS, щоб створювати графіки та візуалізації з даних у базі даних часових рядів (рис. 2). Після налаштування першої інформаційної панелі залежно від потреб, є широкий вибір альтернатив, коли вперше використовується Grafana. Можна створити список відтворення, наприклад, щоб відстежувати інформацію про погоду та статистику щодо розумного будинку. Можна налаштувати надання та автентифікацію, якщо адміністратор підприємства керує Grafana для різних команд. [3]

Допоміжну роль виконує сервіс експортер WMI (рис. 3). Цілі – це вузли, до яких Prometheus має доступ і які показують метрики за певною URL-адресою. Такі цілі постачаються як «експортери», які, насправді є двійковими файлами, які працюють з ціллю та відповідають за збір показників про хост. Запуск « Node Exporter », який відповідає за отримання статистичних даних про використання ЦП або дискового вводу/виводу, який зараз використовується – це спосіб моніторингу системи Linux. Експортер WMI відповідатиме за збір статистики про систему та працюватиме як служба Windows. [6]

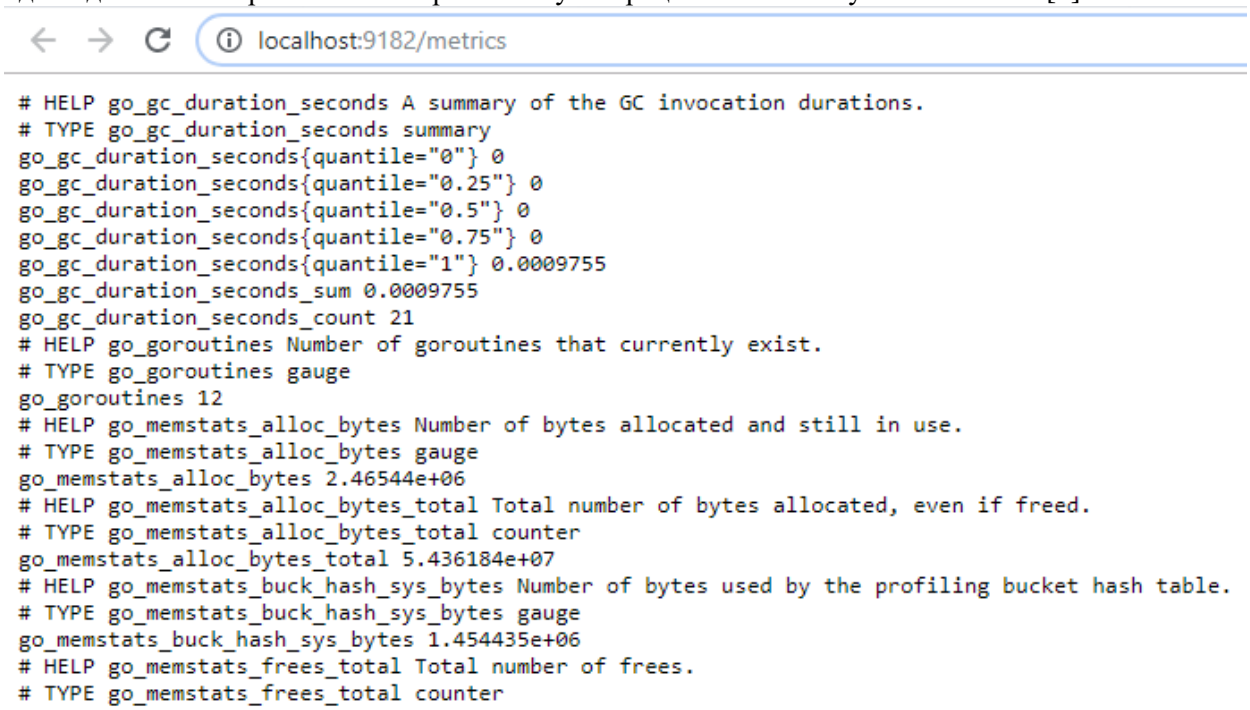

Рисунок 3 – Wmi exporter

Таке поєднання програм дозволить забезпечити не тільки зручність у роботі, а й можливу масштабованість проекту. Масштабованість забезпечена за рахунок мультиопераційності програм та легкості їх поєднання. Означені технології розробки дозволять якнайефективніше розподілити розробку самої моніторингової системи та визначити отримані результати вже на ранніх етапах розробки.

Перш ніж переходити до інструментів Prometheus, важливо зрозуміти цю модель даних повністю. У Prometheus використовуються пари ключ-значення. Значення зберігає фактичне значення як число, тоді як ключ визначає, що ми вимірюємо. Prometheus не призначений для зберігання неструктурованих даних, таких як звичайний текст. Він зберігає вимірювання, які були зібрані протягом усього часу.

Звідси маємо перший і наголовніший маркер – як змінюється використання ресурсів операційною системою різними програмами під час роботи в середовищі, де існує зловмисна програма. Визначення "зміна використання ресурсів" грунтується на припущенні, що під час зараження комп'ютер (представлений фізично) та операційна система (представлена в значенні ядра) гарантовано повинні змінити кількість ресурсів, які будуть використовуватися до аномальної межі – це є маркер фізичного стану. За даним припущенням спостерігає дашборд Win_phys_state та Win_stats. [8]

Win_phys_state – дашборд Grafana, призначений для забору фізичних метрик споживання ресурсів. В даному відображенні міститься інформація суто про стан фізичних властивостей пристроїв, які входять до даного комп'ютера, такі як:

- для процессора: температура центрального процесора, частота роботи, кількість зайнятих фізичних ядер і тд;
- оперативна пам'ять: кількість завантаженої пам'яті, швидкість;
- диск: тип, швидкість читання/писання, середня швидкість відгуку.

Для даного дашборда, в сервісі Win Exporter використовуються спеціальні колектори. Для кожного роду даних існують спеціальні колектори, відповідно до вище описаних властивостей.

Win_stats призначений для відслідкування системних метрик, які будуть важливим маркером не тільки для визначення зловмисного ПЗ, а й при відслідковуванні стану операційної системи, як такої. Метрики, що входять до даного дашборду:

- за процесами: час роботи системи, загальна кількість процесів, кількість дескрипторів, кількість відкритих потоків;
- використанння диску: графік активного часу, графік обміну даних, коефіцієнт середньої завантаженості;
- використання мережі різними процесами в момент часу.

Рисунок 4 – Дашборд Win_phys_state

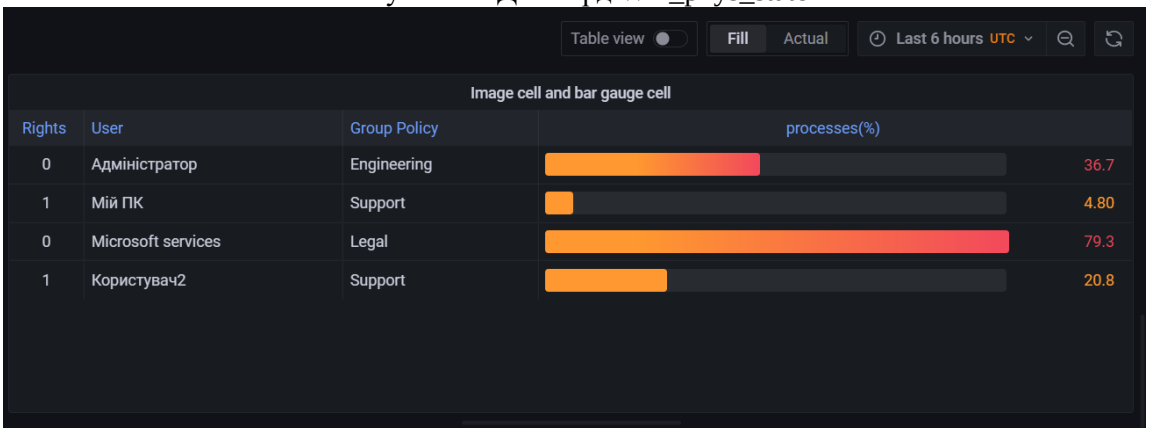

Рисунок 5 – Метрики запусу користувача (дашборд Win_user_info)

На основі даного дашборду отримуємо загальну інформацію про ресурси, які можуть бути задіяні системою, і чітко можна отримати інформацію, чи дійсно надлишок ресурсів використовуються в даний момент. Даний дашборд є загальною інформацією про стан процесів та відповідних служб в момент часу.

Після визначення першого меркеру потрібно зосередити увагу на конкретних маркерах поведінки саме операційної системи та запущених програм. В даному випадку основними метриками є: споживання ресурсів програмами (Prog_user_stats), споживання підсистеми Windows (Win_sys_stats), ресурси Windows програм (Win_exe_stats) та користувачі і політики безпеки (Win_user_info). Дані метрики супроводжують використання русурсів процесами. З даного дашборду можна зробити висновок про аномальну поведінку окремого процесу, відірваного від загальної системи Windows.

Під час запуску процесів, належних до користувачів, ми отримуємо можливість прослідкувати за викликаним процесом, отримати ресурси, які він використовує (дочірні процеси та їхні привілеї). Далі можемо прослідкувати сервіси, які використовує процес (так як при виклику сервісу ми отримаємо те,

що один з пов'язаних процесів почав використовувати відповідні сервіси). І в кінці кінців це і становить перехід з дашборду Win_user_info до Win_exe_stats і згодом в Win_sys_stats.

Дашборд Win_sys_stats – комплекс відслідковування сервісів та підсистеми Windows. Даний інтерфейс є по суті списком сервісів та підсистем з графіком використання процесами. Даний графік надасть результати про надмірне споживання будь-яким процесом ресурсів ядра через сервіси, на графіку це буде чітко видно. Також вартий уваги дашборд Win_TCP_IP, призначений для проглядання споживання мережі процесами. Даний дашборд надає просту інфографіку про загальне користування та процеси на фоні загальної. [7]

Для визначення та дослідження ресурсів безперечно важливим є грамотне розподілення всіх метрик рівномірно по всім дашбордам. Звісно, що для повної інформативності метрик неможливо не повторюватися, але це тільки заради реального моніторингу системи. Основною задачею є дослідження ресурсів операційної системи Windows, але в проміжному результаті очікується насемперед рівномірний розвиток системи, адже проміжні результати будуть надавати все більше і більше інформації, яку необхідно враховувати в подальших дашбордах. Такий розвиток буде точно виділений, причиною цього однозначно буде складність операційної системи.

Дана моніторингова система має можливість отримати в майбутньому наступний маркер – функцію моніторингу стану обладнання та операційної системи, забезпечення інформування про аномальну зміну в роботі обладнання та рекомендацій щодо його обслуговування. Також постійно відслідковуються окремі процеси Windows, які пов'язані з оновленням та запуском деяких служб, список яких можна отримати з панелі управління службами або в реєстрі (regedit). Ці сервіси та служби є вже не такі пріоритетні як ті, що вказані на автозапуск, тому під час запуску системи вона деякий час "підвисає" - додаткові системні служби, служби та сервіси перевірок починають виконувати звичайну системну роботу. І різниця між цими службами тільки в тому, що деякі мають звичайний системний характер, а саме – якщо це перезапуск, ці служби не будуть проводити ці процедури.

Щодо оцінки працездатності моніторингової системи – основне можна побачити з дашборду Monitor_health_host (рис. 6). На ньому видно, що частка ресурсу, яким користується програма хост є дуже мала, навіть для комп'ютерної системи з обмеженим ресурсом.

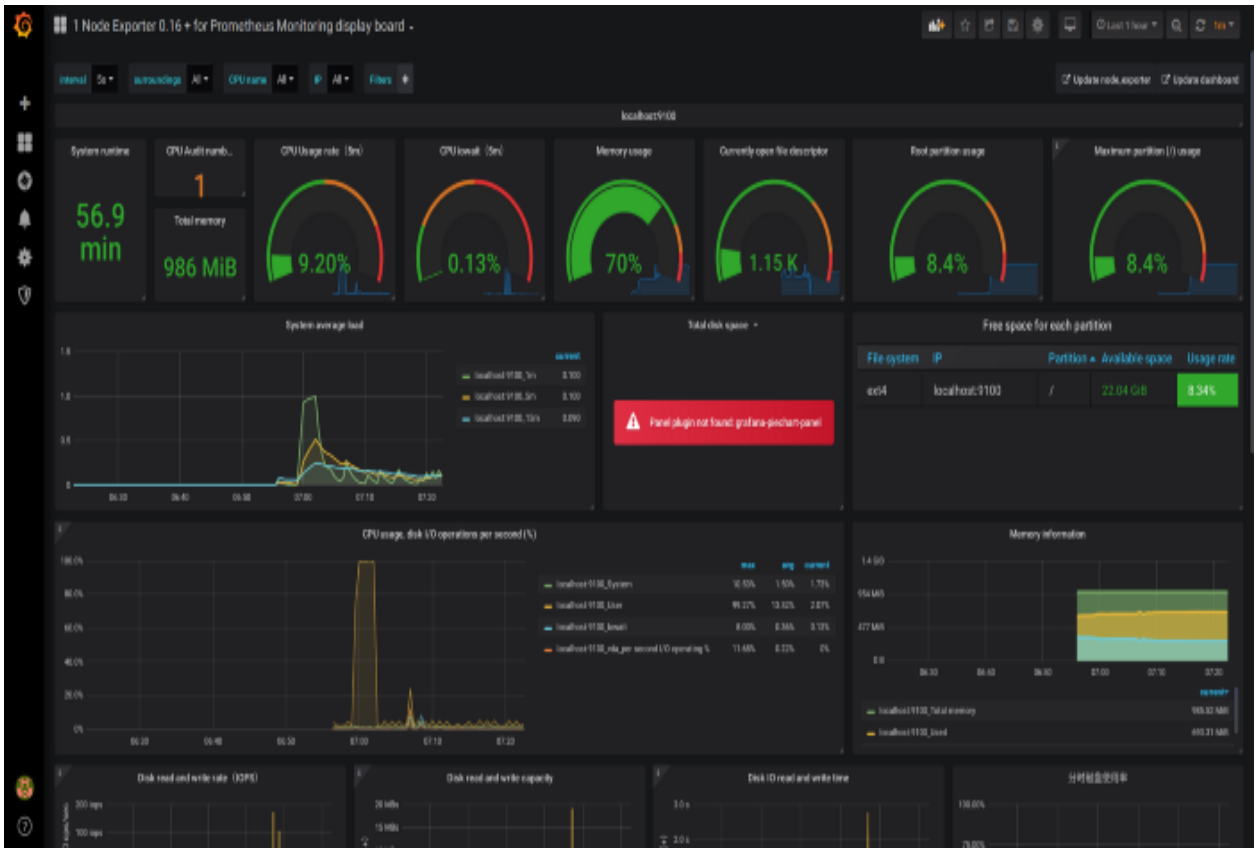

Рисунок 6 – Моніторинг стану сервісів, які складають моніторинг-систему

Висновки та перспективи подальших досліджень

Даний програмний комплекс надає змогу отримувати інформацію про стан операційної системи. Поданий у статті порядок програм забезпечує всі покладені на них задачі та витрачається мала кількість ресурсів, що є важливим для ефективної роботи системи, яку піддають моніторингу. Дашборди, створені для даного програмного забезпечення, повністю відображають дійсний стан ресурсів операційної системи.

Дану розробку можна використовувати не тільки для виявлення шкідливого програмного забазпечення, а й для моніторингу системи. Тобто, дані результати також можуть допомогти отримати результати щодо технічного стану апаратної частини ПК. Також на основі цих даних отримано результати щодо стану операційної системи: швидкість відгуку програм, коректність роботи ядра ОС, стан системних файлів.

Подальші дослідження по цій темі дозволять створити прициндент для наступних досліжень в даній області. Всі дослідження в даній темі дадуть розвиток програмних комплексів, які будуть займатися виявленням зловмисного програмного забезпечення в атоматизованій системі моніторингу.

Список бібліографічного опису

1. Bastos J. Hands-On Infrastructure Monitoring with Prometheus: Implement and scale queries, dashboards, and alerting across machines and containers / J. Bastos, P. Araújo.
2. Docker Desktop | Docker Documentation [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://docs.docker.com/desktop/.
3. Documentation | Grafana Labs [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://grafana.com/docs/.
4. Microsoft security documentation - Security documentation [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/security/.
5. Overview | Prometheus [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://prometheus.io/docs/introduction/overview/.
6. prometheus-community/windows_exporter: Prometheus exporter for Windows machines [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/prometheus-community/windows_exporter#readme.
7. Russinovich M., Margosis A. Troubleshooting with the Windows Sysinternals tools. Redmond : Microsoft, 2015.
8. Salituro E. Learn Grafana 7.0: A beginner's guide to getting well versed in analytics, interactive dashboards, and monitoring / Eric Salituro.. – (1st Edition, Kindle Edition).

References

1. Bastos J. Hands-On Infrastructure Monitoring with Prometheus: Implement and scale queries, dashboards, and alerting across machines and containers / J. Bastos, P. Araújo.
2. Docker Desktop | Docker Documentation [Electronic resource]. – 2022. – URL: https://docs.docker.com/desktop/.
3. Documentation | Grafana Labs [Electronic resource]. – 2022. – URL: https://grafana.com/docs/.
4. Microsoft security documentation - Security documentation [Electronic resource]. – 2022. – URL: https://learn.microsoft.com/uk-ua/security/.
5. Overview | Prometheus [Electronic resource]. – 2022. – URL: https://prometheus.io/docs/introduction/overview/.
6. prometheus-community/windows_exporter: Prometheus exporter for Windows machines [Electronic resource]. – 2022. – URL: https://github.com/prometheus-community/windows_exporter#readme.
7. Russinovich M., Margosis A. Troubleshooting with the Windows Sysinternals tools. Redmond : Microsoft, 2015.
8. Salituro E. Learn Grafana 7.0: A beginner's guide to getting well versed in analytics, interactive dashboards, and monitoring / Eric Salituro.. – (1st Edition, Kindle Edition).