Київський національний університет імені Тараса Шевченка
Факультету радіофізики, електроніки та комп’ютерних систем
Кафедра комп’ютерної інженерії

АВТОМАТИЗАЦІЯ АУДИТУ АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ ПЕРРСОНАЛЬНИХ КОМП’ЮТЕРІВ У ЛОКАЛЬНІЙ МЕРЕЖІ

Дипломна робота магістра
студента 2-го року навчання
спеціальність: 123 «Комп'ютерна інженерія»
Юрія ВЕРЕМІЯ

Науковий керівник
канд. фіз.-мат. наук Юрій БОЙКО,
доцент кафедри комп’ютерної інженерії

Рецензент
доктор фіз.-мат. наук Євген ІВОХІН,
доцент кафедри системного аналізу
та теорії прийняття рішень

До захисту допускаю: Завідувач кафедрою Юрій БОЙКО
Ухвалено на засіданні кафедри “____” _____ 2022 р., протокол № ___

Київ - 2022

РЕФЕРАТ

Робота складається з трьох частин. Перша частина містить розгляд проблеми аудиту апаратного забезпечення в локальній мережі та послідовності її вирішення. Опис технологій що можуть бути використані як онова для розробки рішення обліку та аудиту. Друга частина містить розгляд та порівняння існуючих програм, що вирішують поставлену задачу. Проводиться перевірка роботи існуючого рішення. У третій частині вибрано технології на основі яких буде розроблено рішення автоматичного аудиту. На їх базі розроблено нове рішення. Розроблений програмний продукт розглянено у форматі покрокової інструкції встановлення та запуску. Проведено перевірку роботи розробленого програмного комплексу, що спрощено можна описати наступними кроками:

а. попередній збір інформації про операційні системи об’єктів обліку;
б. збір інформації з використанням вибраної системи оркестрування;
в. експорт результатів у Google документи;
г. повторне проведення пунктів б. і в. та проведення аудиту шляхом порівняння результатів.

Випускна дипломна робота магістра містить 57 сторінок, 23 рисунка, 2 таблиці, 7 додатків, використано 26 інформаційних джерел.

Ключові слова: ОБЛІК, АУДИТ, ІНВЕНТАРИЗАЦІЯ, OCS INVENTORY, ANSIBLE, NMAP, PYTHON, SSH, WINRM, GOOGLE SHEET.

ЗМІСТ

РЕФЕРАТ 2
ЗМІСТ 3
СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ 5
ВСТУП 6
РОЗДІЛ 1. РОГЛЯД ПРОБЛЕМИ АТОМАТИЧНОГО АУДИТУ ТА ТЕХНОЛОГІЙ, ЩО МОЖУТЬ БУТИ ОСНОВОЮ ДЛЯ СТВОРЕННЯ ЇЇ РІШЕННЯ 8
1.2. Виникнення та визначення проблеми автоматичного аудиту 8
1.2. Технічні вимоги до сервісу автоматичного аудиту 9
1.3. Послідовність операцій необхідна для проведення аудиту 10
1.4. Інструментарій який можна використати як базу для подальшого створення рішення аудиту 11
1.4.1. Програми, що збирають інформацію про систему 12
1.4.1.1. Приклад: AIDA64 12
1.4.2. Intel vPro 13
1.4.3. Системи моніторингу 13
1.4.4. Системи оркестрування 14
РОЗДІЛ 2. РОЗГЛЯД ТА ПОРІВНЯННЯ ІСНУЮЧИХ РІШЕНЬ 15
2.1. Існуючі реалізації, що виконують аудит програмного забезпечення 15
2.1.1. Total Network Inventory 15
2.1.2. Checkcfg 15
2.1.3. Network Inventory Advisor 16
2.1.4. Spiceworks 16
2.1.5 OCS Inventory 16
2.1.6 Існуючі реалізації: Висновок 17
2.2. Розгляд робити одного з описаних рішень 18
2.2.1 Встановлення та налаштування OCS Inventory 19
2.2.2. Встановлення агента на Windows 20

2.2.3. Встановлення агента на Linux 22
2.2.4. Розгортання сервісу: Висновок 23
РОЗДІЛ 3. РОЗРОБКА ТА ПЕРЕВІРКА РОБОТИ РІШЕННЯ АВТОМАТИЧНОГО АУДИТУ 24
3.1. Вибір програми, що забезпечить взаємодію компонентів програмного продукту 25
3.1.1. Вибір системи оркестрування 25
3.2. Вибір технологій для збору та обробки інформації, а також середовища в якому працюватиме програмний продукт 26
3.3 Розгортання створеного програмного продукту та перевірка роботи 26
3.3.1. Встановлення та налаштування Ansible 26
3.3.2. Підключення Google Sheets API v4 28
3.3.3. Огляд тестового середовища використаного для перевірки функціоналу розробленого програмного продукту 29
3.3.4. Огляд програмного продукту перед запуском 30
3.3.5 Запуск рішення для проведення обліку 31
3.3.6 Перевірка правильності роботи аудиту 36
ВИСНОВКИ 41
ПЕРЕЛІК ПОСИЛАНЬ 42
ДОДАТКИ 45
Додаток А. Вміст файлу ansible_work.sh 45
Додаток В. Вміст файлу basic_check.py 49
Додаток С. Вміст файлу fill_invent_sheet.py 51
Додаток D. Вміст файлу fill_main.py 54
Додаток Е Вміст файлу new_spreadsheet.py 56
Додаток F. Вміст файлу OS_sort 58
Додаток G Вміст файлу program.sh 59

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

OS — **O**perating **S**ystem — операційна система;

БД — **Б**аза **Д**аних є організованою структурою призначеною для зберігання даних;

WinRM — **Win**dows **R**emote **M**anagement — віддалене управління Windows. Служба для віддаленого управління та конфігурування Windows;

SSH — **S**ecure **Sh**ell — безпечна оболонка. Мережевий протокол четвертого рівня моделі OSI, що дозволяє безпечно проводити віддалене управління та конфігурування комп'ютерів;

API — **A**pplication **P**rogramming **I**nterface — сукупність методів, протоколів, технологій чи/та програм для швидкої розробки програмного забезпечення.

CPU — **C**entral **P**rocessing **U**nit — центральний процесор;

RAM — **R**andom **A**ccess **M**emory — оперативна пам’ять;

Proprietary — Власницьке програмне забезпечення — програмне забезпечення, що знаходиться у приватній власності, здебільшого є платним та сирцевий код якого не доступний;

Freeware — безкоштовне програмне забезпечення з закритим кодом;

Shareware — умовно безкоштовні програми з закритим кодом.

Open-source software — безкоштовне програмне забезпечення з відкритим кодом.

SNMP — **S**imple **N**etwork **M**anagement **P**rotocol — простий протокол керування мережею, що покликаний забезпечити моніторинг та керування в мережі між агентом та керуючою системо.

WMI — **W**indows **M**anagement **I**nstrumentation — інструментарій керування Windows є однією з технологій для централізованого керування та стеження за роботою різних частин комп'ютерної інфраструктури Windows.

ВСТУП

Оцінка сучасного стану об’єкта розробки. Існує більше дюжини програм що виконують автоматичний аудит апаратного забезпечення. Історично завдання виникало окремо в кожній компанії по мірі її росту, що призводило до того, що компанії вирішували її самостійно в рамках їх же потреб. Пізніше з’явились рішення розроблені окремими компаніями, проте вони все ж здебільшого були вузько направленими — наприклад тільки для Windows. Всі знайдені та розглянуті програми мають свої недоліки.

Серед недоліків виявлених у розглянутих рішеннях можна виокремити наступні: використання клієнтів, програмне забезпечення не є вільним, відсутність інтеграції з хмарними сховищами, відсутня кросплатформність, та інші.

Актуальність роботи та підстави для її виконання. Існує необхідність у створенні максимально універсального рішення, яке б вирішувало більшість проблем наявних у конкурентів на ринку.

Метою кваліфікаційної роботи є дослідження підходів до автоматичного аудиту апаратного забезпечення, та пізніше створення програмного засобу для аудиту, що вирішує знайдені проблеми — програмний комплекс з відкритим вихідним кодом на базі відкритих продуктів, що не використовує встановлення програм (клієнтів) та експортує результати в хмарні середовища (Google Sheets). Для досягнення цієї мети поставлено такі завдання:

- дослідити існуючі рішення, що виконують аудит;
- дослідити технології на базі яких можливо розробити рішення аудиту;
- розробити програмний продукт.

Об’єктом розроблення програмного засобу є процес автоматичного аудиту персональних комп'ютерів за допомогою відкритих програмних засобів. Предметом роботи є програмний засіб для аудиту персональних комп’ютерів з можливістю завантаження результатів в хмарне сховище.

Розробці програмного засобу передувало дослідження наявних на ринку рішень, їхнього інтерфейсу та результатів виконання. Особливо пильно було проведено аналіз програми OCS Inventory Version 2.7.

Інструменти розроблення: в якості операційної системи, що буде надавати послуги автоматичного аудиту було обрано Ubuntu 18.04.4 LTS. В якості інструменту створення програмного засобу — Ansible v2.4 — безкоштовне, вільно поширюване програмне забезпечення, що надає засоби для управління конфігурацією, оркестровки, централізованої установки застосунків і паралельного виконання типових завдань на групі систем. Використовувались такі мови програмування: python – мова, що використовувалась для взаємодії з Google Sheets, bash – вбудована в операційну систему мова, що дозволяє зручно виконувати проміжні задачі в процесі роботи програми.

Результати роботи: виконано загальний огляд наявних електронних засобів аудиту, проаналізовано їх переваги та недоліки, досліджено програми для розроблення засобів аудиту, розроблено програмний продукт, який дозволяє проводити аудит в локальних мережах.

Можливими сферами застосування є використання програмного продукту в локальних мережах на суб’єкті Linux (або віртуальній машині з Linux, так як рішення не вимагає значних ресурсів для роботи) та об’єктах — персональних комп’ютерах з сучасними операційними системами.

РОЗДІЛ 1. РОГЛЯД ПРОБЛЕМИ АТОМАТИЧНОГО АУДИТУ ТА ТЕХНОЛОГІЙ, ЩО МОЖУТЬ БУТИ ОСНОВОЮ ДЛЯ СТВОРЕННЯ ЇЇ ВИРІШЕННЯ

В даному розділі розглядається задача автоматичного аудиту, а саме технології та методи, які зазвичай використовуються для її вирішення. Наводиться конкретний набір технологій, що можуть бути використані для розробки нового рішення.

1.2. Виникнення та визначення проблеми автоматичного аудиту

З середини XX століття і до сьогоднішнього дня відбувається активний розвиток комп’ютерних технологій. Кількість комп’ютерів підключених до мережі налічує понад 7 мільярдів (якщо враховувати не тільки комп’ютери а й інші пристрої – 17 мільярдів) **Помилка! Джерело посилання не знайдено.**.

У сьогоднішніх реаліях досить часто одній людині доводиться обслуговувати більше ніж один комп’ютер. Нехай існує підприємство яке орендує певне приміщення і розміщує там свої комп’ютери. До даного приміщення мають доступ сторонні особи, що можуть користуватись розміщеними в переміщені комп’ютерами. В такому випадку обслуговування та моніторинг працездатності компонентів комп’ютерів перетворюється на рутину — потрібно підійти до кожного пристрою та власноруч зібрати всю необхідну інформацію. Підприємству потрібен інструмент, що дозволить спостерігати за станом комп'ютерів та за компонентами, що входять до їх складу і за потреби порівнювати результати спостережень з певним еталонним зразком. Більше того, можливі випадки коли потрібно інвентаризувати пристрої, які не мають підключених інтерфейсів вводу/виводу, або взагалі знаходяться у недоступних місцях.

Для опису такого функціоналу на жаль немає чітко визначеного терміну. Терміни близькі за значенням: «аудит», «інвентаризація», «облік», «інспекція».

Термін «аудит» підходить найкраще, проте є суттєві проблеми — під час першого запуску програми новим користувачем ще немає певного еталону для перевірки, через, що використання терміну «аудит» є фактично неправильно. Крім того певна частина програм розглянутих в цій роботі не проводять аудит — вони тільки збирають потрібну інформацію, що аудитом очевидно не є. Тому для першого запуску програми, а також, якщо програма використовується тільки для збору інформації надалі буде використовуватись термін «облік». Узагальнюючи в рамках даної роботи будуть використовуватись терміни:

- облік — це збір інформації про обладнання (hardware) та програмне забезпечення (software) комп'ютерів;
- аудит — це виконання нового обліку та порівняння результатів попередніми обліками;
- автоматичний аудит — це проведення аудиту віддалено використовуючи програму встановлену на об'єкті аудиту, або використовуючи протоколи віддаленого доступу.

1.2. Технічні вимоги до сервісу автоматичного аудиту

В даній роботі ідеальним рішенням аудиту вважається рішення з такими характеристиками:

- розповсюдження за моделлю open source;
- розміщення результатів роботи в хмарному сховищі (наприклад у Google Sheets**);**

- зручний, інтуїтивно доступний інтерфейс, деревоподібна структура та невелика кількість елементів керування у кожному з блоків інтерфейсу та простота встановлення;
- кросплатформність програми (можливість встановити програму на різні сучасні OS);
- кросплатформеність клієнтів (можливість проведення обліку клієнтів з різними сучасними OS);
- проведення аудиту;
- відсутність клієнтів, що встановлюються на об’єкт обліку.

Якщо не буде знайдено програми, що задовольняє всім вище наведеним пунктам, надалі у цій роботі буде розроблено рішення, що покриває якомога більшу кількість наведених критеріїв.

1.3. Послідовність операцій необхідна для проведення аудиту

Перш за все, одразу після того, як було підключено до мережі сервер з установленою програмою автоматичного обліку, має бути проведено сканування мережі для виявлення підключених пристроїв та проведено неповний облік – зібрати базову інформацію про пристрої щоб визначитись з подальшим набором команд (повинна бути дозволена робота відповідних протоколів). Перелік засобів яким можна це виконати:

- провести ping-сканування пристроїв;
- перевірка на відкритість найпопулярніших TCP-портів;
- перевірка працездатності протоколів, що працюють на UDP;
- проведення ARP-сканування мережі для виявлення пристроїв;
- опитування по SNMP протоколів CDP/LLDP, за відповідної наявності агентів протоколу;
- NetBIOS сканування, або схожі реалізації інших поставників (працює тільки для Windows);

- переглянути останні журнали DHCP-сервера, в яких зберігається інформація про пристрої в мережі;
- запустити ARPWatch, передавати на нього копію трафіку всієї мережі;
- проаналізувати Forwarding DataBase таблиці комутаторів мережі [2].

Наступним кроком буде проведення обліку на базі інформації отриманої, в попередньому пункті. Перелік варіантів як це можна зробити:

- зайти на сервер з допомогою протоколів віддаленого доступу (залежить від операційної системи, наприклад для Linux — це SSH) та виконати команди (або набори команд), що повернуть потрібну інформацію (наприклад systeminfo на Windows). Передати її на сервер, вийти з сесії віддаленого доступу;
- використати протокол SNMP (або інші які реалізують схожий функціонал, наприклад WBEM). На об’єктах обліку налаштувати відповідний клієнт. За допомогою команд отримати інформацію про комп’ютер;
- встановити на об’єкта обліку програму, що буде збирати всю потрібну інформацію та передавати її на сервер.

Окрім цього потрібна БД або інший варіант збереження інформації зібраної після обліку. У випадку якщо облік уже проводився потрібно щоб програма порівняла результат нового обліку з еталоном і у випадку розбіжностей повідомляє про них адміністратору, себто проводить аудит.

1.4. Інструментарій який можна використати як базу для подальшого створення рішення аудиту

Так як розробляти проект з нуля часто дуже важко, непоганою ідеєю є використання сторонніх програм, що не писались безпосередньо для вирішення поставленої задачі, тим не менш мають функціонал, що дозволяє їх використати саме цього. Далі будуть розглянуті програми дбайливо категоризовані

відповідно до принципів роботи, що можуть за певних умов використовуватись для розробки рішення аудиту.

1.4.1. Програми, що збирають інформацію про систему

Ці програми встановлюються на клієнта та збирають необхідну інформацію. Принцип роботи таких програм відрізняється між кожною з них та залежить від операційної системи. Їх можна використати не зважаючи на принцип роботи. Базою ідеї є зібрати інформацію та пізніше певними доопрацьованими методами змусити їх передавати цю інформацію на суб'єкт обліку. Приклад того, як це можна реалізувати буде наведено далі.

До таких програм можна віднести:

- AIDA64
- HWMonitor
- CPU-Z
- Open Hardware Monitor
- Hardinfo
- та ін.

1.4.1.1. Приклад: AIDA64

AIDA64- закрите платне програмне забезпечення з закритим кодом створене компанією FinalWire Ltd.

AIDA64 Network Audit Edition дозволяє віддалено збирати інформацію про комп'ютери в мережі. При цьому на кожному комп'ютері повинна бути встановлена версія програми, що підтримує облік. Також необхідно щоб було включено вхідні віддалені підключення.

Як варіант, враховуючи портативність застосунку, можна використовувати файловий сервер з програмою на ньому. Таке рішення вимагає додатково написання програм, що буть запускати AIDA64 на кожному, окремо взятому комп'ютері. Докладний звіт по кожному комп'ютері в мережі може бути записаний у файл формату CSV, XML або SQL-базу., що породжує про-

блему пізнішого переформатування отриманої інформації у єдиний звіт, що знову ж таки потребує вирішення з допомогою сторонніх програм [17].

1.4.2. Intel vPro

Intel vPro являє собою апаратно-програмний комплекс розроблений компанією Intel. Ця технологія дозволяє використовувати віддалений доступ до ПК (в тому числі моніторингу, обслуговування та управління) незалежно від стану операційної системи (ОС) або стану живлення ПК і функції захисту. Для використання потрібно написати програму, що буде реалізовувати опитування всіх клієнтів, зберігати інформацію та передавати її далі в інші програми [18].

Використання цього рішення має суттєву проблему — воно буде працювати тільки на процесорах, що очевидно, де воно наявне.

1.4.3. Системи моніторингу

Для збору інформації можна використати спеціалізовані протоколи та відповідні програми, що реалізовані для моніторингу, які як і у випадку 1.5.1 можна використати для аудиту. До них можна віднести:

- SNMP;
- WMI.

Ідея використання програм моніторингу досить проста — в них уже реалізовано можливість доступу до клієнтів. Залишається тільки реалізувати збір потрібної інформації, проведення аудиту та передачу даних у хмарне середовище. Проблемою є архітектура такого рішення так я тоді воно буде або клієнт серверним не залежно від супровідного програмного забезпечення, або виключно для однієї операційної системи.

Перелік технологій, що тоді доведеться використовувати:

- Net-SNMP
- Zabbix

- Check_MK
- Nagios
- Icinga
- OpenNMS
- Pandora FMS
- Та ін.

1.4.4. Системи оркестрування

Ідея використання систем оркестрування дуже схожа до наведеної вище, проте є і відмінності. Дані програми зазвичай підтримують ширший набір операційних систем. Частина з систем оркестрування не використовує агентів для роботи. Перелік систем оркестрування:

- Puppet
- Ansible
- Chef
- Spacewalk
- SaltStack
- Juju
- SmartFrog
- Та ін.

РОЗДІЛ 2. РОЗГЛЯД ТА ПОРІВНЯННЯ ІСНУЮЧИХ РІШЕНЬ

Перед тим як приступити до створення власного рішення аудиту варто знати про інші реалізації що вирішують завдання автоматичного аудиту. Тому в даному розділі проводиться порівняння існуючих реалізацій та перевірка роботи однієї з них.

2.1. Існуючі реалізації, що виконують аудит програмного забезпечення

Далі будуть розглянуті найбільш популярні програми, що вже реалізують функцію обліку та аудиту(надалі використовувався опис з офіційних сайтів програм або з сайтів, що продають чи розповсюджують ці програми).

2.1.1. Total Network Inventory

Total Network Inventory [3] — це програма розроблена, компанією Softinventive Lab. Встановлюється лише на Windows, проте сканує комп’ютери на базі Windows, OS X, Linux, FreeBSD і ESX / ESXi без використання встановлених агентів – для роботи потрібно знати пароль адміністратора. Крім того програма реалізує такі функції як запланований облік та можливість слідкувати за станом комп’ютерів в реальному часі., що до мінусів, ця програма з закритим кодом та постачається платно, також в ній лише частково реалізовано API для експорту результатів обліку – програма здатна завантажувати результати в файли, проте інтеграції з хмарними сервісами немає.

2.1.2. Checkcfg

Checkcfg [4] — це програмний коплекс (для коректної роботи потрібно встановити пакет програм, що складається із checkcfg, Doberman, Sklad та

Sklad_w) розроблена ентузіастами, тому розповсюджується безплатно, проте з закритим кодом. Вона працює на та здатна працювати з операційними системами Windows розпочинаючи з 95 і закінчуючи 8. Сервіс активно розроблявся в 2000-них, і повністю перестав розвиватись в 2012, тому нині є застарілим. Відсутня можливість хоч якось зберегти результати роботи у вигляді звітів.

2.1.3. Network Inventory Advisor

Network Inventory Advisor [5] — це програма розроблена компанією ClearApps LLC. Вона є платним пропрієтарним рішенням та не є кросплатформенною, працює лише на Windows. Здатна працювати з іншими операційними системами. По свої суті програма є ідентичною Total Network Inventory з пункту 1.3.1.

2.1.4. Spiceworks

Spiceworks [6] — це програма розроблена компанією ziff davis b2b focus, inc. поширюється безплатно, проте в ній є можливість докупити платні функції. Тим не менш код програми на момент написання є закритим. Працює виключно на Windows, але сканує і інші операційні системи. Часткова реалізація експорту звітів — тільки у файли.

2.1.5 OCS Inventory

OCS Inventory [7] — це безкоштовне програмне забезпечення, що розповсюджується за моделлю open source. Працює на Linux та інвентаризує: Microsoft Windows, Linux, BSD, Sun Solaris, IBM AIX, HP-UX, MacOS X, Android. Працює через встановлення на об’єкти обліку агентів, хоча можлива і робота через мережу. Нереалізований експорт в інші програми.

2.1.6 Існуючі реалізації: Висновок

З проведеного дослідження очевидно, що уже існує досить велика кількість реалізацій поставленої задачі. Тому було прийнято рішення зупинитись на вище наведених п’яти і сформувати таблицю з переліком найбільш популярних програм. Варто зазначити, що стовпець «API експорту результатів» матиме значення «+» тільки якщо буде реалізовано експорт до Google sheets і «+/–» у випадку коли результати можна скачати. Колонка «Зручний інтерфейс» суб’єктивна — основним показником був час потрібний на те щоб розібратись в інтерфейсі і додати 1 комп’ютер для обліку та провести облік.

Таблиця 2.1.6.1. Порівняння існуючих реалізацій автоматичного обліку

Назва програм	Парадигма поширення	API експорту результатів	Зручний інтерфейс	Кросплатформеність суб’єкта	Кросплатформеність об’єкта	Агенти обліку?
Total Network Inventory	Proprietary	+/–	+	–	+	-
Checkcfg	Freeware	+/–	–	–	–	+
Network Inventory Advisor	Proprietary	+/–	+	–	+	–
Spiceworks	Shareware	+/–	+	–	+	–
OCS Inventory	Open source	+/–	+	+–	+	+
Lansweeper [8]	Shareware	+/–	–	–	+	+/–
Open–AudIT [9]	Open source	+/–	+	+	+	–

Назва програм	Парадигма по-ширення	API експорту результатів	Зручний інтер-фейс	Кросплатфор-меність суб'єкта	Кросплатфор-меність об'єкта	Агенти обліку?
EMCO Network Inventory [10]	Shareware	+/–	+	–	+	–
10-Strike Network Inventory Explorer [11]	Shareware	+/–	–	–	–	–
Asset Tracker for Networks [12]	Shareware	+/–	–	–	+	–
Network Inventory PRO [13]	Shareware	+/–	–	–	+	–
Network Inventory Expert [14]	Shareware	–	+	–	+	–
Alloy Network Inventory [15]	Freeware	–	–	–	+	–
RedBaron Network Inventory System [16]	Freeware	+/–	–	+	+	–

Продовження таблиці 2.1.6.1 — Порівняння існуючих реалізацій автоматичного обліку

2.2. Розгляд робити одного з описаних рішень

Серед описаних раніше рішень OCS Inventory є одним із найоптимальніших, тому було вирішено провести більш детальний розгляд його роботи. Далі наводиться покрокове розгортання та перевірка роботи цього рішення автоматичного аудиту.

2.2.1 Встановлення та налаштування OCS Inventory

Встановлення сервісу відносно простою задачею, яка тим не менш, може забрати досить багато часу. Для встановлення використовувався інструкції з офіційного сайту [19]. Після встановлення варто впевнитись у правильності конфігурації. Для цього варто зайти на сайт http://127.0.0.1/ /ocsreports/. Якщо завантажується сторінка, тоді все в порядку. На сайті потрібно ввести наступну інформацію:

- Root
- «пароль від рута»
- Ocsweb
- Localhost

Після цього буде створено базу даних, з якою буде працювати сервер.

Після встановлення заходимо на сервер використовуючи логін «admin»

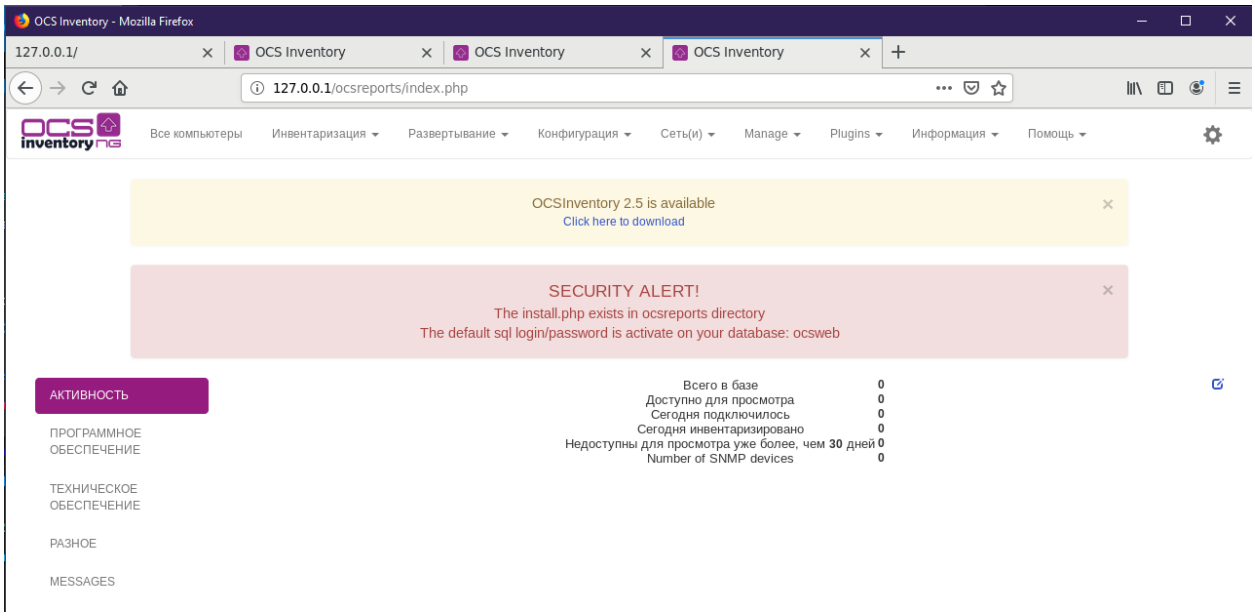

Рисунок 2.2.1.1 Головне меню сервера OCS Inventory

і пароль «admin»

2.2.2. Встановлення агента на Windows

Під час установки потрібно вибрати мережевий тип обліку та ввести дані для підключення до серверу [20][21].

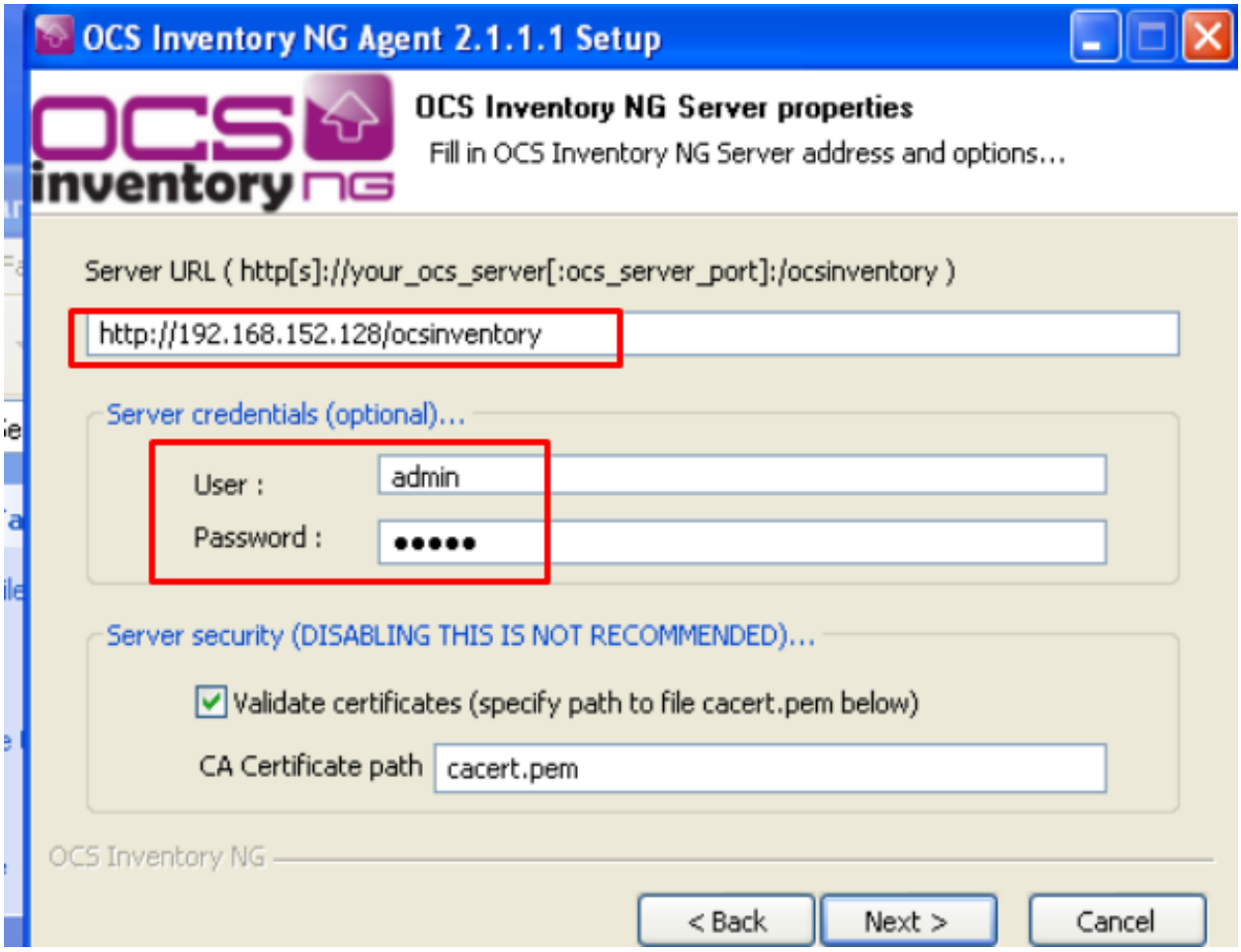

Рисунок 2.2.2.1 Установка клієнта на Windows. Введення password “admin”

Після цього потрібно оновити сторінку на сервері і, якщо все було зроблено правильно, комп’ютер буде додано до списку.

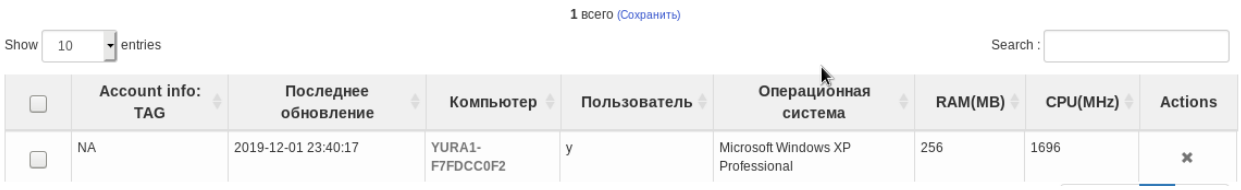

Рисунок 2.2.2.2 Доданий до списку комп’ютер (Windows)

Тепер ми з нашого сервера маємо доступ до всієї інформації про цей комп’ютер. Наприклад можна спробувати вивести інформацію про процесор.

Рисунок 2.2.2.3 Інформація про процесор (Windows)

2.2.3. Встановлення агента на Linux

Команди необхідні для встановлення OCS inventory agent на Linux систему можна знайти на офіційному сайті [22]. Після встановлення налаштування відбувається по більшій частині зі значеннями за замовчуванням окрім наступних:

[1] Where do you want to write the configuration file?

2

[2] What is the address of your ocs server?

http://192.168.152.128/ocsinventory

[3] Do you want to activate debug configuration option ?

n

[4] Specify log file path you want to use?>

/var/log/ocs_agent.log

[5] Specify CA certificate chain file path?>

/etc/ocsinventory-agent/cacert.pem

Після завершення налаштувань спробуємо передати інформацію на сервер за допомогою команди «/usr/sbin/ocsinventory-agent». Якщо все нала-

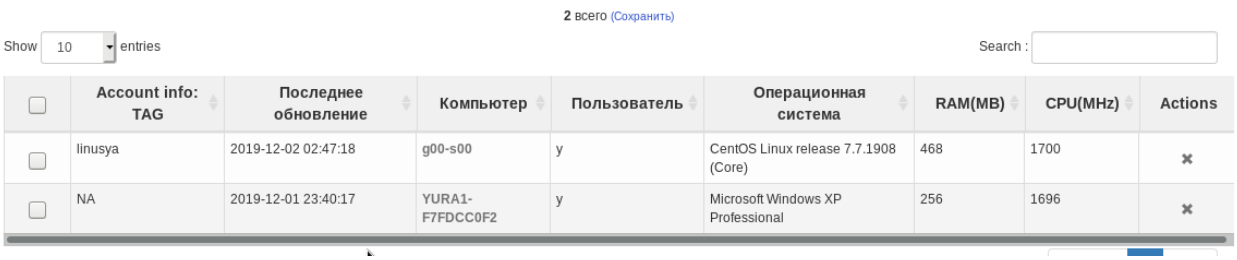

Рисунок 2.2.3.1 Доданий до списку комп’ютер (Linux)

штовано правильно буде додано персональний комп’ютер до списку на сервері.

2.2.4. Розгортання сервісу: Висновок

Програма OCS Inventory справляється з поставленою задачею — проведення обліку. Проте виявлені недоліки описані у 2.1.5 — використання агентів, суб’єкт на Linux. Програма збирає інформацію в БД і виводить її на веб сторінці, проте не було знайдено певної документації або API, що б надавали можливість прямого експорту інформації в хмарне середовище.

РОЗДІЛ 3. РОЗРОБКА ТА ПЕРЕВІРКА РОБОТИ РІШЕННЯ АВТОМАТИЧНОГО АУДИТУ

Перед початком розробки рішення було сформульовано загальний принцип роботи майбутнього продукту. Після встановлення всіх необхідних для роботи програм, на базі яких працює програмний продукт, скачування скриптів, та підготовки до роботи з Google Api все, що має бути потрібно від користувача — це записати в файл (нехай він буде мати назву «hosts») IP-адреса, включити комп'ютери, що мають піддатися обліку та запустити програму (нехай вона називається «program»). Надалі при кожному запуску програми повинен бути створений новий файл обліку та проведений аудит.

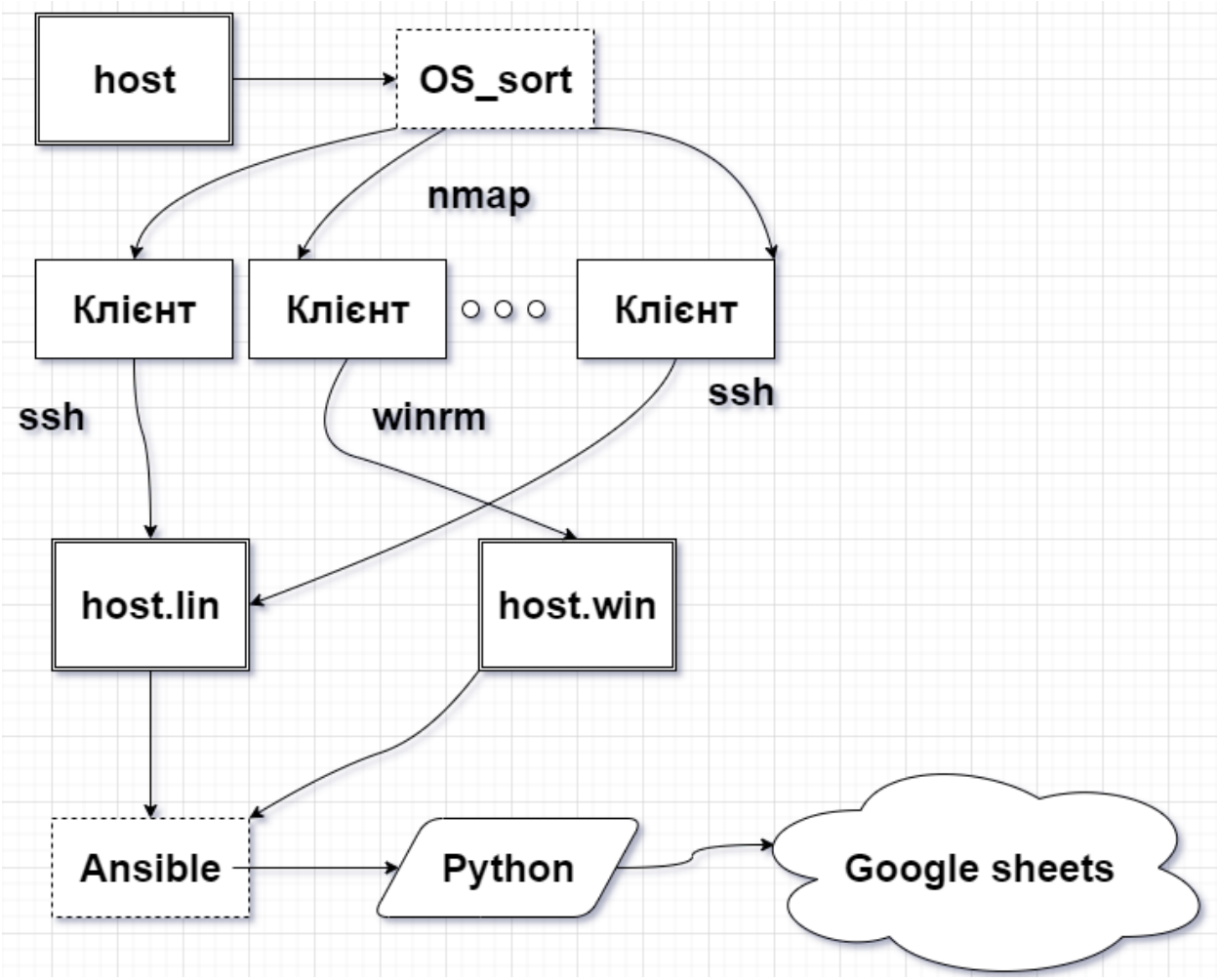

Рисунок 3.1. Схема взаємодії складових програмного комплексу

3.1. Вибір програми, що забезпечить взаємодію компонентів програмного продукту

Серед наведених в пункті 1.6 було вирішено зупинитись на системах оркестрування тому, що або відсутнє покриття різних OS, або надається надлишковий функціонал, що не буде використовуватись. Окрім того в процесі роботи було вирішено зупинити вибір саме на системах оркестрування.

3.1.1. Вибір системи оркестрування

Так як систем оркестрування є декілька, було вирішено провести їх порівняння для подальшого вибору системи, що буде використано для розробки рішення автоматичного аудиту.

Таблиця 3.1.1.1 Порівняння систем оркестрування

Назва програми	Простота	Без клієнтів	Поширення	Push vs Pull	Мова
Ansible	+	+	GPLv3+	1&2	Python
CFEngine	–	–	GPLv3	2	C
Puppet	+	–	Apache 2.7	2	C++
Chef	–	–	Apache 2.0	2	Ruby
SaltStack	+/–	+/–	Apache 2.0	1&2	Python

Було вирішено зупинити свій вибір на Ansible, так як він найкраще підходить за принципом роботи та задовольняє поставлені.

3.2. Вибір технологій для збору та обробки інформації, а також середовища в якому працюватиме програмний продукт

Ansible можна встановити на Windows10, проте для цього буде потрібно зробити додаткові кроки, що в цій роботі вважається недоцільним, тому як сервер було вирішено вибрати OS Linux Ubuntu 18.04.4 LTS. Відповідно для обробки результатів планується використовувати мову bash.

Ansible використовує файли інвентаризації, разом зі параметрами до них у форматі YAML. Для оркестрування Unix-подібних систем використовується SSH, для Windows WinRM (можливі варіанти будуть розглянуті пізніше). Через, що використання одного файлу інвентаризації не підходить. Для того щоб все ж досягти поставлених вимог в пункті 1.1 було вирішено використовувати Nmap для сортування OS і автоматичного створення файлів інвентаризації. Окрім того таке рішення захищає від проблеми коли один із об’єктів обліку виключений, хоча все ж знаходиться в списку на облік. Тим не менш така поведінка програми не передбачається, тому вивід програми може бути некоректним.

Для вивантаження результатів у Google Sheets буде використано мову програмування python та API, що надається компанією Google під назвою «Google Sheets API v4».

3.3 Розгортання створеного програмного продукту та перевірка роботи

3.3.1. Встановлення та налаштування Ansible

Після встановлення потрібно згенерувати пару SSH ключів та передати ключ адміністратору (root) об’єктів обліку [23]. Також можна використовувати і звичайного користувача, проте тоді доведеться дещо змінювати налаштування у файлі «OS_sort» до того ж не всі команди, що потрібно вийде

запустити і, відповідно, тоді файл обліку буде неповний. Так є команда «lshw», яка виводить потрібні дані тільки від імені адміністратора.

Для перевірки правильності налаштувань можна виконати команду, що зображена на Рисунок 3.5.1. Файл hosts.lin повинен містити ip адреса об’єктів обліку, що були налаштовані. Через аргумент «-i hosts.lin» Ansible розуміє який файл містить інформацію інвентаризації. «-m ping» вказує який модуль ми хочемо використати. В кінці потрібно вказати ip адрес який ми хочемо

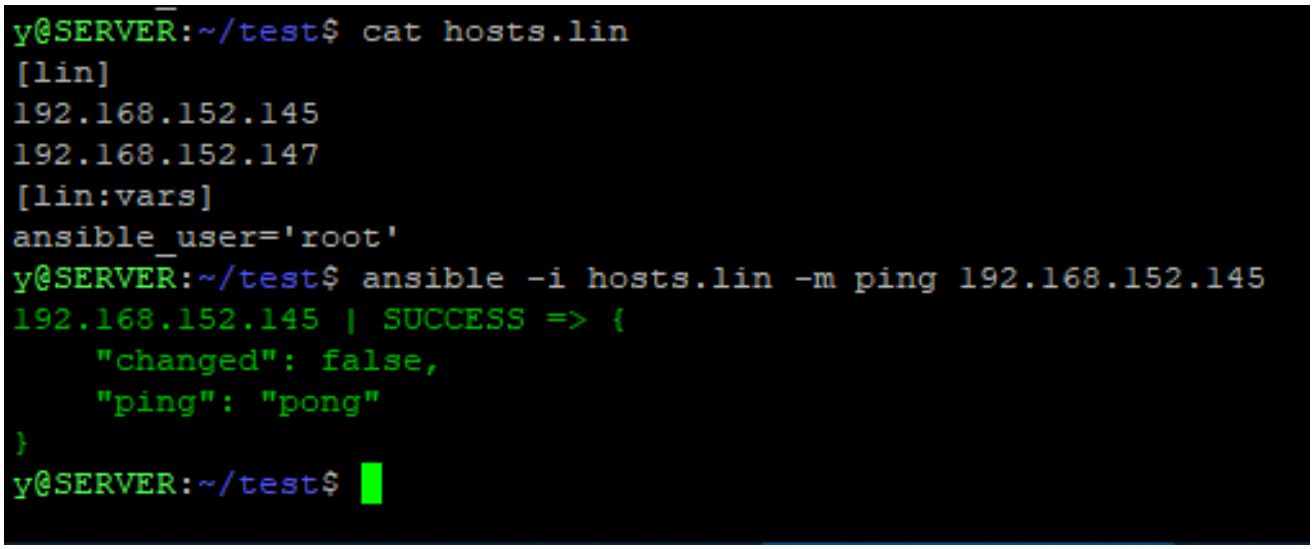

Рисунок 3.3.1.1 Перевірка правильності налаштувань Ansible + Linux

перевірити.

З Windows буде трохи складніше. Є 2 можливих варіанта як можна налаштувати Windows [24]:

- WinRM;
- SSH, через встановлення Linux підсистеми та Windows OpenSSH.

Другий варіант більш цікавий — так як це дещо складніше у налаштуванні, проте значно простіше у подальшому написанні програми та використанні — це варіант дозволив би використовувати лише 1 файл (відповідно відмовитись від nmap) інвентаризації в Ansible, та це б в результаті призвело до консолідованого виводу в результуючому файлі. Крім того це збільшило б безпечність програми, так як в першому варіанті доводиться передавати па-

ролі при під’єднанні. Проте через емерджентний феномен взаємодії середовища тестування та програмного забезпечення даний варіант так і не запрацював, тому було вирішено зупинитись на першому.

Перевірка налаштувань аналогічна попередній. На відміну від Linux де використовується SSH без пароля завдяки доданому на клієнт ключу для Windows для ansible_user та ansible_password потрібно ввести дані про користувача Windows.

```
y@SERVER:~/test$ cat hosts.win
[win]
192.168.152.148
[win:vars]
ansible_user='y'
ansible_password=y
ansible_connection=winrm
ansible_port=5986
ansible_port=5986
ansible_winrm_server_cert_validation=ignore
y@SERVER:~/test$ ansible -i hosts.win -m win_ping 192.168.152.148
192.168.152.148 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
y@SERVER:~/test$
```

Рисунок 3.3.1.2 Перевірка правильності налаштувань Ansible + Windows

3.3.2. Підключення Google Sheets API v4

Для того щоб завантажувати дані в Google Sheets потрібно мати акаунт Google та створити проект, за рахунок якого отримати доступ до сервісів Google [25][26]. Це тривіальне завдання для якого досить документації, тому в цій роботі його було вирішено опустити.

3.3.3. Огляд тестового середовища використаного для перевірки функціоналу розробленого програмного продукту

Для тестування було використано 4 віртуальні машини. Три з встановленою Ubuntu 18.04.4 LTS і четверта з Windows 10.

Важливо звернути увагу на характеристики першого клієнта, а саме кі-

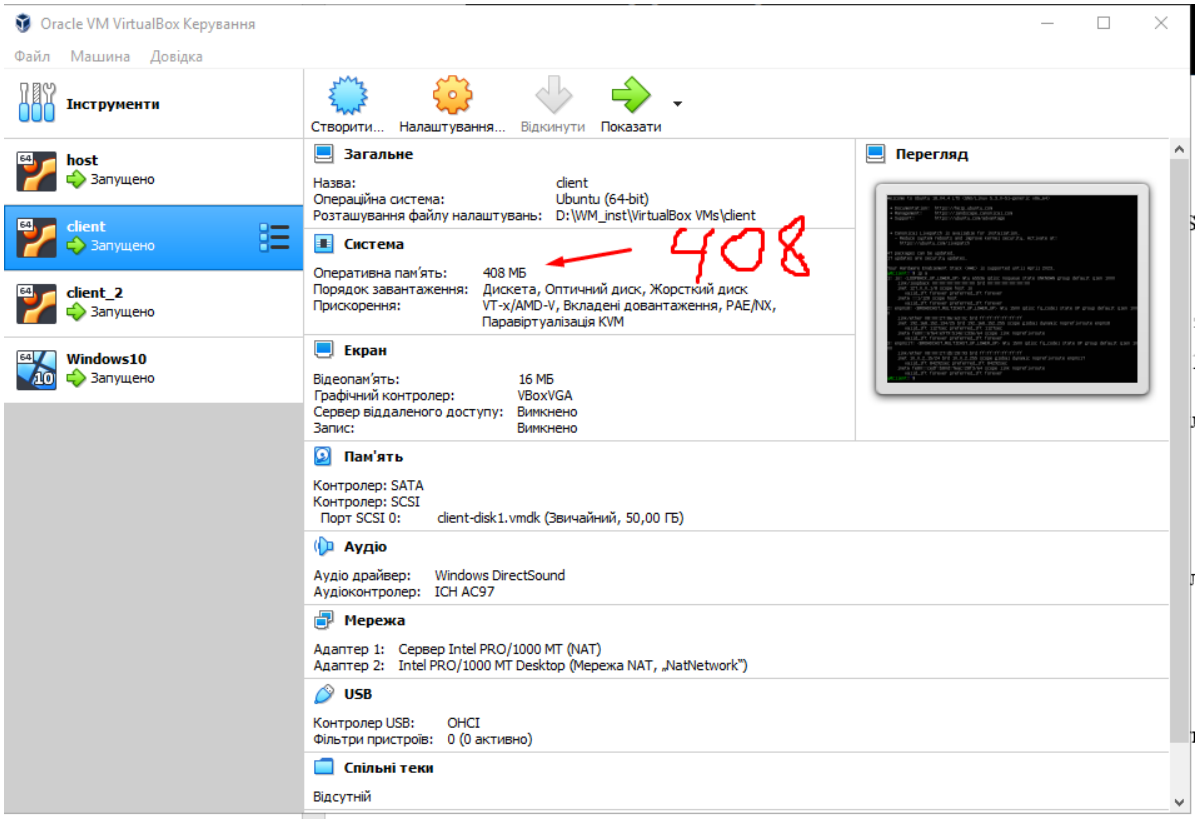

Рисунок 3.3.3.1 Характеристики першого клієнта. Акцент на кількості RAM – це буде виористано далі у роботі.

лькість оперативної пам'яті – ця інформація буде використана пізніше.

3.3.4. Огляд програмного продукту перед запуском

Всі файли програми повинні міститись в 1-ній директорі.

```
y@SERVER:~/test$ ll
total 48
drwxrwxr-x  2 y y 4096 Jun  4 10:42 ./
drwxr-xr-x 21 y y 4096 Jun  4 10:40 ../
-rwxr-xr-x  1 y y 4410 May 27 06:25 ansible_work.sh*
-rw-rw-r--  1 y y 1534 Jun  2 08:38 basic_check.py
-rw-rw-r--  1 y y 2302 May 26 05:14 diplosheets-4490f33dbc26.json
-rw-rw-r--  1 y y 2302 May 26 05:14 diplosheets-91642d60da4d.json
-rw-rw-r--  1 y y 3820 May 26 05:14 fill_invent_sheet.py
-rw-r--r--  1 y y 2724 May 26 05:14 fill_main.py
-rw-rw-r--  1 y y 2830 May 26 08:22 new_spreadsheet.py
-rwxr-xr-x  1 y y  442 Jun  4 10:37 OS_sort*
-rwxrwxrwx  1 y y  518 May 27 06:29 program.sh*
```

Рисунок 3.3.4.1 Директорія з файлами програмного продукту

Короткий опис призначення даних файлів (вміст цих файлів можна знайти в додатках):

- ansible_work.sh — це основа програмного комплексу — збирає дані з Комп’ютери , що додано до файлів інвентаризації;
- basic_check.py — перевірка збіжності результатів поточного обліку з певним еталоном (який буде створено після першого обліку);
- diplosheets-4490f33dbc26.json та diplosheets-91642d60da4d.json файли які отримуються в пункті 3.6. Достатньо одного, 2-й про запас.
- fill_invent_sheet.py — заповнення файлів обліку в Google Sheets
- fill_main.py — заповнення файлу файлів (містить посилання на файли в яких зберігаються результати попереднього обліку)
- new_spreadsheet.py — файл, що створює нові файли у Google Sheets
- OS_sort — створення файлів інвентаризації з якими може працювати Ansible з файлу користувача «hosts»
- program.sh — запуск програмного комплексу. Тільки з цією частиною програми взаємодіє користувач.

3.3.5 Запуск рішення для проведення обліку

```
y@SERVER:~/test$ echo "192.168.152.134" >hosts
y@SERVER:~/test$ echo "192.168.152.135" >>hosts
y@SERVER:~/test$ echo "192.168.152.132" >>hosts
y@SERVER:~/test$ ll
total 52
drwxrwxr-x  2 y y 4096 Jun  4 10:42 ./
drwxr-xr-x 21 y y 4096 Jun  4 10:40 ../
-rwxr-xr-x  1 y y 4410 May 27 06:25 ansible_work.sh*
-rw-rw-r--  1 y y 1534 Jun  2 08:38 basic_check.py
-rw-rw-r--  1 y y 2302 May 26 05:14 diplosheets-4490f33dbc26.json
-rw-rw-r--  1 y y 2302 May 26 05:14 diplosheets-91642d60da4d.json
-rw-rw-r--  1 y y 3820 May 26 05:14 fill_invent_sheet.py
-rw-r--r--  1 y y 2724 May 26 05:14 fill_main.py
-rw-rw-r--  1 y y   48 Jun  4 10:43 hosts
-rw-rw-r--  1 y y 2830 May 26 08:22 new_spreadsheet.py
-rwxr-xr-x  1 y y  442 Jun  4 10:37 OS_sort*
-rwxrwxrwx  1 y y  518 May 27 06:29 program.sh*
```

Рисунок 3.3.5.1 Створення файлу hosts

Перед запуском потрібно додати ip адреса в файл «hosts».

Після цього можна проводити перший запуск програми.

Програма видає 2 посилання:

- https://docs.google.com/spreadsheets/d/1mfKh_hdWr9z1xmBuvju1mnmT34MY3YijV_mXCY9nLUU
- https://docs.google.com/spreadsheets/d/1-gssNYtQA2tnaAQSFjBUatnBS6Y17cC8vJgg4OU1CH8

Перше містить файл файлів обліку. Кожного разу коли створюється новий файл обліку, посилання на нього записується до цього файлу. Цей файл ство-

```
y@SERVER:~/test$ sudo ./program.sh
start of ./program.sh
Thu Jun  4 10:53:52 PDT 2020
new progect
https://docs.google.com/spreadsheets/d/1mfKh_hdWr9z1xmBuvju1mnmT34MY3YijV_mXCY9nLUU

https://docs.google.com/spreadsheets/d/1-gssNYtQA2tnaAQSFjBUatnBS6Y17cC8vJgg4OU1CH8

Running (JUST GUESSING): Microsoft Windows XP|7|2008 (87%)
Aggressive OS guesses: Microsoft Windows XP SP2 (87%), Microsoft Windows 7 (85%), Mi
r Windows Server 2008 R2 (85%)
192.168.152.134
192.168.152.135
192.168.152.132
Thu Jun  4 10:58:42 PDT 2020
end of ./program.sh
```

Рисунок 3.3.5.2 Перший запуск та проведення обліку

рюється лише один раз, надалі програма буде повертати лише одне посилання — на новий файл інвентаризації. Цей файл є відображенням локального файлу proect_all_spreadsheet — якщо його видалити то програма вважатиме, що вона запускається вперше.

Друге, власне, і є файл обліку. По завершенню виконання програми в ньо-

```
y@SERVER:~/test$ ll
total 116
drwxrwxr-x  8 y    y    4096 Jun  4 11:00 ./
drwxr-xr-x 21 y    y    4096 Jun  4 11:00 ../
-rw-r--r--  1 root root 8797 Jun  4 10:55 all1.txt
-rw-r--r--  1 root root 8812 Jun  4 10:56 all2.txt
-rw-r--r--  1 root root 2788 Jun  4 10:58 all3.txt
-rwxr-xr-x  1 y    y    4410 May 27 06:25 ansible_work.sh*
-rw-rw-r--  1 y    y    1534 Jun  2 08:38 basic_check.py
-rw-rw-r--  1 y    y    2302 May 26 05:14 diplosheets-4490f33dbc26.json
-rw-rw-r--  1 y    y    2302 May 26 05:14 diplosheets-91642d60da4d.json
-rw-rw-r--  1 y    y    3820 May 26 05:14 fill_invent_sheet.py
-rw-r--r--  1 y    y    2724 May 26 05:14 fill_main.py
-rw-rw-r--  1 y    y      48 Jun  4 10:43 hosts
-rw-r--r--  1 root root   69 Jun  4 10:55 hosts.lin
-rw-r--r--  1 root root  174 Jun  4 10:55 hosts.win
-rw-rw-r--  1 y    y    2830 May 26 08:22 new_spreadsheet.py
-rwxr-xr-x  1 y    y     445 Jun  4 10:52 OS_sort*
-rw-r--r--  1 root root  144 Jun  4 10:54 .proect_all_spreadsheet
-rwxrwxrwx  1 y    y     518 May 27 06:29 program.sh*
drwxr-xr-x  2 root root 4096 Jun  4 10:55 temp1/
drwxr-xr-x  2 root root 4096 Jun  4 10:55 temp2/
drwxr-xr-x  2 root root 4096 Jun  4 10:57 temp3/
drwxr-xr-x  2 root root 4096 Jun  4 10:58 temp.check1/
drwxr-xr-x  2 root root 4096 Jun  4 10:58 temp.check2/
drwxr-xr-x  2 root root 4096 Jun  4 10:58 temp.check3/
```

Рисунок 3.3.5.3 Стан директорії після першого запуску

го запишеться інформація про останній облік.

Після першого запуску в директорії з’явилось декілька нових файлів. Вони є службовими, тим не менш ось їх опис:

- all*.txt — файли з детальною інформацією про PC;
- hosts.win & hosts.lin — файли створені на основі hosts таким чином щоб Ansible міг їх прочитати (за умови якщо Комп’ютери налаштовані правильно);
- .proect_all_spreadsheet — локальна версія файла файлів;
- temp* — директорії з інформацією про останній аудит;

- temp.check* — директорії з еталонною інформацією.

Результати останнього обліку можливо переглянути і локально — на

```
y@SERVER:~/test$ cat temp2/*
08:00:27:63:48:bf
08:00:27:1a:58:7c
        Manufacturer: Oracle Corporation        Product Name: VirtualBox
             product: 82540EM Gigabit Ethernet Controller
             product: 82545EM Gigabit Ethernet Controller (Copper)
192.168.152.135
Linux second_client x86_64 GNU/Linux
Model name:          Intel(R) Core(TM) i3-4005U CPU @ 1.70GHz
00:02.0 VGA compatible controller: InnoTek Systemberatung GmbH VirtualBox Graphics Adapter
MemTotal:         383000 kB
192.168.152.135 | FAILED | rc=1 >>
non-zero return code
       product: HARDDISK
        Vendor: innotek GmbH    Version: VirtualBox
y@SERVER:~/test$
```

Рисунок 3.3.5.4 Перегляд результатів останнього обліку локально

комп’ютері, що проводить аудит.

Переглянемо перше посилання. Там міститься Google таблиця з посиланнями на всі обліки.

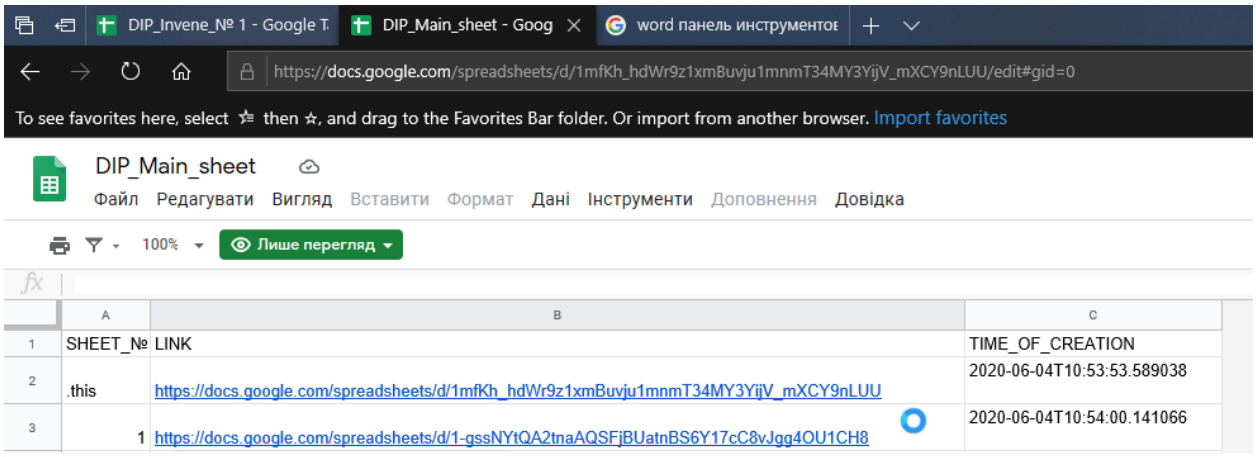

Рисунок 3.3.5.5 Google таблиця за першим посилання, що містить посилання на всі проведені обліки.

Переглянемо друге посилання:

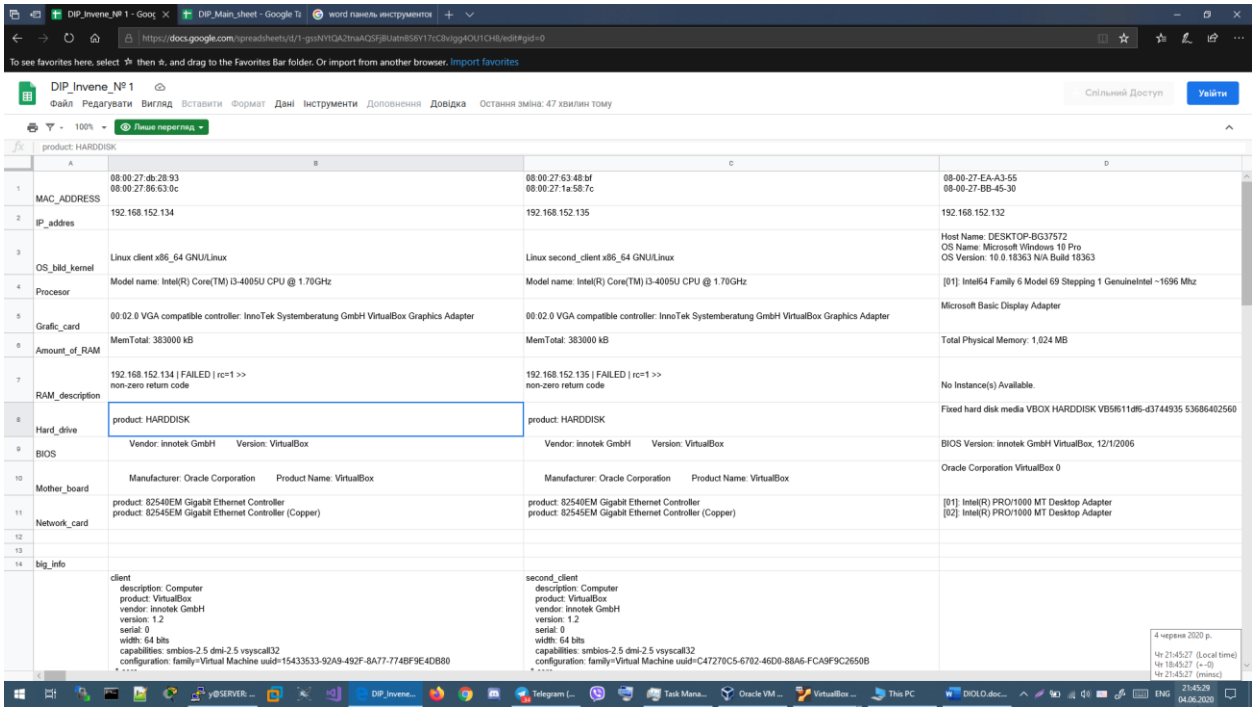

Рисунок 3.3.5.6 Друге посилання з Google таблицею що містить результати першого обліку

Рисунок 3.3.5.7 Друге посилання. Декілька клітинок при збільшенні

Якщо звернути увагу на деякі клітинки (Рисунок 3.3.5.7 або Рисунок 3.3.5.4) можна подумати, що програма при виводі деяких специфічних значень (в даному випадку модель RAM та Hard drive) працює не правильно. Справа в тому, що програма розроблялась для реального обладнання. Те ж, що надає VirtualBox всього на всього емуляція. Тільки імітація апаратного забезпечення. VirtualBox не напише назви RAM планок. VirtualBox не перетворить менеджмент жорстких дисків в мистецтво.

Потрібно перевірити чи немає хибно-позитивного спарцювання під час проведення другого обліку.

Для цього необхідно запустити облік ще раз не змінюючи нічого у об’єктах обліку.

```
y@SERVER:~/test$ sudo ./program.sh
[sudo] password for y:
start of ./program.sh
Thu Jun  4 11:58:56 PDT 2020
new inwentarization
https://docs.google.com/spreadsheets/d/1ijJVTtu2ZzWvlX8N_0N8SJRge74NYCoUMfVOXoBLFf8

Running (JUST GUESSING): Microsoft Windows 2008 (85%)
Aggressive OS guesses: Microsoft Windows Server 2008 SP1 or Windows Server 2008 R2 (85%)
192.168.152.134
192.168.152.135
192.168.152.132
Thu Jun  4 12:03:43 PDT 2020
end of ./program.sh
y@SERVER:~/test$
```

Рисунок 3.3.5.8 Запуск програмного комплексу без змін для виявлення хибно-позитивних спрацювань

Вивід консолі не змінився, у випадку якщо програмний комплекс зафіксував якісь зміни в апаратному забезпеченню на рисунку 3.3.5.8 повинно було б з’явитись повідомлення «варнінг!!!», що далі у роботі можна побачити на рисунку 3.3.6.2. В директорії без змін, тому відразу перейдемо до перегляду посилання на файл файлів — туди записалось нове посилання. Якраз те, що можна побачити на знімку екрана Рисунок 3.8.8. Самий файл за посиланням нічим не відрізняється від попереднього (тому знімок екрана не наводиться).

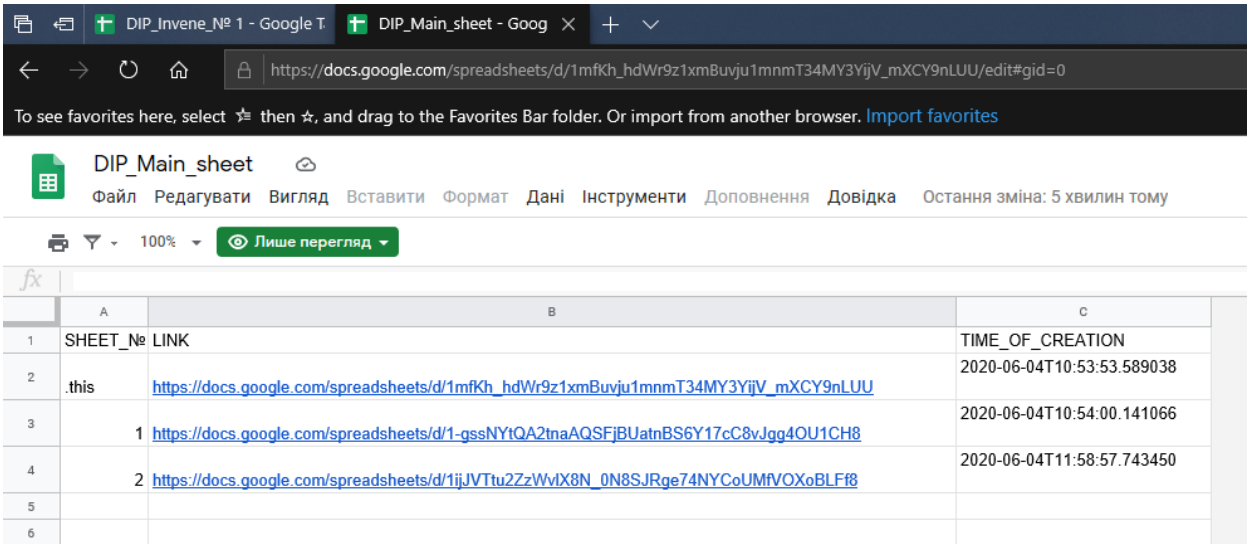

Рисунок 3.3.5.9 Стан Google таблиці що містить посилання на інші таблиці після повторного обліку

3.3.6 Перевірка правильності роботи аудиту

З попередніх кроків виходить, що облік працює. Залишається лише перевірити правильність роботи аудиту. Для цього спробуємо змінити кількість оперативної пам’яті на клієнті, що повинно призведе до зміни даних при наступному обліку.

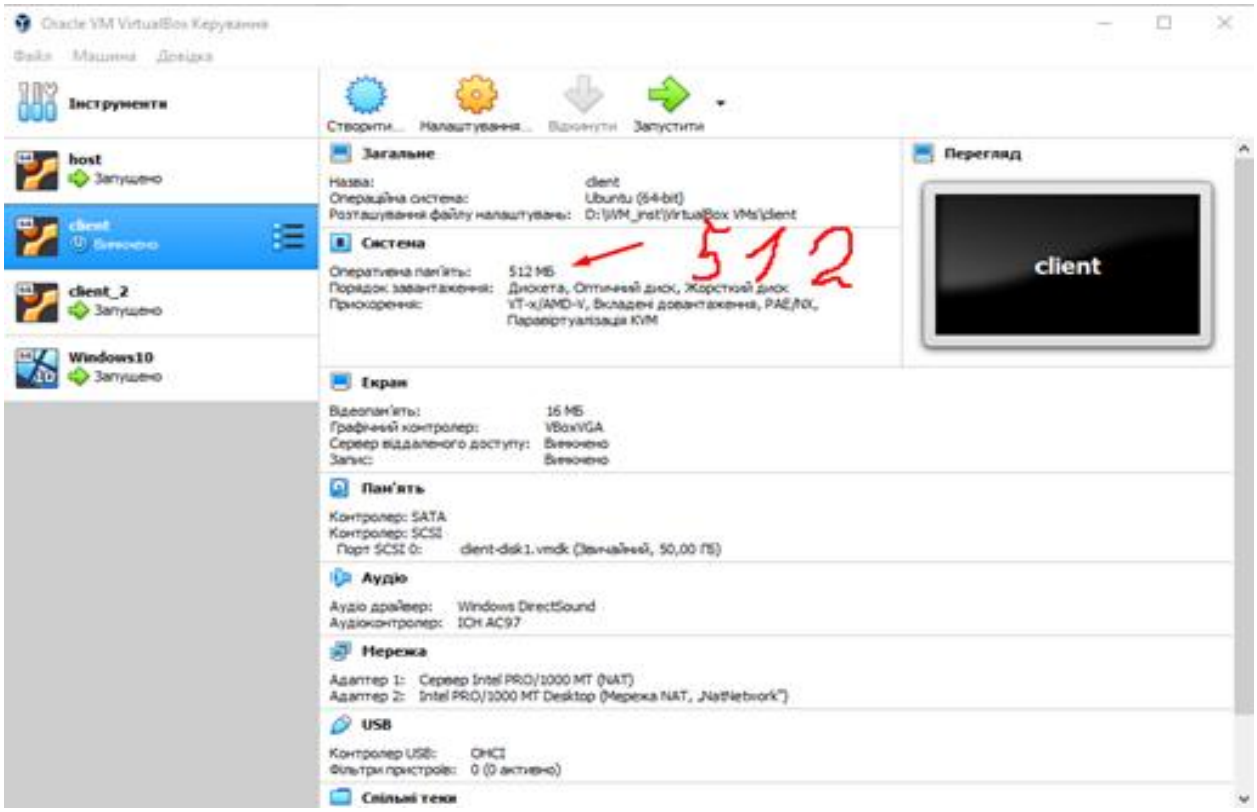

Рисунок 3.3.6.1 Зміна кількості RAM у об'єкта обліку з 408 до 512 MB

```
y@SERVER:~/test$ sudo ./program.sh
start of ./program.sh
Thu Jun  4 12:10:47 PDT 2020
new inwentarization
https://docs.google.com/spreadsheets/d/1Ey0AVETeRDC3NjgiqNhnG0Fz73mkzyQflaI9VOGHcvU

Running (JUST GUESSING): Microsoft Windows XP (85%)
Aggressive OS guesses: Microsoft Windows XP SP2 (85%)
192.168.152.134
192.168.152.135
192.168.152.132
Warning !!! change in last oblick in List_of_all_inventarizateon!B6:B6
Thu Jun  4 12:15:31 PDT 2020
end of ./program.sh
y@SERVER:~/test$
```

Рисунок 3.3.6.2 Запуск програми після змін в об'єктах обліку. Консоль виводить сповіщення про зміни в певній клітинці таблиці

Запустимо все ще раз, уже зі змінами у об'єкта обліку.

У виводі з’явилось попередження про те, що були якісь зміни. Як і раніше, для початку файл файлів.

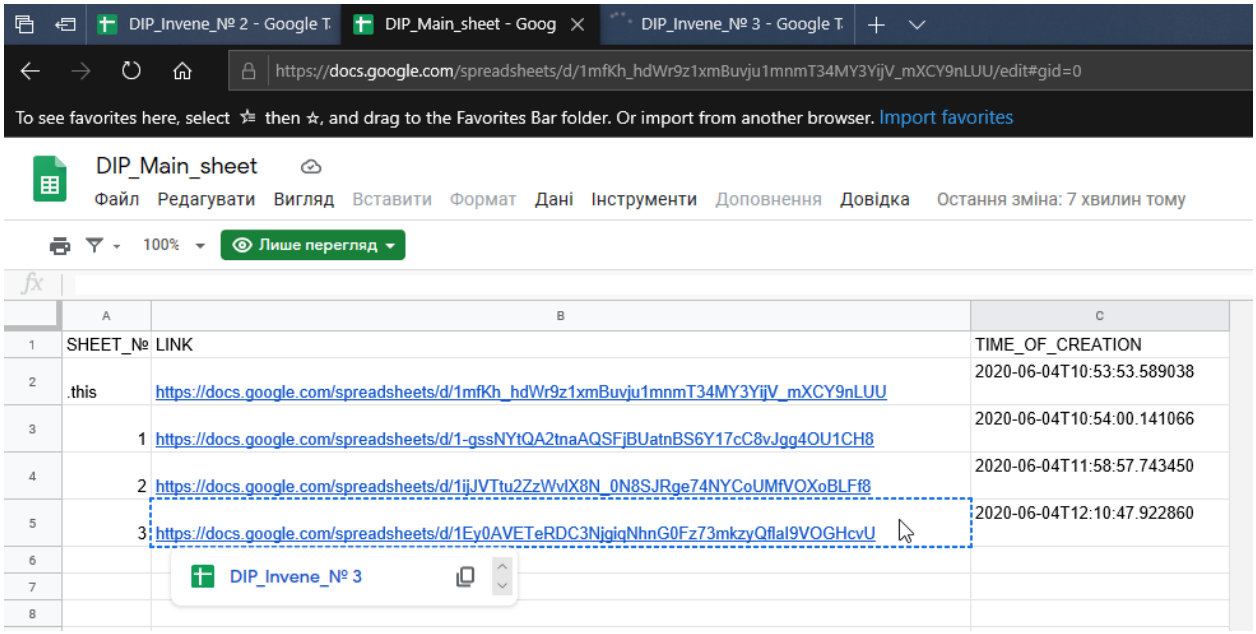

Рисунок 3.3.6.3 Стан Google таблиці що містить посилання на інші таблиці

Тепер відкриємо сам файл останнього обліку.

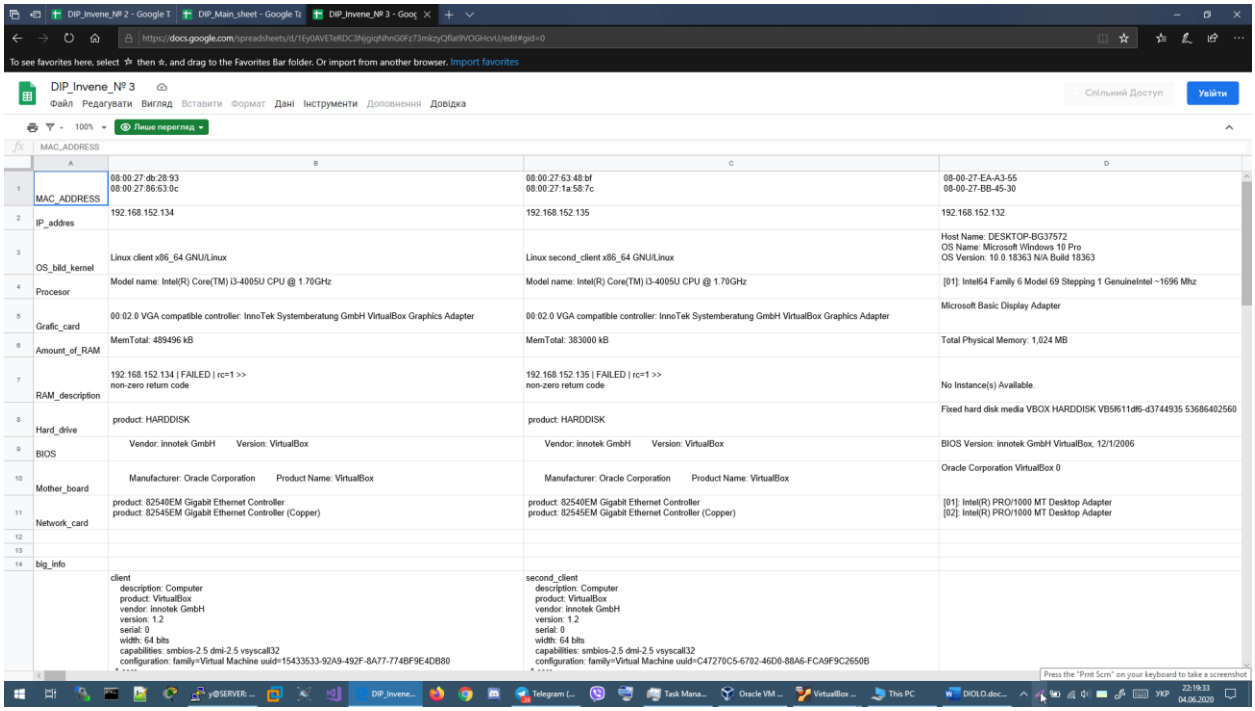

Рисунок 3.3.6.4 Результат третього обліку після змін кількості RAM

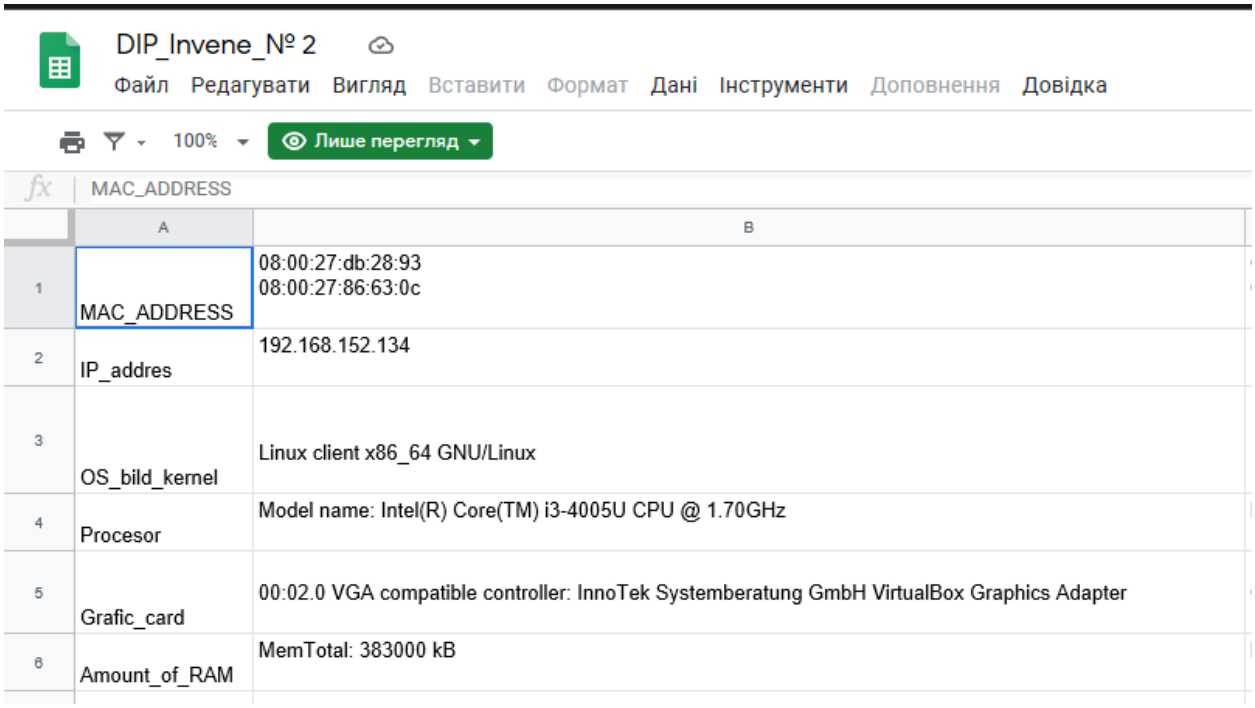

Рисунок 3.3.6.5 Результат другого обліку, де ще не було змін кількості RAM

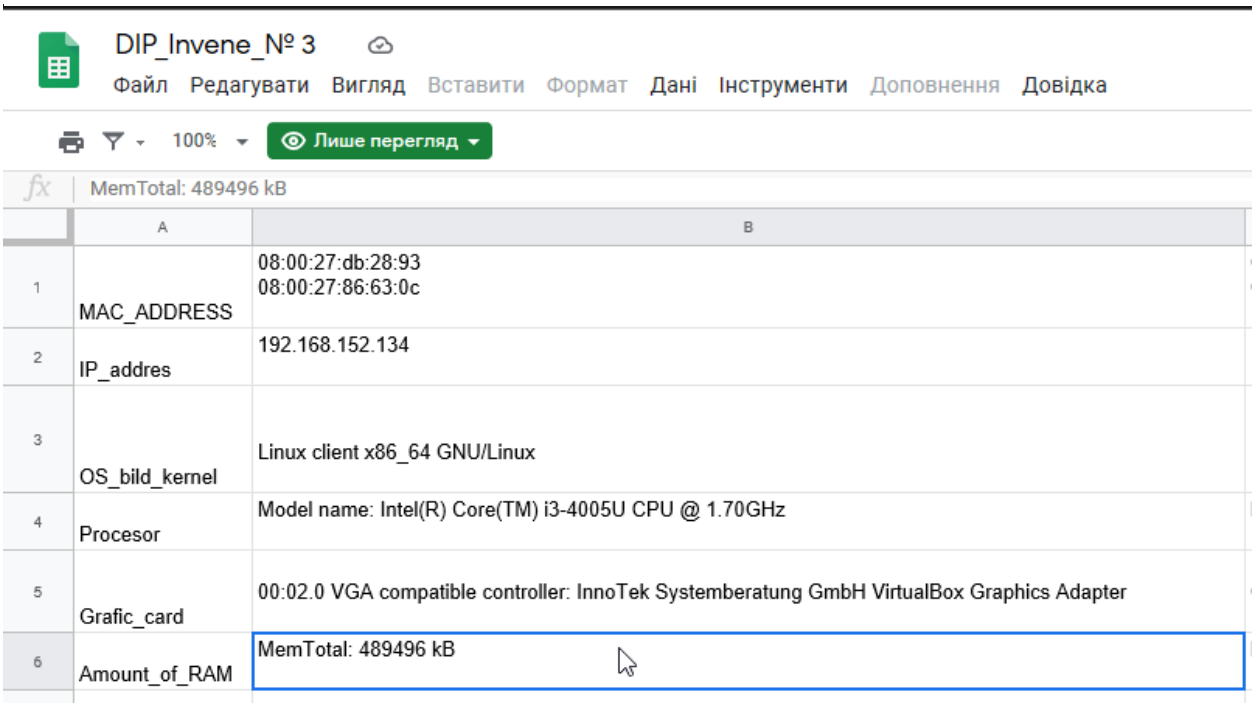

Рисунок 3.3.6.6 Результат третього обліку після змін кількості RAM

Для наглядності порівняємо результати другого та третього обліків.

Перевіримо пошту адміністратора, що вказувалась під час підключення Google Sheet API. Прийшов лист з координатами клітинки в якій змінились дані. Що свідчить про те що аудит у розробленому рішенні працює правиль-

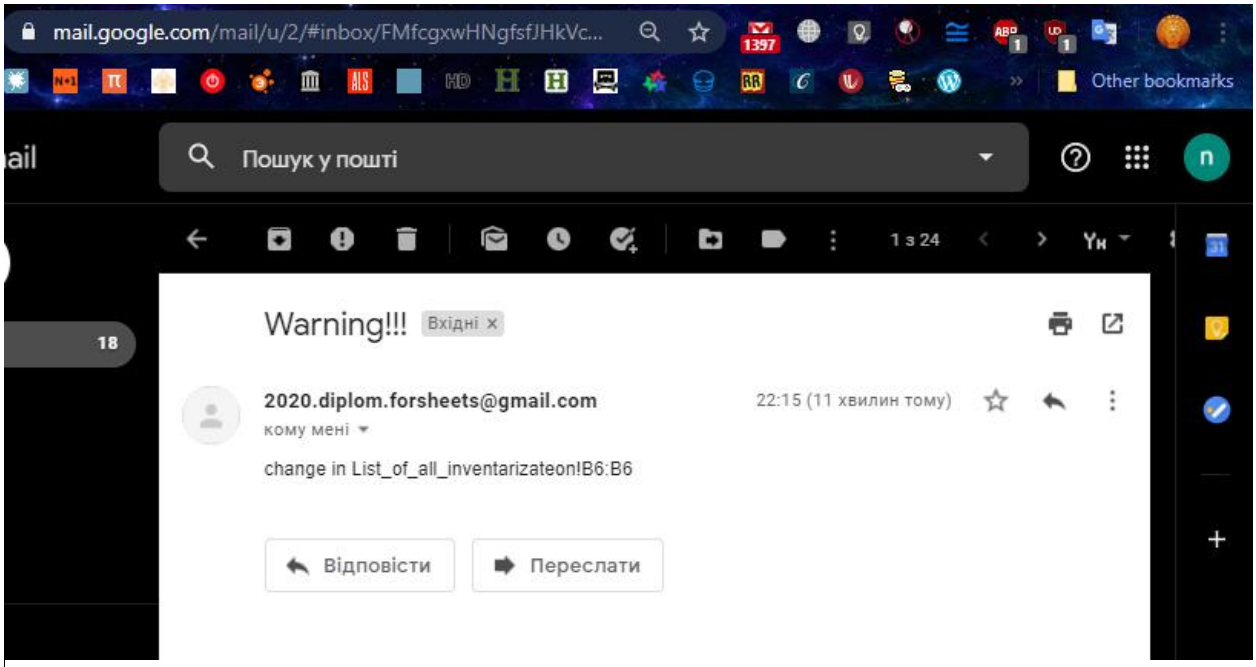

Рисунок 3.3.6.7 Повідомлення на пошту адміністратору про зміни в апаратному забезпеченні об’ктів обліку

но.

ВИСНОВКИ

З проведеного у роботі дослідження існуючих рішень автоматизованого аудиту апаратного забезпечення було виявлено, що вони мають наступні недоліки: використання клієнтів для збору інформації, програмне забезпечення не є вільним, відсутність інтеграції з хмарними сховищами, відсутня можливість доступу до обліків віддалено, відсутня кросплатформність об’єктів та суб’єктів обліку. Відсутність інтеграції з хмарними сховищами властива для всіх рішень.

Для створення нового рішення проаналізовано технології та вибрано наступні:

- для проведення неповного обліку було вибрано сканування TCP та UDP-портів використовуючи відкрите програмне забезпечення NMAP;
- для проведення обліку використано мови bash та python на яких було написано програмний комплекс, що використовує відкрите програмне забезпечення оркестрування Ansible. Ansible дозволяє з вертатись до об’єктів та залежно від їх операційної системи та використовуючи SSH та WinRM збирати інформацію про апаратне забезпечення. Окрім цього на відміну від інших систем оркестрування він не використовує клієнтів на об’єктах оркестрування.
- Для експорту результатів вибрано Google Документи. Використано мову python та Google Sheet API v4. Проведення аудиту відбувається перед експортом результатів, та у випадку змін адміністратор повідомляється листом на пошту.

Створене рішення не використовує клієнтів, об’єкти обліку є к мультиплатформними. Програмний комплекс здатний проводити облік та аудит основних компонентів апаратного забезпечення у межах локальної мережі з подальшим експортом у Google Документи.

ПЕРЕЛІК ПОСИЛАНЬ

[1] State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating [Електронний ресурс] – Режим доступу до ресурсу: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/ (дата звернення 08.05.2022)

[2] Обнаружение сетевых устройств [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/350394/ (дата звернення 08.05.2022)

[3] Total Network Inventory [Електронний ресурс] – Режим доступу до ресурсу: https://www.softinventive.com/products/total-network-inventory/ (дата звернення 08.05.2022)

[4] Учет и контроль компьютеров в сети (комплект программ) [Електронний ресурс] – Режим доступу до ресурсу: http://checkcfg.narod.ru/index.htm (дата звернення 08.05.2022)

[5] Professional Network Inventory Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.network-inventory-advisor.com/ (дата звернення 08.05.2022)

[6] Inventory your network devices. [Електронний ресурс] – Режим доступу до ресурсу: https://www.spiceworks.com/free-pc-network-inventory-software/ (дата звернення 08.05.2022)

[7] About OCS inventory [Електронний ресурс] – Режим доступу до ресурсу: https://ocsinventory-ng.org/?lang=en (дата звернення 08.05.2022)

[8] IT Asset Management Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.lansweeper.com/ (дата звернення 08.05.2022)

[9] Introducing Open-AudIT [Електронний ресурс] – Режим доступу до ресурсу: https://www.open-audit.org/ (дата звернення 08.05.2022)

[10] Network Inventory: Hardware and Software Inventory and Network Audit [Електронний ресурс] – Режим доступу до ресурсу: https://emcosoftware.com/network-inventory (дата звернення 08.05.2022)

[11] 10-Strike Network Inventory Explorer 4.0 [Електронний ресурс] – Режим доступу до ресурсу: http://www.all-nettools.com/network-inventory-3/10-strike-network-inventory-explorer-20172.htm (дата звернення 08.05.2022)

[12] Asset Tracker for Networks [Електронний ресурс] – Режим доступу до ресурсу: http://www.alchemy-lab.com/products/atn/index.html (дата звернення 08.09.2021)

[13] Network Inventory PRO [Електронний ресурс] – Режим доступу до ресурсу: http://www.alchemy-lab.com/products/berry/index.html (дата звернення 28.01.2020)

[14] Network Inventory Expert [Електронний ресурс] – Режим доступу до ресурсу: https://www.kviptech.com/network-inventory-expert/index.shtml (дата звернення 08.05.2022)

[15] Network Inventory Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.alloy-software.com/solutions/network-inventory/ (дата звернення 08.05.2022)

[16] RedBaron Network Inventory System [Електронний ресурс] – Режим доступу до ресурсу: https://sourceforge.net/projects/rnis/ (дата звернення 08.05.2022)

[17] Описание AIDA64 [Електронний ресурс] – Режим доступу до ресурсу: http://www.aida64.ru/desc (дата звернення 31.03.2022)

[18] Intel vPro [Електронний ресурс] – Режим доступу до ресурсу: https://www.intel.ru/content/www/ru/ru/architecture-and-technology/vpro/vpro-platform-general.html (дата звернення 8.11.2021)

[19] Setting up OCS Inventory Server [Електронний ресурс] – Режим доступу до ресурсу: https://wiki.ocsinventory-ng.org/03.Basic-

documentation/Setting-up-a-OCS-Inventory-Server/ (дата звернення 09.05.2022)

[20] OCS Inventory NG Agent 2.X on Windows Operating Systems [Електронний ресурс] – Режим доступу до ресурсу: https://wiki.ocsinventory-ng.org/03.Basic-documentation/Setting-up-the-Windows-Agent-2.x-on-client-computers/ (дата звернення 08.05.2022)

[21] Agent Windows 2.1.1.3 [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/OCSInventory-NG/WindowsAgent/releases/tag/2.1.1.3 (дата звернення 09.05.2022)

[22] OCS Inventory NG Agent 2.x on Unix Operating Systems [Електронний ресурс] – Режим доступу до ресурсу: https://wiki.ocsinventory-ng.org/03.Basic-documentation/Setting-up-the-UNIX-agent-on-client-computers/ (дата звернення 21.02.2020)

[23] Installing Ansible [Електронний ресурс] – Режим доступу до ресурсу: https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html (дата звернення 09.05.2022)

[24] Setting up a Windows Host [Електронний ресурс] – Режим доступу до ресурсу: https://docs.ansible.com/ansible/latest/user_guide/windows_setup.html (дата звернення 09.05.2022)

[25] Google Sheets API Overview [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/sheets/api/guides/concepts (дата звернення 09.05.2022)

[26] How-to guides [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/resource-manager/docs/how-to (дата звернення 09.05.2022)

ДОДАТКИ

Додаток А. Вміст файлу ansible_work.sh

```
#!/bin/bash
var=0
ls|egrep temp\[0-9]\+|xargs rm -rf
while read LINE
do
    if [[ $LINE =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
    let var=$var+1
    echo "$LINE"
    mkdir "temp${var}"
    ansible -i hosts.lin -m shell -a 'cat /sys/class/net/*/address' "$LINE" | sed '/SUCCESS/d' |grep -v 00:00:00:00:00:00 > "temp${var}/1" #MAC_Address
    echo "$LINE" > "temp${var}/2"   #IP_addres
    ansible -i hosts.lin -m shell -a 'uname -somn' "$LINE" | sed '/SUCCESS/d' > "temp${var}/3"      #OS_bild_kernel
    ansible -i hosts.lin -m shell -a 'lscpu | grep "Model name"'  "$LINE" | sed '/SUCCESS/d' > "temp${var}/4"       #PROCESOR
    ansible -i hosts.lin -m shell -a 'lspci | grep -E "VGA|3D"'  "$LINE" | sed '/SUCCESS/d' > "temp${var}/5"       #grafic_card
    ansible -i hosts.lin -m shell -a 'cat /proc/meminfo |grep MemTotal'  "$LINE" | sed '/SUCCESS/d' > "temp${var}/6" #operativ_memory
    ansible -i hosts.lin -m shell -a 'lshw -C memory |grep bank:0 -A 25 |grep description |grep -v empty'  "$LINE" | sed '/SUCCESS/d' > "temp${var}/7" #operativ_memory_description
    ansible -i hosts.lin -m shell -a 'lshw -class disk | grep product | head -n1' "$LINE" | sed '/SUCCESS/d' > "temp${var}/8" #hard_drive
```

```
	ansible -i hosts.lin -m shell -a ' dmidecode |grep "BIOS Information" -A 10 | grep Vendor -A 1'  "$LINE" | sed '/SUCCESS/d' |awk '{print }' ORS=' ' | awk '{print}' > "temp${var}/9" #BIOS
	ansible -i hosts.lin -m shell -a 'dmidecode |grep "Base Board" -A 10 |grep Manufacturer -A 1' "$LINE" | sed '/SUCCESS/d'  |awk '{print }' ORS=' ' | awk '{print}' > "temp${var}/10" #Mother_board
	ansible -i hosts.lin -m shell -a ' lshw | grep network -A 3 | grep product' "$LINE" | sed '/SUCCESS/d' > "temp${var}/11" #network_card
	ansible -i hosts.lin -m shell -a 'lshw'  "$LINE" | sed '/SUCCESS/d' | awk -v A=-1 '/DISABLED/ { A = 18 } A-- >= 0 {next } 1' | awk -v A=-1 '/UNCLAIMED/ { A = 4 } A-- >= 0 {next } 1' > "all${var}.txt"
	fi
done < hosts.lin
while read LINE
do
	if [[ $LINE =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
	let var=$var+1
	echo "$LINE"
	mkdir "temp${var}"
	ansible -i hosts.win -m win_command -a "ipconfig /all" " $LINE" | sed '/SUCCESS/d' | grep "Physical Address"| egrep -o ' [0-9A-F][0-9A-F]-[0-9A-F][0-9A-F]-[0-9A-F][0-9A-F]-[0-9A-F][0-9A-F]-[0-9A-F][0-9A-F]-[0-9A-F][0-9A-F]-[0-9A-F][0-9A-F]' |grep -v "00:00:00:00:00:00" > "temp${var}/1" #MAC_Address
	echo "$LINE" > "temp${var}/2"  #IP_addres
	ansible -i hosts.win -m win_command -a "systeminfo" " $LINE" | sed '/SUCCESS/d' |grep "Host Name:" -A 2 |tr -s ' ' > "temp${var}/3" #OS_bild_kernel
```

```
ansible -i hosts.win -m win_command -a "systeminfo" " $LINE" | sed '/SUCCESS/d' |grep "Processor(s):" -A 10 | grep -e [\[][0-9][0-9]\]: > "temp${var}/4"       #PROCESOR

ansible -i hosts.win -m win_command -a "wmic path win32_VideoController get name" " $LINE" | egrep -v -e "^$" -e "Name" -e "SUCCESS" > "temp${var}/5.dos"       #grafic_card

dos2unix -q "temp${var}/5.dos"

sed '/^$/d' temp${var}/5.dos >  "temp${var}/5"

ansible -i hosts.win -m win_command -a "systeminfo" " $LINE" | sed '/SUCCESS/d' |grep "Total Physical Memory"|egrep -v -e "^$" > "temp${var}/6" #operativ_memory

ansible -i hosts.win -m win_command -a "wmic MEMORYCHIP get DeviceLocator, MemoryType, Capacity, Speed, CreationClassName" " $LINE" | egrep -v -e "^$" -e "SUCCESS" > "temp${var}/7.dos" #operativ_memory_description

dos2unix -q "temp${var}/7.dos"

sed '/^$/d' temp${var}/7.dos >  "temp${var}/7"

ansible -i hosts.win -m win_command -a "wmic diskdrive get model,serialNumber,size,mediaType" " $LINE" | egrep -v -e "^$" -e "SUCCESS" -e "MediaType" > "temp${var}/8.dos" #hard_drive

dos2unix -q "temp${var}/8.dos"

sed '/^$/d' temp${var}/8.dos >  "temp${var}/8"

ansible -i hosts.win -m win_command -a "systeminfo" " $LINE" | sed '/SUCCESS/d' |grep "BIOS Version:"  > "temp${var}/9" #BIOS

ansible -i hosts.win -m win_command -a "wmic baseboard get product,Manufacturer,serialnumber" " $LINE" |grep -v -e "Manufacturer" -e "SUCCESS" -e "^$" > "temp${var}/10.dos" #Mother_board

dos2unix -q "temp${var}/10.dos"

sed '/^$/d' temp${var}/10.dos >  "temp${var}/10"
```

```
        ansible -i hosts.win -m win_command -a "systeminfo" " $LINE" | sed '/SUCCESS/d' |grep "Network Card(s):" -A 50 | grep -e [\[][0-9][0-9]\]:[" "][A-Z] > "temp${var}/11" #network_card
        ansible -i hosts.win -m win_command -a "systeminfo" " $LINE" | sed '/SUCCESS/d' > "all${var}.txt"
        fi
done <hosts.win
```

Додаток В. Вміст файлу basic_check.py

```
import smtplib
from email.mime.text import MIMEText
import os.path
from pathlib import Path
import os
import re
import filecmp
class Gmail(object):
    def __init__(self, email, password):
        self.email = email
        self.password = password
        self.server = 'smtp.gmail.com'
        self.port = 587
        session = smtplib.SMTP(self.server, self.port)
        session.ehlo()
        session.starttls()
        session.ehlo
        session.login(self.email, self.password)
        self.session = session
    def send_message(self, subject, body):
        ''' This must be removed '''
        headers = [
            "From: " + self.email,
            "Subject: " + subject,
            "To: " + self.email,
            "MIME-Version: 1.0",
           "Content-Type: text/html"]
        headers = "\r\n".join(headers)
```

```
        self.session.sendmail(
            self.email,
            self.email,
            headers + "\r\n\r\n" + body)
count = 1
while os.path.exists("temp"+str(count)):
        var=1
        while os.path.isfile("temp"+str(count)+"/"+str(var)):
                f1="temp"+str(count)+"/"+str(var)
                f2="temp.check"+str(count)+"/"+str(var)
                if not filecmp.cmp(f1, f2, shallow=False):
                        print("Warning !!! change in last oblick in List_of_all_inventarizateon!"+chr(65+count)+str(var)+":"+chr(65+count)+str(var))
                        gm = Gmail('2020.diplom.forsheets@gmail.com', 'smtp.gmail.com')
                        gm.send_message('Warning!!!', "change in List_of_all_inventarizateon!"+chr(65+count)+str(var)+":"+chr(65+count)+str(var))
                var+=1
        count+=1
```

Додаток С. Вміст файлу fill_invent_sheet.py

```
import httplib2
import apiclient.discovery
from oauth2client.service_account import ServiceAccountCredentials
import os.path
from pathlib import Path
import os
import re
import json
from apiclient import discovery
from apiclient.http import MediaFileUpload
from oauth2client import client
from oauth2client import tools
from oauth2client.file import Storage
CREDENTIALS_FILE = 'diplosheets-4490f33dbc26.json'
credentials =
ServiceAccountCredentials.from_json_keyfile_name(CREDENTIALS_FILE,
['https://www.googleapis.com/auth/spreadsheets',
'https://www.googleapis.com/auth/drive'])
httpAuth = credentials.authorize(httplib2.Http())
service = apiclient.discovery.build('sheets', 'v4', http = httpAuth) # Выбираем ра-
боту с таблицами и 4 версию API
with open('.proect_all_spreadsheet', 'r') as f:
    lines = f.read().splitlines()
    last_line = lines[-1]
eject_id=(last_line).split(' ')
spreadsheetId =  eject_id[0]
results = service.spreadsheets().values().batchUpdate(spreadsheetId =
spreadsheetId, body = { "valueInputOption": "USER_ENTERED", "data": [
```

```
{"range": "List_of_all_inventarizateon!A1:A15", "majorDimension": "ROWS", "values": [ ["MAC_ADDRESS"], ["IP_addres"], ["OS_bild_kernel"], ["Procesor"], ["Grafic_card"], ["Amount_of_RAM"], ["RAM_description"], ["Hard_drive"], ["BIOS"], ["Mother_board"], ["Network_card"], [""], [""], ["big_info"] , ["big_info"]]}]}).execute()
count = 1
while os.path.exists("temp"+str(count)):
    var=1
    while os.path.isfile("temp"+str(count)+"/"+str(var)):
        file = open("temp"+str(count)+"/"+str(var), "r")
        s=file.read()
        s = re.sub(" +", " ", s)
results = service.spreadsheets().values().batchUpdate(spreadsheetId = spreadsheetId, body = {"valueInputOption": "USER_ENTERED","data": [{"range": "List_of_all_inventarizateon!"+chr(65+count)+str(var)+":"+chr(65+count)+str(var),"majorDimension": "ROWS",  "values": [  [s]]}]}).execute()
        var+=1
        file.close()
    count+=1
count = 1
def get_grid_id(service):
  response = service.spreadsheets().get(spreadsheetId=spreadsheetId, includeGridData=True).execute()
  return response['sheets'][0]['properties']['sheetId']
def sheets_update(sheets_service, grid_id):
  body = {"requests": [{"autoResizeDimensions": {"dimensions": {"sheetId": grid_id,  "dimension": "COLUMNS"}  }}  ]}
```

```
    response = sheets_service.spreadsheets().batchUpdate(spreadsheetId=spreadsheetId, body=body).execute()
sheets_update(service, get_grid_id(service))
while os.path.isfile("all"+str(count)+".txt"):
    file = open("all"+str(count)+".txt", "r")
    lshw = file.read()
    results = service.spreadsheets().values().batchUpdate(spreadsheetId = spreadsheetId, body = {"valueInputOption": "USER_ENTERED","data": [{"range": "List_of_all_inventarizateon!"+chr(65+count)+"15:"+chr(65+count)+"15","majorDimension": "ROWS","values": [[lshw]]}]}).execute()
    count+=1
    file.close()
```

Додаток D. Вміст файлу fill_main.py

```
import httplib2
import apiclient.discovery
from oauth2client.service_account import ServiceAccountCredentials
import os.path
from pathlib import Path
import os
import json
from apiclient import discovery
from apiclient.http import MediaFileUpload
from oauth2client import client
from oauth2client import tools
from oauth2client.file import Storage
CREDENTIALS_FILE = 'diplosheets-4490f33dbc26.json'
credentials =
ServiceAccountCredentials.from_json_keyfile_name(CREDENTIALS_FILE,
['https://www.googleapis.com/auth/spreadsheets',
'https://www.googleapis.com/auth/drive'])
httpAuth = credentials.authorize(httplib2.Http())
service = apiclient.discovery.build('sheets', 'v4', http = httpAuth
file = open(".proect_all_spreadsheet", "r")
first_lint=(file.readline()).split(' ')
spreadsheetId =  first_lint[0]
file.close()
results = service.spreadsheets().values().batchUpdate(spreadsheetId =
spreadsheetId, body = { "valueInputOption": "USER_ENTERED", "data":
[{"range": "List_of_all_inventarizateon!A1:C1","majorDimension":
"COLUMNS","values": [ ["SHEET_№"] ,[ "LINK"] ,[
"TIME_OF_CREATION"]]}  ]}).execute()
```

```
f = open(".proect_all_spreadsheet", "r")
count_of_lines=2
first_cell_in_col=".this"
for line in f:
    if count_of_lines>=3:
        first_cell_in_col=str(count_of_lines-2)
    lst = line.split(' ')
    results = service.spreadsheets().values().batchUpdate(spreadsheetId = spreadsheetId, body = { "valueInputOption": "USER_ENTERED", "data": [{"range": "List_of_all_inventarizateon!A"+str(count_of_lines)+":C"+str(count_of_lines),"majorDimension": "COLUMNS","values": [[first_cell_in_col],['https://docs.google.com/spreadsheets/d/' + lst[0]] ,[ lst[1] ]]}]}).execute()
    count_of_lines+=1
f.close()
def get_grid_id(service):
  response = service.spreadsheets().get(spreadsheetId=spreadsheetId, includeGridData=True).execute()
  return response['sheets'][0]['properties']['sheetId']
def sheets_update(sheets_service, grid_id):
  body = {"requests": [{"autoResizeDimensions": {"dimensions": {"sheetId": grid_id, "dimension": "COLUMNS" }}}]}
  response = sheets_service.spreadsheets().batchUpdate(spreadsheetId=spreadsheetId, body=body).execute()
sheets_update(service, get_grid_id(service))
```

Додаток Е Вміст файлу new_spreadsheet.py

```
from pprint import pprint
import httplib2
import apiclient.discovery
import sys
import os.path
import googleapiclient.errors
from oauth2client.service_account import ServiceAccountCredentials
from datetime import datetime
current_datetime = datetime.now()
def htmlColorToJSON(htmlColor):
    if htmlColor.startswith("#"):
        htmlColor = htmlColor[1:]
    return {"red": int(htmlColor[0:2], 16) / 255.0, "green": int(htmlColor[2:4], 16) /
255.0, "blue": int(htmlColor[4:6], 16) / 255.0}
class SpreadsheetError(Exception):
    pass
class SheetNotSetError(SpreadsheetError):
    pass
if os.path.exists('.proect_all_spreadsheet'):
        title_spreadsheet='DIP_Invene_№ '+str((sum(1 for line in
open('.proect_all_spreadsheet', 'r'))))
        title_sheet='List_of_all_inventarizateon'
        rowCount=30
        columnCount=1000
else:
    title_spreadsheet='DIP_Main_sheet'
    title_sheet='List_of_all_inventarizateon'
    rowCount=1000
```

```
    columnCount=3
CREDENTIALS_FILE = 'diplosheets-4490f33dbc26.json'
credentials                                                    =
ServiceAccountCredentials.from_json_keyfile_name(CREDENTIALS_FILE,
['https://www.googleapis.com/auth/spreadsheets',
'https://www.googleapis.com/auth/drive'])
httpAuth = credentials.authorize(httplib2.Http())
service = apiclient.discovery.build('sheets', 'v4', http = httpAuth)
spreadsheet = service.spreadsheets().create(body = {
    'properties': {'title': title_spreadsheet, 'locale': 'en_US', 'timeZone': 'Etc/GMT'},
    'sheets': [{'properties': {'sheetType': 'GRID', 'sheetId': 0, 'title': title_sheet,
'gridProperties': {'rowCount': rowCount, 'columnCount': columnCount}}}]
}).execute()
spreadsheetId = spreadsheet['spreadsheetId']
driveService = apiclient.discovery.build('drive', 'v3', http = httpAuth)
access = driveService.permissions().create(
    fileId = spreadsheetId,
    body = {'type': 'user', 'role': 'writer', 'emailAddress': 'yura1veremij@gmail.com'},
fields = 'id' ).execute()
driveService = apiclient.discovery.build('drive', 'v3', http = httpAuth)
shareRes = driveService.permissions().create( fileId = spreadsheet['spreadsheetId'],
    body = {'type': 'anyone', 'role': 'reader'},  # доступ на чтение кому угодно
    fields = 'id' ).execute()
f = open('.proect_all_spreadsheet', 'a')
f.write(str(spreadsheetId) + " " + current_datetime.isoformat()+"\n")
f.close()
print('https://docs.google.com/spreadsheets/d/' + spreadsheetId+"\n")
```

Додаток F. Вміст файлу OS_sort

```
#!/bin/bash
var=0
rm -rf hosts.*
echo "[win]"> hosts.win
echo "[lin]" >hosts.lin
while read LINE
do
	if nmap -O $LINE |grep Windows; then
		echo "$LINE" >> hosts.win
		else
		echo "$LINE" >> hosts.lin
	fi
done < hosts
echo "[win:vars]
ansible_user='y'
ansible_password=y
ansible_connection=winrm
ansible_port=5986
ansible_port=5986
ansible_winrm_server_cert_validation=ignore" >> hosts.win
echo "[lin:vars]
ansible_user='root'" >> hosts.lin
```

Додаток G Вміст файлу program.sh

```
#!/bin/bash
echo "start of $0"
date
FILE=.proect_all_spreadsheet
if test -f "$FILE"; then
    echo "new inwentarization"
else
echo "new progect"
python3 new_spreadsheet.py
fi
python3 new_spreadsheet.py
python3 fill_main.py
rm -rf hosts.*
./OS_sort
./ansible_work.sh
python3 fill_invent_sheet.py
DIRECTORY="./temp.check1/"
if [[ -d "${DIRECTORY}" && ! -L "${DIRECTORY}" ]]; then
python3 basic_check.py
else
i=1
while test -d "./temp$i/"
do
cp -r "temp$i" "temp.check$i"
let i=i+1
done
fi
date
echo "end of $0"
```