Технічне завдання на створення сервісу файлового антивірусного захисту

(шифр – VScan API Platform)

Версія документу 2.1

**Київ -2024**

Зміст

[Зміст 2](#_Toc165369829)

[1. Загальні вимоги до сервісу 3](#_Toc165369830)

[2. Вимоги до сервісу 3](#_Toc165369831)

[2.1. Вимоги до backend частини. 3](#_Toc165369832)

[2.2. Вимоги до frontend частини. 3](#_Toc165369833)

[2.3. Вимоги до документації. 4](#_Toc165369834)

[3. Перелік бізнес процесів та алгоритм роботи сервісу 4](#_Toc165369835)

[Функції, які реалізує frontend : 4](#_Toc165369836)

[Керування користувачами. 4](#_Toc165369837)

[Керування налаштуваннями. 5](#_Toc165369838)

[Запис та перегляд статистики. 5](#_Toc165369839)

[Функції, які реалізує backend : 5](#_Toc165369840)

[Перевірка файлів 5](#_Toc165369841)

[Перевірка файлів 6](#_Toc165369842)

[Процес завантаження файлу та його доставки. 7](#_Toc165369843)

[4. Етапність виконання робіт та терміни їх виконання 8](#_Toc165369844)

# Загальні вимоги до сервісу

Для проекту МРІЯ, що охоплюватиме навчальні заклади України та надаватиме відповідний навчальний простір для учнів. У застосунку будуть відповідні шкільні та персональні чати, тому необхідно буде перевіряти контент, який надсилатиметься учнями, на предмет наявності шкідливих елементів у ньому. Контент може містити як посилання, так і файли. На перевірку файлів на першому етапі і розрахований сервіс, вимоги до якого описані в цьому документі.

Сервіс файлового антивірусного захисту (далі – сервіс VScan або просто сервіс) призначений для перевірки наданих по API файлів на предмет наявності в них шкідливого вмісту (вірусні компоненти, malware, бекдори та інше).

Сервіс надає REST API для завантаження файлів та перевірки їх на наявність шкідливих програм.

Користувач може надіслати файл на сервер через POST-запит. Авторизація користувачів обов’язкова і відбувається за допомогою попередньо виданого API ключа.

Сервіс аналізує файл і повертає результат перевірки (шкідливий вміст виявлено чи ні). В разі виявлення шкідливого вмісту сервіс повинен повертати інформацію про знайдену загрозу (назву, тип, рівень загрози).

Кодова база сервісу повинна бути створена з використанням сучасного фреймворку.

Повинна бути витримана єдина стилістика програмування (іменування класів, оформлення коду, його коментування).

По можливості слід уникати складних заплутаних рішень, віддаючи перевагу простим і ефективним рішенням. Складні елементи повинні бути прокоментовані. Програмний код сервісу має бути в достатній мірі коментованим для забезпечення прозорості в подальшій технічній підтримці та супроводі сервісу. При цьому коментуються неочевидні рішення або ж рішення, які можуть мати кілька варіантів написання із зазначенням чому обрано саме поточний (якщо вибрано не перше за своєю очевидністю рішення).

В якості СКБД рекомендується використання програмного забезпечення з відкритим кодом, яке не потребує сплати під час його використання.

# Вимоги до сервісу

## 2.1. Вимоги до backend частини.

Сервіс повинен бути високопродуктивним та масштабованим.

Взаємодія із сервісом повинна бути доступна лише за протоколом HTTPS.

Використання сервісу можливе лише за умови вдалої автентифікації.

Сервіс повинен коректно працювати при його використанні за відповідним обладнанням/сервісами захисту від DDoS атак (Firepower, Cloudflare, Imperva, Arbor).

## 2.2. Вимоги до frontend частини.

Frontend частина сервісу повинна бути крос-браузерною та коректно відображатися в сучасних браузерах: Chrome, Firefox. Для кожного браузера коректне відображення стосується останньої мажорної версії.

Допускається використання специфічних браузерних префіксів, якщо це необхідно для коректної верстки.

Кодування сервісу має бути UTF-8. DOCTYPE сервісу повинен бути HTML5.

Допускається використання сторонніх css-фреймворків, для яких є можливість модифікації стилів.

Використання нестандартних CSS або JS фреймворків повинне бути попередньо узгоджене із замовником.

Frontend частина розроблюється для десктопних пристроїв і, в свою чергу, адаптивно налаштовується для мобільних пристроїв за розміром екрану.

Мова інтерфейсів – українська, за виключенням тексту відповідей на API запити.

## 2.3. Вимоги до документації.

Повинна бути створена докладна документація щодо використання API (ендпоінти, параметри, приклади запитів та відповідей).

Мова документації – англійська.

Допускається автоматична генерація документації API шляхом використання відповідних інструментів (Swagger, pdoc3, Scribe, Slate та інші).

Відповідна документація повинна бути достатньою для використання сервісу сторонніми організаціями.

# Перелік бізнес процесів та алгоритм роботи сервісу

Авторизація користувачів відбувається по логіну/паролю. При цьому доступ адміністраторів до веб-інтерфейсу може бути обмежено відповідним переліком IP адрес із налаштувань.

## Функції, які реалізує frontend :

### Керування користувачами.

В цілому користувачі можуть бути 2 категорій:

* Адміністратори (вносять зміни в налаштування системи, керують користувачами)
* Зовнішні користувачі (клієнти, що користуються функціями зовнішньої частини системи)

Адміністратори можуть здійснювати керування користувачами (CRUD): їх атрибутами, правами (адміністратор, зовнішній користувач), API ключами (відповідні ключі можуть бути і в адміністратора).

На першому етапі зовнішньому користувачу доступний процес логіну в кабінет та здійснення наступних операцій:

* Зміна паролю
* Генерація API ключа
* Зміна ПІБ користувача
* Зміна посади користувача
* Зміна контактного телефону

Створення зовнішнього користувача покладається на адміністратора. Тобто публічного процесу реєстрації немає.

Атрибути користувача:

* пошта (вона ж і є логіном)
* пароль
* роль
* API ключ (1 користувач може мати до 3 ключів)
* ПІБ
* Посада користувача
* Контактний телефон

### Керування налаштуваннями.

Адміністратори можуть змінювати налаштування системи через відповідну сторінку.

Доступні налаштування frontend у:

|  |  |  |
| --- | --- | --- |
| Параметр | Значення | Пояснення |
| ip\_allowed\_access | Перелік IP адрес розділених комою, наприклад:  12.12.12.0/24, 11.11.11.11/32 | Перелік IP адрес із яких дозволений доступ до веб інтерфейсу frontend у. При вказанні 0.0.0.0/0 доступ дозволено із будь-якої адреси |
| Allowed\_file\_types | Text | Набір розширень файлів, що перераховані через кому у форматі .[розширення]. Наприклад: .doc, .docx, .xls, .xlsx |
| Allow\_macroses | Boolean | Чи дозволене використання офісних документів із макросами |
| Macros2sandbox | Boolean | Чи відправляти документи з макросами **примусово** до пісочниці |
| Virustotal\_api\_key |  | API ключ сервісу VirusTotal |
| Email\_appliance\_address |  | Адреса поштового серверу, що буде перевіряти файли |

### Запис та перегляд статистики.

При перевірці файлів система пише відповідну статистичну інформацію в окрему базу даних. За кожну годину пишуться наступні дані:

* Кількість перевірених файлів із привʼязкою до клієнта та результату перевірки файлу
* Кількість використаних запитів до сторонніх сервісів за годину

Адміністратори можуть переглядати статистику використання запитів до різних платформ, передбачених системою. Зокрема доступна статистика наступних типів:

|  |  |  |
| --- | --- | --- |
| Тип статистики | Доступні часові проміжки | Опис даних |
| Використання запитів до clamav кластеру | Поточна доба, останні 7 днів, останній місяць | Вивід відповідної кількості виконаних запитів до поштового серверу |

## Функції, які реалізує backend :

### Перевірка файлів

Backend частина дозволяє зовнішньому авторизованому користувачу відправити файл на перевірку.

Авторизація відбувається за допомогою API ключа, що передається в хедері запиту.

Результатом перевірки є відповідь сервісу чи міститься у наданому файлі ознаки шкідливого коду.

Технічні деталі:

* + **Endpoint**: /api/v1/scan/file
  + **Метод**: POST
  + **Параметри запиту**:
    - file: файл для перевірки (multipart/form-data)
  + **Параметри відповіді**:
    - status: статус перевірки (успішно або заражено)
    - message: опис результату (наприклад, “Файл без вірусів” або “Виявлено вірус: Trojan.Win32.Generic”)
  + **Приклад відповіді**:
  + {
  + "status": "success",
  + "message": "Файл без вірусів"
  + }
  + **При виявленні зловмисного шкідливого:**
  + {
  + "status": "infected",
  + "message": "Виявлено вірус: Trojan.Win32.Generic"
  + }

Алгоритм перевірки файлів

### Перевірка файлів

Backend частина дозволяє зовнішньому авторизованому користувачу відправити файл на перевірку.

Щоб зменшити загальне навантаження на систему і забезпечити надійність детектування пропонується для використання багаторівневу модель детектування можливого шкідливого вмісту, а саме:

* перевірка розширення файлів (якщо не в списку дозволених - приймання файлу блокується)
* підрахунок файлового хешу та його перевірка (на цьому етапі перевіряється лише чи був такий файл раніше і яка відповідь по ньому)
* відправка файлу на перевірку до поштового серверу, а саме Cisco Secure Email Gateway (далі - SEG)

Нижче деталізовано кожний із цих кроків:

#### Перевірка розширення файлу.

Перевіряється поточне розширення файлу та порівняння його із списком дозволених із відповідних налаштувань в адміністративній частині. Якщо поточне розширення файлу відповідає дозволеним. То файл відправляється на подальшу обробку. Якщо ні, то повертається відповідь із кодом 403 і відповідним інформаційним повідомленням.

#### Підрахунок файлового хешу.

При отриманні файлу рахуємо його файловий хеш SHA256. Отриманий хеш порівнюється із існуючою базою даних сервісу. Якщо такий хеш вже був в базі, то відправляємо клієнту відповідний результат обробки файлу із бази даних. Зазначений механізм буде особливо ефективним враховуючи природу активності шкільних чатів (наявність повторюваних файлів, що пересилаються інтенсивно протягом короткого періоду часу). Якщо хеша в базі немає, то ми його записуємо в базу (після отримання результатів перевірки).

#### Відправка файлу на перевірку до поштового серверу

Для основної перевірки файлу він відправляється на SEG на перевірку. Поштовий сервер працює як комбіноване рішення, тобто самостійно спершу перевіряє файловий хеш файлу на предмет його наявності у глобальних базах, далі перевіряє відповідність типу файлу його розширенню, далі файл йде на перевірку антивірусними засобами. Якщо файл містить ознаки шкідливого, але не детектується явним чином, то він відправляє файл до пісочниці на перевірку. Налаштування логіки пересилки у пісочницю відбуваються на стороні SEG, механізми пересилки та отримання відповіді від пісочниці також на стороні SEG.

### Процес завантаження файлу та його доставки.

Вище було розглянуто безпосередньо процес перевірки файлів, надалі описано процеси до моменту перевірки та після.

Процес завантаження файлу виглядає наступним чином:

- користувач відправляє файл в чат

- здійснюється початкова перевірка розширення файлу засобами самого месенджеру (якщо таке розширення заборонене, то файл не завантажується)

- дозволений файл тимчасово зберігається в окреме сховище (файлова система без біту виконання)

- файл відправляється на перевірку

- для початкової перевірки файлу здійснюється затримка його доставки на термін до 30 секунд. Ці 30 секунд система витрачає на перевірку файлу по внутрішній базі даних хешів і у випадку відсутності хешу - очікує відповідь від поштового серверу. Якщо перевірка триває більше 30 секунд – тоді файл відправляється у чат, але після отримання результатів перевірки – система приймає рішення стосовно подальших дій із файлом. У разі його зловмисного вмісту – він видаляється із чатів (реалізується на стороні розробників додатку МРІЯ).

- за результатами перевірки він отримує вердикт. Якщо він шкідливий, то файл із поміткою користувача, що його завантажив та іншими мета даними відправляється в карантин. Якщо файл не містить ознак наявності шкідливого вмісту, то він переміщується в перевірене сховище, а посилання на нього зʼявляється в чаті.

На поточному етапі питання генерації посилання на файл та його переміщення між сховищами покладено на розробників месенджеру.

# Етапність виконання робіт та терміни їх виконання

#### Етапність робіт.

Система повинна бути реалізована у 2 етапи:

1 етап – розробка MVP – minimum viable product.

2 етап – доробка налаштувань та можливість підключень інших користувачів.

1 етап повинен включати в себе реалізацію функцій backend у та створення документації.

Тобто на першому етапі один зовнішній користувач із заздалегідь визначеним API ключем може здійснювати перевірки файлів. На цьому етапі немає налаштувань, керування користувачами, статистики. Тобто функцій frontend немає. Лише отримання файлів, перевірка локального хешу, відправка файлу на SEG, отримання відповіді від SEG і повернення цієї відповіді користувачу.

2 етап включає в себе доопрацювання frontend у (додавання функцій frontend у в межах описаних в цьому документі).

#### Орієнтовні терміни виконання.

Термін виконання MVP – 27.05.2024

Термін бета версії 2 етапу – 02.08.2024