ШИФР РОБОТИ

SAFEBOT

НАЗВА РОБОТИ

РОЗРОБКА ПЛАТФОРМИ ДЛЯ ПЕРЕВІРКИ ФАЙЛІВ НА НАЯВНІСТЬ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Львів 2021

АНОТАЦІЯ

ШИФР РОБОТИ

SAFEBOT

НАЗВА РОБОТИ

РОЗРОБКА ПЛАТФОРМИ ДЛЯ ПЕРЕВІРКИ ФАЙЛІВ НА НАЯВНІСТЬ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Актуальність. У наш час в наслідок пандемії Covid-19 людство стикнулось з однією з найбільших реформацій – діджиталізацією. Цей процес відбувався уже давно, але саме внаслідок пандемії цей процес прискорився в рази, миттєво перевівши в онлайн більшу частину бізнесу, навчання та навіть розваг. Це, звичайно, дає багато нових можливостей і є безперечним кроком вперед. Проте з різким збільшенням онлайн активності, збільшується активність зловмисників, котрі використовують усі можливі методи для викрадення чужих даних, або руйнування працездатності систем. Найпростіший метод – поширення вірусного програмного забезпечення під виглядом звичайних файлів. А поширення файлів відбувається через файлообмінники та месенджери.

Саме тому **метою роботи** є створення автоматизованої платформи, яка буде виконувати роль «пісочниці» та віддалено перевіряти файли, які передаються у месенджері Telegram на наявність шкідливого програмного забезпечення.

Для вирішення цієї мети були виконані такі **завдання**: розроблення платформи, яка буде розгортати віртуальну машину на сервері, отримуючи дані через Telegram, скануватиме отримані файли на наявність шкідливого програмного забезпечення та надсилатиме відповідь користувачеві у доступній формі. Крім цього розгортатиметься веб-сервер, котрий дасть можливість ізольовано переглядати файли через веб-браузер без прямого доступу до них. Ця розробка є повністю автоматизована і не вимагає жодних

додаткових дій, а також дає змогу економити ресурси власного пристрою.

Об'єкт дослідження – автоматизована платформа за перевірки файлів на наявність шкідливого програмного забезпечення.

Наукова новизна полягає у застосуванні «пісочниці», тобто ізольованого механізму для безпечного виконання програм до Telegram месенджера, що дає можливість перевіряти підозрілі файли на наявність шкідливого програмного забезпечення, без негативних наслідків для користувача.

Практична цінність роботи полягає у масовому застосуванні розробки серед користувачів Telegram, як ресурсу для перевірки підозрілих файлів та посиланнь для уникнення зараження вірусами та шкідливим програмним забезпеченням. Ця розробка є повністю автоматизована і не вимагає жодних додаткових дій зі сторони користувача, а також дає змогу економити ресурси власного пристрою користувача.

Наукова робота складається з текстової частини, що містить три розділи, загальний обсяг 30 с., 32 рис., 10 джерел літератури.

Ключові слова: пісочниця, Telegram бот, захист від шкідливого програмного забезпечення.

Інформація про наукові публікацій (за наявністю).

Тези конференції – 1

ЗМІСТ

РОЗДІЛ 1. ПРОБЛЕМАТИКА ПОШИРЕННЯ ВІРУСІВ В МЕСЕНДЖЕРІ TELEGRAM

З розвитком сучасних технологій Інтернет перейшов у мобільні телефони, що дало можливість практично весь час перебувати «online» і, відтак, соціальні комунікації набули нових особливостей.

З розвитком Інтернету, з'явилась можливість використовувати всі його досягнення, одним з яких є месенджери. З кожним роком популярність месенджерів збільшується, вони стають невід'ємним атрибутом сучасного життя, зростає кількість постійних користувачів. При цьому, саме ця нова форма комунікації стає одним із актуальних напрямків наукових досліджень.

Найпопулярнішим месенджером на даний момент є Telegram. Він завоював всесвітню любов через свою анонімність та відсутність цензури. Що також є проблемою, оскільки не можливо проконтролювати файли, які передаються.

1.1. Статистичні дані месенджера Telegram

Telegram – це програмне забезпечення для смартфонів, планшетів та комп'ютерів, яке дає змогу обмінюватися текстовими повідомленнями, графічними та відеофайлами, а також безкоштовно телефонувати іншим користувачам програми.

Загалом, Telegram є більш надійною, ніж інші, про це інформують світові компанії, що займаються питаннями кібербезпеки, такі як San, Isaca, Enisa та інші. Зокрема, аналітичний центр Falcongaze, компанія-розробник програмних рішень в галузі запобігання витоків даних по різних каналах (електронна пошта, соціальні мережі, месенджери, USB і ін.), склала рейтинг найбільш популярних месенджерів залежно від рівня їх захищеності (рис. 1.1) [1]. На першому місці Telegram.

Рис. 1.1.Рейтинг найбільш захищених популярних месенджерів

За даними всесвітнього порталу statista.com, цією платформою користуються значне число користувачів. На рис. 1.2 наведено діаграму росту числа користувачів з 2014 – 2020 роки [2].

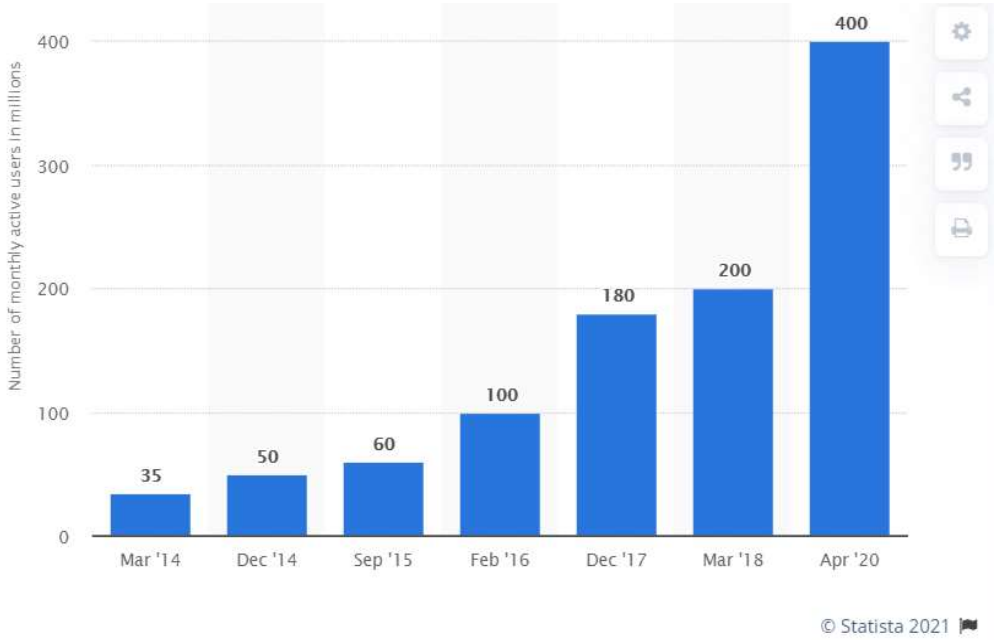

Рис. 1.2. Статистика кількості активних користувачів Telegram

Telegram відомий як приватна, а головне безпечна служба обміну повідомленнями, оскільки вона автоматично шифрує всі повідомлення та пропонує такі функції, як самознищення облікового запису, якщо користувачі не затримуються протягом певного періоду часу.

1.2. Віруси у Telegram

Кібершахраї завжди знаходять способи використовувати, навіть здавалося найбільш захищену платформу Telegram для нанесення шкоди користувачам. Мова йде про пересилання файлів з вірусами.

Вірус у Telegram – це можлива загроза, яку можна розповсюдити за допомогою програми обміну повідомленнями або навіть представляти як саму програму. Цей термін використовується для опису нової кіберзагрози, яка створюється для багатьох цілей. Хакери розповсюджують шкідливе програмне забезпечення за допомогою цілком законного програмного забезпечення для обміну повідомленнями на основі хмари [3].

Мета цієї інфекції поширити майнер криптовалюти [4], або проникнути в систему шпигунським програмним забезпеченням та іншими потенційно небажаними програмами та отримати дистанційне керування пристроєм. На жаль, ідентифікувати вірус важко, оскільки він приховує свою присутність у системі.

Зрозуміло, що ресурс Telegram сам по собі не має нічого спільного з цими шахрайствами та шкідливим програмним забезпеченням, це просто зручна платформа, яка використовується для розповсюдження інфекцій.

Як тільки зловмисне програмне забезпечення потрапляє в систему, воно осідає в найглибших місцях на смартфоні чи комп’ютері і починає працювати у фоновому режимі.

На основі аналізу літературних джерел, одне з найпоширеніших шкідливих програм, вважається поширення через Telegram майнерів криптовалют [3-4]. Ці інфекції не є особливо серйозними, оскільки зазвичай не завдають шкоди комп’ютеру. Вони головним чином зосереджуються на використанні ресурсів вашого комп’ютера для видобутку таких криптовалют, як Monero, ZCash тощо. Окрім використання ресурсів комп’ютера, це призведе до довгострокової шкоди вашій системі. Однак майнінг спричиняє короткочасні проблеми, особливо пов’язані з роботою комп’ютера. Ваша система стане млявою, програми

постійно завантажуватимуться і будуть аварійно завершувати роботу, а у диспетчері завдань з'являться дивні процеси з великим використанням процесора. Це все ознаки криптовалютного зловмисного програмного забезпечення. На щастя, оскільки це так помітно, якщо ваш комп'ютер заразиться, ви відразу про це дізнаєтесь.

На думку експертів, конкретний майнер Telegram може мати додаткові функції до основного, включаючи відкриття задніх дверей для проникнення в систему іншими потенційно небажаними програмами (PUP) або використання пульта дистанційного керування для прямої установки шкідливих програм. Таким чином, якщо ваш комп'ютер заражений, ви ризикуєте своїми приватними даними, обліковими даними та важливою інформацією.

Інше небезпечне шкідливе програмне забезпечення, яке ви можете отримати на таких платформах, як Telegram, – це трояни, які можуть шпигувати за вами та викрадати вашу інформацію. Троянські програми також можуть діяти як бекдор і дозволяти іншому шкідливому програмному забезпеченню встановлюватись. Більшість антивірусних програм зможуть виявляти та видаляти більшість вірусів Telegram, тому поки у вас встановлена якась програма захисту, вона повинна мати можливість захистити ваш комп'ютер [3].

На основі статистичних даних експертів антивірусної компанії Zillya! за 2018-2019 років в Україні відзначають зростання частоти використання відомих троянських програм та їх модернізованих варіантів [5].

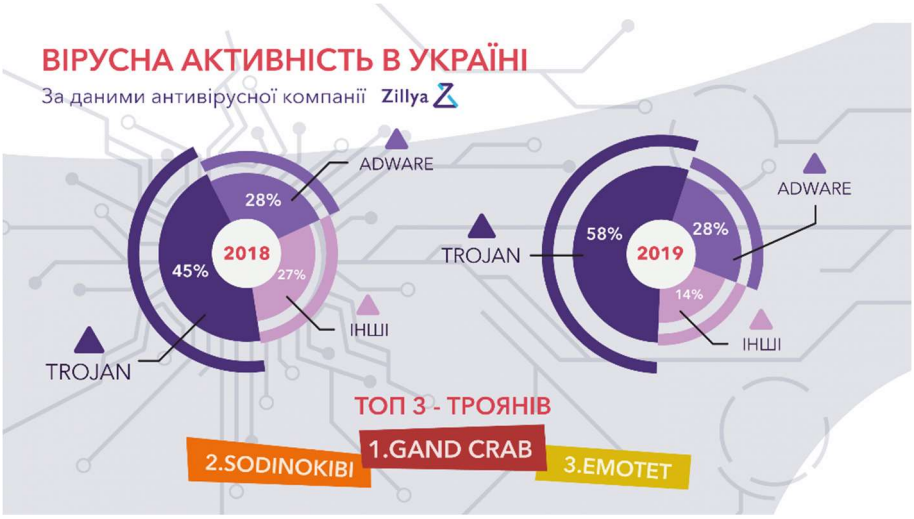

Рис. 1.3. Статистика вірусної активності в Україні

РОЗДІЛ 2. ВИРІШЕННЯ ПРОБЛЕМИ

2.1. Алгоритм розробки платформи

Враховуючи інформацію описану у розділі 1, нами прийнято рішення розробити платформу, яка дає змогу перевіряти надіслані файли на наявність шкідливого програмного забезпечення.

Для реалізації ізольованого механізму для безпечного виконання програм, який називають «Пісочниці» чи «Sandbox» [6, 7] нам потрібно розгорнути віртуальну машину, встановити на неї антивірус та налаштувати можливість скачування файлів. Для завантаження файлів до віртуальної машини було обрано популярний месенджер Telegram, оскільки він є досить популярним сьогодні (див. рис. 1.1). Так як, Telegram є анонімним файлообмінником, існує велика загроза скачати заражений файл, надісланий зловмисником. Можливість взаємодії Telegram з «Пісочницею» дає змогу перевірити отримані файли, переславши їх боту не відкриваючи їх [7]. В якості посередника між користувачем та «Пісочницею» буде платформа (далі – бот), який перенаправлятиме файли безпосередньо у віртуальну машину, а також надавати користувачеві кінцеву інформацію про результати сканування.

Логіка роботи полягає у послідовній розробці кроків, що описано на рис. 2.1.

Крок 1. Початком роботи бота будемо вважати отримання файлу в повідомленні від користувача. Слід зазначити, що бот реагує на переслані повідомлення з файлом аналогічно вкладеним повідомленням напряму від користувача. Цей нюанс є дуже важливим, адже дозволяє переслати повідомлення від зловмисника боту, не скачуючи файл чи не переходячи по посиланню.

Після чого бот отримає дане повідомлення, визначить тип повідомлення (текстове, медійне, аудіо, файлове, та ін.). Цей етап потрібний, оскільки боту потрібно знати, з яким типом даним йому прийдеться працювати.

Наступним кроком буде, формування прямого посилання на файл. Цей крок необхідний для скачування файлу надісланого користувачем.

Крок 2. Маючи пряме посилання на файл, бот виконує розгортання Docker

Container з встановленням необхідного програмного забезпечення, за допомогою якого буде відбуватись перевірка файлу на наявність вірусів.

Рис. 2.1. Алгоритм роботи

Крок 3. Після сканування, ми записуємо файл логування (файл з даними про виконання попередніх команд), з якого формуємо відповідь клієнту про стан файла. Це триває приблизно 16 секунд. Відповідь буде надіслана користувачеві одразу по закінченню сканування.

Крок 4. Після сканування файлу ми переходимо до розгортання веб-сервера WebDav. Встановлюємо на нього файли користувача, робимо це теж у віртуальній машині для ізоляції файлу від робочої мережі сервера.

Після чого формуємо посилання та надаємо його користувачеві для інтерактивного доступу до файлу через веб-браузер.

Крок 5. Надсилання відповіді з результатом сканування.

Описаний алгоритм є повністю автоматизованою, це означає, що користувач не буде бачити процесу розгортання віртуальної машини та сканування файлу, а лише відповідь бота.

Підсумовуючи вищеописане, на рис. 2.2 наведено архітектуру робочої директорію проекту.

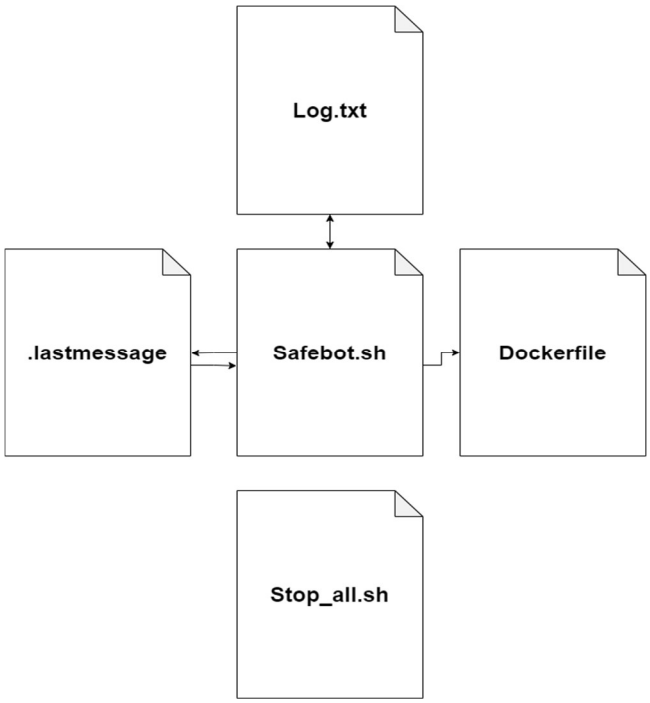

Рис. 2.2. Архітектура робочої директорії проєкту

Safebot.sh – головний файл, який керує усіма процесами, запускає програму, генерує пряме посилання на файл, отриманий ботом від користувача, створює віртуальну машину, запускаючи Dockerfile. Також даний файл керує всією логікою нашого бота: збирає усю потрібну інформацію про файл, користувача від якого ми отримали повідомлення; надсилає користувачеві повідомлення з інформацією про стан файла.

Dockerfile – створює контейнер на базі операційної системи Kali Linux, інсталює антивірусне програмне забезпечення, отримує інформацію від Safebot.sh, який файл встановлювати на віртуальну машину.

Stop_all.sh – зупиняє та видаляє усі створені контейнери в ручному режимі. Корисний коли система переповнюється зайвим кешом, який витрачає операційну та фізичну пам'ять.

Log.txt – записує логи виконання усіх дій, містить інформацію про процес розгортання віртуальної машини, сканування файла та можливі помилки в ході виконання програми.

.**lastmessage** – записує дані про останнє повідомлення, щоб співставляти їх кожних 3 секунди з останнім оновленням для відслідковування нових(необроблених) сповіщень. Оновлюється після виконання усіх команд призначених для отриманого повідомлення.

Для практичного ознайомлення з нашою розробкою потрібно перейти за URL адресою: **https://t.me/sandboxxxxbot.** Важливо, перейти за посиланням потрібно з того пристрою, на якому є встановлено Telegram.

2.2. Процес розробки платформи

2.2.1. Створення Віртуальної машини, як основи для «Пісочниці»

На сьогодні відомо багато різновидів віртуальних машин, таких як: VMware, Docker Container Microsoft Hyper-V, Oracle VM VirtualBox, Parallels Desktop, Quick Emulator, Xen Docker Container та інші.

Для реалізації «Пісочниці» найбільше підходить Docker Container, оскільки він є швидким, підтримує усі відомі операційні системи, є простим у використанні та безкоштовним.

Нижче наведено список основних можливостей Docker Container [8]:

- можливість розміщення в ізольованому оточенні різноманітних функцій, що включають різні комбінації виконуваних файлів, бібліотек, файлів конфігурації, скриптів, файлів jar, gem, tar, тощо;
- ізоляція на рівні файлової системи: кожен процес виконується у повністю окремій кореневій файловій системі;

- ізоляція ресурсів: споживання системних ресурсів, таких як витрата пам'яті і навантаження на CPU, можуть обмежуватися окремо для кожного контейнера з використанням cgroups;
- ізоляція на рівні мережі: кожен ізольований процес має доступ тільки до пов'язаного з контейнером мережевого простору імен, включаючи віртуальний мережевий інтерфейс і прив'язані до нього IP-адреси;
- та інші.

Для створення віртуальної машини на базі Docker нам необхідно завантажити Docker-образ, який сформує контейнер.

Docker-образ – це read-only шаблон. Наприклад, образ може містити операційну систему Ubuntu з Apache і додатком на ній [9]. Образи використовуються для створення контейнерів. Docker дає змогу легко створювати нові образи, оновлювати існуючі, або ви можете завантажити образи створені іншими людьми. Образи – це компонента збірки docker-a.

Docker-образи можуть створитися з базових образів, кроки опису для створення цих образів називають інструкціями. Кожна інструкція створює новий образ. Інструкціями є наступні дії:

- запуск команди;
- додавання файлу або директорії;
- створення змінної оточення;
- вказівки що запускати коли запускається контейнер цього способу.

Ці інструкції зберігаються в файлі Dockerfile.

Dockerfile – це текстовий документ, який містить усі команди, які користувач може викликати в командному рядку для складання зображення [10]. Користувачі збірки докерів можуть створювати автоматизовану збірку, яка послідовно виконує кілька інструкцій командного рядка (рис. 2.3).

```
FROM kalilinux/kali-rolling
RUN mkdir /test
WORKDIR /test
ARG FILEURL
ENV FILEURL=${FILEURL}


RUN apt-get update && apt-get install -y \
  wget \
  clamav \
  clamav-daemon \
  && rm -rf /var/lib/apt/lists/*

RUN freshclam
```

Рис. 2.3. Створення Dockerfile

Інструкції, описані вище дають змогу розгорнути Docker-образ на базі операційної системи Kali Linux.

Kali Linux – це дистрибутив Debian-похідних Linux, призначений для цифрової криміналістики і тестування на проникнення [11].

У цьому Dockerfile нами були використані такі інструкції:

- FROM – задає базовий (батьківський) образ;
- RUN – виконує команду і створює шар образу. Використовується для встановлення в контейнер пакетів. А також виконує будь-яку Shell-команду.
- WORKDIR – задає робочу директорію для наступної інструкції.
- ARG – задає змінні для передачі Docker в процесі побудови образу;
- ENV – встановлює постійні змінні середовища.

В даному випадку інструкцією FROM описуємо операційну систему, що буде працювати на основі вже створеного Docker-image Kali Linux. Після чого інструкцією RUN оголошуємо shell-команду mkdir, котра створює папку “test” в корінній директорії. Одразу оголошуємо її робочою директорією, тобто такою, у яку будуть потрапляти усі скачані файли, або з якої будуть виконуватись усі задані команди. Також важливим моментом є оголошення змінної інструкціями ARG та ENV, котру ми передаємо з головного скрипта, котрий запускатиме Dockerfile. Останнім кроком є оновлення пакетів за допомогою пакетного менеджера apt-get. Після успішного оновлення пакетів переходимо до встановлення наступних інструментів:

- Wget – неінтерактивна консольна утиліта для завантаження файлів за протоколами HTTP, HTTPS та FTP.
- Clamav-Clam AntiVirus – пакунок антивірусного ПЗ, що працює у багатьох операційних системах, включаючи Unix-подібні ОС, Microsoft Windows та Apple Mac OS X. Детальніше про цей антивірус описано в наступному підпункті.

Процес створення образу віртуальної машини наведено на рис. 2.4.

```
Sending build context to Docker daemon  97.79kB
Step 1/9 : FROM kalilinux/kali-rolling
latest: Pulling from kalilinux/kali-rolling
4403fb8989b2: Already exists
Digest: sha256:808e5f9be6e92f6c8135a4461cbd0b2fbbb52c6d022d9e0b2527764a0eb11ac9
Status: Downloaded newer image for kalilinux/kali-rolling:latest
 ---> 7102d9904bc5
Step 2/9 : RUN mkdir ~/test
 ---> Running in db01d2bdf813
Removing intermediate container db01d2bdf813
 ---> 42aba9cf1c7b
Step 3/9 : WORKDIR ~/test
 ---> Running in 19bf4644ce3a
Removing intermediate container 19bf4644ce3a
 ---> 780fce4259a0
Step 4/9 : ARG qqq
 ---> Running in e749851e55f2
Removing intermediate container e749851e55f2
 ---> aec36ac321f8
Step 5/9 : ENV qqq=${qqq}
 ---> Running in 4f363eeb0f5c
Removing intermediate container 4f363eeb0f5c
 ---> 27828c00e31d
Step 6/9 : RUN apt-get update && apt-get install -y   wget   clamav   clamav-daemon   && rm -rf /var/lib/apt/lists/*
 ---> Running in 0183a193f50d
Get:1 http://kali.koyanet.lv/kali kali-rolling InRelease [30.5 kB]
Get:2 http://kali.koyanet.lv/kali kali-rolling/contrib amd64 Packages [106 kB]
Get:3 http://kali.koyanet.lv/kali kali-rolling/non-free amd64 Packages [203 kB]
Get:4 http://kali.koyanet.lv/kali kali-rolling/main amd64 Packages [17.5 MB]
Fetched 17.8 MB in 12s (1517 kB/s)
Reading package lists...
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
```

Рис. 2.4. Процес створення образу віртуальної машини через Dockerfile командою “sudo docker build”

Процес запуску віртуальної машини та передання до неї отриманого файлу реалізовано на рис. 2.5.

```
[[ $(docker ps -f "name=cont$USERNAME" --format '{{.Names}}') == cont$USERNAME ]] || docker run -d -t --name cont$USERNAME clamav:latest
docker exec -e FILEURL=$FILEURL cont$USERNAME wget -O 1.docx $FILEURL  > /dev/null 2>&1
docker exec cont$USERNAME clamscan 1.docx > log.txt
```

Рис. 2.5. Запуск Docker Container

Зазначена частина коду складається з набору команд docker-клієнта, які дають змогу управляти та налаштовувати контейнери, в котрих будемо завантажувати та перевіряти файли.

Перший рядок перевіряє, чи немає ще такого контейнера. Логіка створення контейнерів для різних клієнтів є наступною: для кожного користувача створюється окремий контейнер, де будуть перевірятись лише файли даного користувача. В разі виявлення вірусного файлу контейнер негайно видаляються, а також його кеш. Це дасть можливість унеможливити втручання вірусного програмного забезпечення в роботу віртуальної машини. Виходячи з цього, перевірка нам потрібна для того, щоб розмежовувати віртуальні машини та уникнути будь-яких конфліктів.

Наступним пунктом є створення контейнера. Ми присвоюємо контейнеру ім'я користувача, щоб чітко розрізняти, де знаходяться файли того чи іншого користувача.

Командою docker exec ми входимо в створений контейнер, передаємо йому параметри файла, який йому потрібно завантажити, та виконуємо завантаження файла.

Останнім кроком є сканування отриманого файла та запис результатів сканування в файл log.txt котрий створюється в директорії, з якої був запущений бот.

2.2.2. Розгортання Веб-сервера Webdav

Web Distributed Authoring and Versioning, WebDAV – це протокол для передачі даних і роботи з ними, побудований поверх HTTP 1.1 [12]. Тут варто зауважити, що передача може бути як захищеною, так і незахищеною. У самому протоколі захист відсутній, але він може бути доданий через реалізацію аутентифікації на веб-сервері і шифрування за допомогою SSL, отже, в такому випадку буде використовуватися не HTTP, а HTTPS.

Webdav є альтернативою протоколу FTP, проте має рад переваг, а саме:

- працюючи через одне з'єднання TCP, простіше налаштувати його на обхід брандмауерів, NATs і проксі-серверів. В FTP канал передачі даних може викликати проблеми з правильним налаштуванням NAT.
- WebDAV є трохи швидше, ніж FTP при передачі багатьох невеликих

файлів-немає необхідності створювати з'єднання для передачі даних для кожного файлу.

- стиснення GZIP є стандартом для HTTP, але не для FTP.
- HTTP має широкий вибір методів аутентифікації, які не визначені в FTP. наприклад, NTLM і аутентифікація Kerberos поширена в HTTP, а в FTP важко отримати належну підтримку для них, якщо ви не пишете як клієнтську, так і серверну боку FTP.
- WebDAV підтримує часткову передачу, а в FTP часткове завантаження неможлива (Ви не можете перезаписати блок в середині файлу).

У нашому випадку Webdav буде служити транслятором файлів від нашого сервера до клієнта. Цей функціонал дає змогу використати хмарне зберігання та передачу файлів для безпечного відображення контенту. Це дасть можливість демонструвати користувачеві надісланий файл, без безпосереднього доступу до самого файлу. В такому випадку навіть якщо файл виявиться зараженим, він не зможе втрутитись в роботу комп'ютера користувача, оскільки фізично буде розташований на ізольованій віртуальній машині на сервері.

Тобто користувач отримає унікальне посилання на веб-сторінку, котра буде транслювати список файлів, які він надіслав боту. Дане посилання можна буде відкрити в будь-якому браузері. Також буде доступна можливість переглянути та скачати цей файл (проте це не рекомендується, адже втрачається головна мета даного веб-сервісу).

Розгортання Webdav буде реалізовано в Docker Container (рис. 2.6). Для кожного користувача буде створено окремий контейнер. Це дозволить розмежувати файли різних користувачів. На кожен контейнер буде встановлено унікальний пароль, який буде надіслано користувачеві ботом.

```
docker run --restart always -v /srv/dav:/var/lib/dav -e AUTH_TYPE=Digest -e USERNAME=alice -e PASSWORD=secret1234  /
--publish 80:80 --name webdav$USERNAME -e LOCATION=/webdav -d bytemark/webdav
docker exec webdav$USERNAME wget $FILEURL -O /var/lib/dav/data/$DOCUMENTNAME
```

Рис. 2.6. Запуск веб-сервера Webdav у Docker Container

Розглянемо детальніше використані команди:

- *Docker run* запускає контейнер з наступними параметрами:

- *–restart always* проводить оновлення контейнера, що забезпечує вчасну реакцію на додані файли*;*
- *–v /srv/dav:/var/lib/dav* монтує папку, в яку завантажується файл та надсилає його в вебсервер*;*
- *–e AUTH_TYPE-Digest …* реалізовує автентефікацію за наведеними параметрами*;*
- *–publish 80:80* перенаправляє 80 порт з контейнера на сервер*;*
- -- name назва контейнера*;*
- *–e LOCATION=/webdav* шлях, який потрібно ввести в адресній строці для доступу до веб-сервера;
- *–d bytemark/webdav* використання Docker Image для розгортання Webdav, розташованого в мережі, а саме на DockerHub*;*

- *Docker exec* інтерактивне управління контейнером*;*
  - *Wget $FILEURL –O /var/lib/data/$DOCUMENTNAME* встановлює файли в примонтовані папки.

Запуск сервера наведено на рис. 2.7 та 2.8.

Рис. 2.7. Перший запуск веб-сервера Webdav у браузері

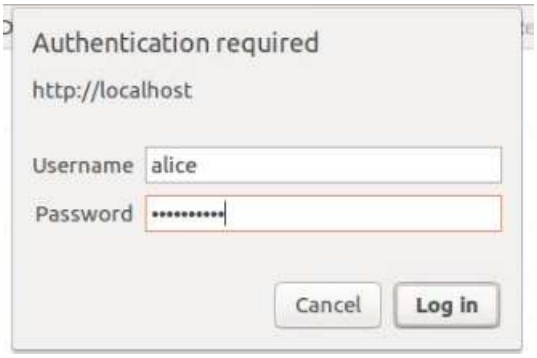

Рис. 2.8. Ввід даних для авторизації

2.2.3. Алгоритм створення бота

Кожному боту при його створенні надається унікальний маркер автентифікації. Створення бота відбувається через головного бота @BotFather через команду /newbot. Скріншот створення бота наведено на рис. 2.9. Маркер виглядає приблизно так:

123456: ABC-DEF1234ghIkl-zyx57W2v1u123ew11

Але замість цього ми будемо використовувати просто <token> для спрощення візуального сприйняття.

Усі запити до API Telegram Bot повинні подаватися через HTTPS і подаватися у такій формі: *https://api.telegram.org/bot <token> / METHOD_NAME*

Наприклад:

https://api.telegram.org/bot123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11/getMe

Телеграм підтримує методи GET та POST HTTP.

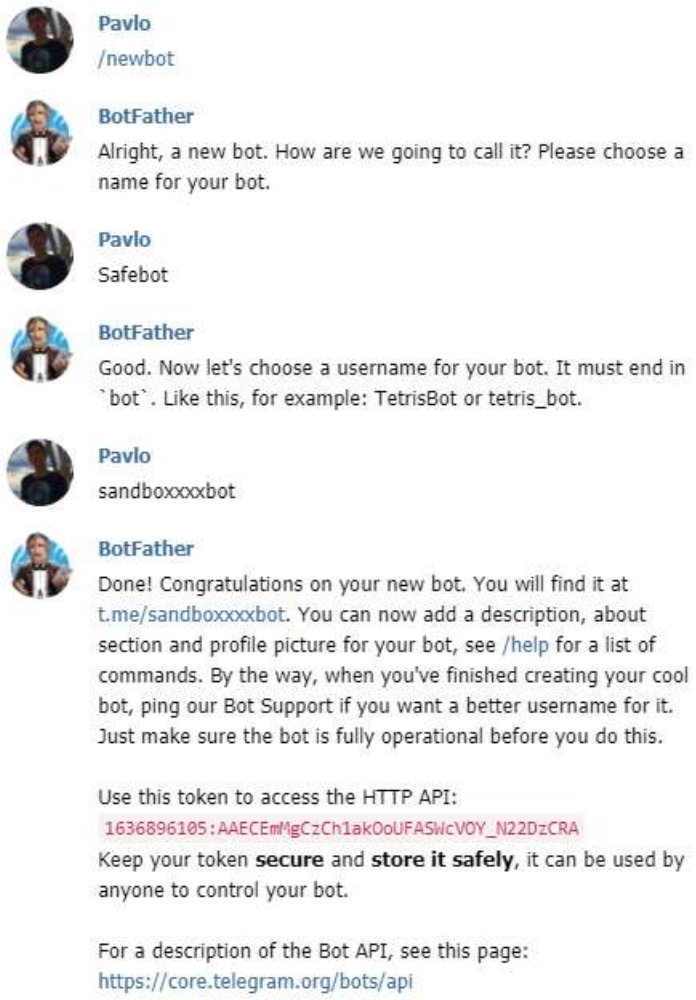

Рис. 2.9 Створення бота за допомогою бота BotFather

На даному етапі ми отримали токен, за допомогою якого ми матимемо доступ до нашого бота через API Телеграму. Наступним етапом є ідентифікація користувача. Перейшовши по посиланню, отриманому від BotFather, t.me/sandboxxxxbot потрібно перейти на створений нами бот та ввести команду /start, а також було введено тестове повідомлення для того, аби переконатись в тому, що наш бот правильно читає наші повідомлення.

Рис. 2.10 Створений бот

Тепер у нас є можливість перевірити отримане повідомлення за допомогою GET запиту:

Curl-s https://api.telegram.org/<token> /getUpdates | jq '.'

cURL – це утиліта для організації вибірки даних з веб, що надає можливість гнучкого формування запиту із завданням таких параметрів, як cookie, user_agent, referrer і будь-яких інших заголовків.

jq – надвисокорівнева мова функційного програмування з підтримкою бектрекінгу для роботи з потоками даних в форматі JSON.

```
pavlo@lwo1-lhp-f70493:~/Diplome/Main/Email$ curl -s https://api.telegram.org/bot1636896105:AAECEmMgCzCh1akOoUFASWcVOY_N22DzCRA/getUpdates | jq '.'
{
  "ok": true,
  "result": [
    {
      "update_id": 119320733,
      "message": {
        "message_id": 2,
        "from": {
          "id": 391827007,
          "is_bot": false,
          "first_name": "Pavlo",
          "last_name": "Mikush",
          "username": "ambidexterrrr",
          "language_code": "uk"
        },
        "chat": {
          "id": 391827007,
          "first_name": "Pavlo",
          "last_name": "Mikush",
          "username": "ambidexterrrr",
          "type": "private"
        },
        "date": 1611851410,
        "text": "Some_message"
      }
    }
  ]
}
```

Рис.2.11. Отримане повідомлення через запит cURL

З даного JSON ми маємо таку інформацію, як Ідентифікатор користувача, час та зміст повідомлення. Схожим способом ми можемо отримати файл (рис. 2.12).

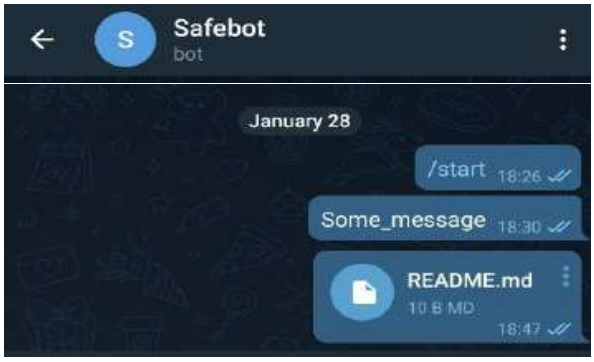

Рис. 2.12. Надсилання файла боту.

```
{
  "update_id": 119320734,
  "message": {
    "message_id": 3,
    "from": {
      "id": 391827007,
      "is_bot": false,
      "first_name": "Pavlo",
      "last_name": "Mikush",
      "username": "ambidexterrrr",
      "language_code": "en"
    },
    "chat": {
      "id": 391827007,
      "first_name": "Pavlo",
      "last_name": "Mikush",
      "username": "ambidexterrrr",
      "type": "private"
    },
    "date": 1611852426,
    "document": {
      "file_name": "README.md",
      "mime_type": "text/markdown",
      "file_id": "BQACAgIAAxkBAAMDYBLqiuTHTZk911ho02IYgK6wbz8AAssMAALXpphIC0vXcIsQd_QeBA",
      "file_unique_id": "AgADywwAAtemmEg",
      "file_size": 10
    }
  }
```


Рис. 2.13. Інформація про отриманий ботом файл

З рис. 2.13 видно інформацію, про поданий нами файл. Варто звернути увагу на те, що цього запиту не достатньо для безпосереднього доступу до файлу. Для цього нам потрібно виконати наступні запити:

Curl https://api.telegram.org/bot<*token*>

/getFile?file_id=YgK6wbz8AAssMAALXpphIC0vXcIsQd_QeBA | jq '.'

В даному запиті передаємо ідентифікатор файла (file_id) в метод getFile для отримання шляху (рис. 2.14) до нашого файла.


```
pavlo@lwo1-lhp-f70493:~/Diplome/Main/Email$ curl https://api.telegram.org/bot1636896105:AAECEmMgCzCh1akOoUFASWcVOY_N22DzCRA/getFile?file_id=BQACAgIAAxkBAAMDYBLqiuTHTZk911ho02IYgK6wbz
8AAssMAALXpphIC0vXcIsQd_QeBA | jq '.'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   189  100   189    0     0    836      0 --:--:-- --:--:-- --:--:--   832
{
  "ok": true,
  "result": {
    "file_id": "BQACAgIAAxkBAAMDYBLqiuTHTZk911ho02IYgK6wbz8AAssMAALXpphIC0vXcIsQd_QeBA",
    "file_unique_id": "AgADywwAAtemmEg",
    "file_size": 10,
    "file_path": "documents/file_0.md"
  }
}
```


Рис. 2.14. Отримання шляху до файла

На цьому етапі, маючи шлях до файла ми можемо завантажити файл за допомогою команди wget (рис. 2.15):

Wget https://api.telegram.org/file/bot<*token*> /documents/file_0.md

```
pavlo@lwo1-lhp-f70493:~/Diplome/Main/Email$ wget https://api.telegram.org/file/bot1636896105:AAECEmMgCzCh1akOoUFASWcVOY_N22DzCRA/documents/file_0.md
--2021-01-28 19:07:12--  https://api.telegram.org/file/bot1636896105:AAECEmMgCzCh1akOoUFASWcVOY_N22DzCRA/documents/file_0.md
Resolving api.telegram.org (api.telegram.org)... 149.154.167.220, 2001:67c:4e8:f004::9
Connecting to api.telegram.org (api.telegram.org)|149.154.167.220|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10 [application/octet-stream]
Saving to: 'file_0.md'

file_0.md                                   100%[===========================================================================================>]

2021-01-28 19:07:13 (1,18 MB/s) - 'file_0.md' saved [10/10]
```

Рис. 2.15. Процес скачування файлу

Для зручності автоматизовано отримувати файли від великої кількості користувачів було створено наступні змінні (рис. 2.16).

```
TELEGRAM_BOT_ID="1495647488:AAEzCQzGZCuI_0ghVqvyp0X5qyw5lEqIDJg"
TELEGRAM_API="https://api.telegram.org/bot";
TELEGRAM_GET_FILE="https://api.telegram.org/file/bot";
TELEGRAM_SEND_MSG=$TELEGRAM_API$TELEGRAM_BOT_ID"/sendMessage";
TELEGRAM_GET_MSG=$TELEGRAM_API$TELEGRAM_BOT_ID"/getUpdates";
TELEGRAM_GET_FILE_PATH=$TELEGRAM_API$TELEGRAM_BOT_ID/getFile?file_id=
TELEGRAM_CHAT_IDS=(391827007)
```

Рис. 2.16. Оголошення змінних для роботи з Телеграм ботом

Опис даних змінних спрощує нам надсилання запитів до телеграм-сервера. Тепер маємо змогу викликати запит ввівши лише одну змінну. Наприклад, для отримання останньої інформації про повідомлення боту нам достатньо виконати команду:

Curl $TELEGRAM_GET_MSG

2.3. Сканування потенційно небезпечного файлу

Сканування файлів на наявність вірусу це робота антивірусу. Ми обрали ClamAV.

Clam AntiVirus – пакунок антивірусного програмного забезпечення, що працює у багатьох операційних системах, включаючи Unix-подібні ОС, Microsoft Windows та Apple Mac OS X. Є безплатним програмним забезпеченням. Головна мета Clam AntiVirus – інтеграція з серверами електронної пошти для перевірки файлів, прикріплених до повідомлень. У пакунок входить масштабований демон clamd, керований з командного рядка сканер clamscan, а також модуль оновлення сигнатур з Інтернету freshclam [13].

Основними перевагами ClamAV є:

- управління з командного рядка;
- визначення понад 8 698 643 вірусів, черв'яків, троянів, повідомлень фішингу;
- аналіз стиснутих файлів RAR (2.0, 3.0) Zip, Gzip, Bzip2, MS OLE2, MS Cabinet, MS CHM (стиснений HTML) і MS SZDD
- та інші.

Головною перевагою цього антивірусу є можливість роботи в командному рядку, що дає змогу автоматизувати процес сканування файла та перенаправляти вивід напряму користувачу. У випадку з інтерфейсовими антивірусами, ми не можемо автоматично налаштовувати перевірку потрібного файлу.

Для перевірки файлу на наявність вірусного програмного забезпечення потрібно скористатись командою clamscan (рис. 2.17).

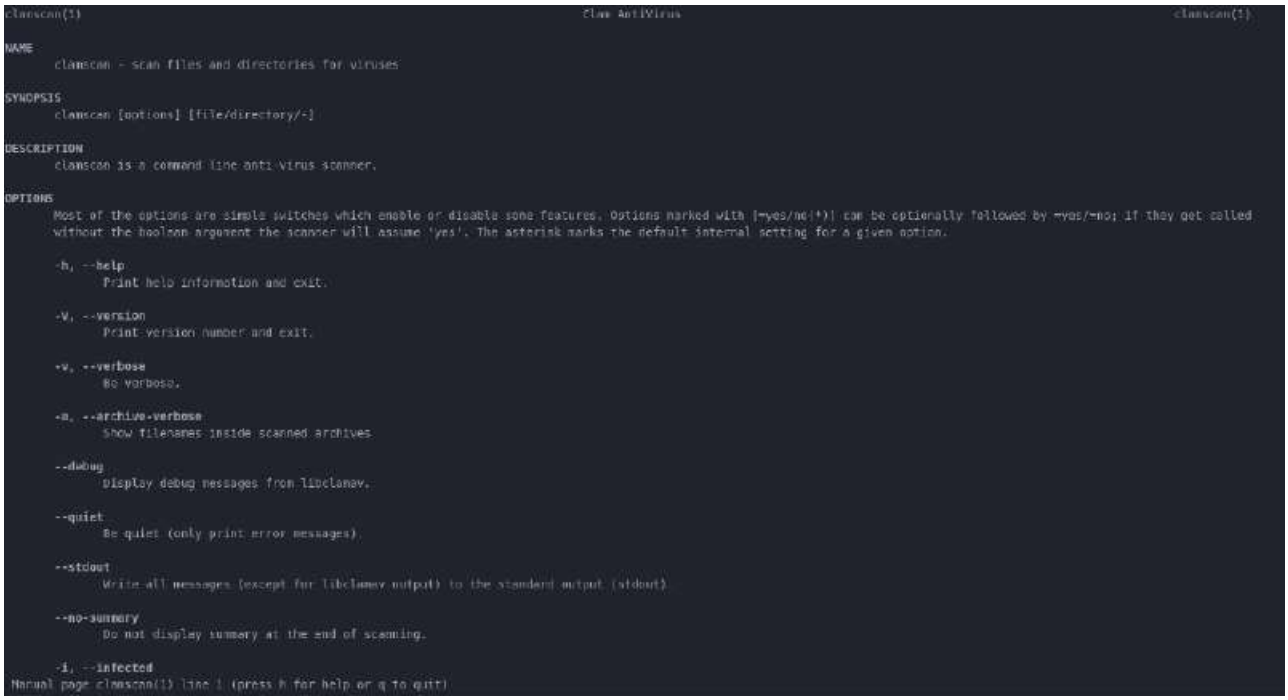

Рис. 2.17. Інформаційна сторінка антивірусу ClamAV

Антивіруси працюють наступним чином: формують вірусну базу даних, у якій вказаний приклад кодування кожним відомим вірусом та порівнюють її з отриманим файлом. У випадку збігу антивірус визначає файл, як небезпечний. З цього можемо зробити висновок, що для успішного сканування нам варто оновити вірусну базу даних. Оновлення вірусної бази даних (рис. 2.18) дозволяє нам

порівнювати файл з більшою кількістю вірусів, а також брати до уваги останні дані вірусів. Регулярне оновлення вірусної бази даних – найкращий спосіб забезпечити максимальний захист комп’ютера. Модуль оновлення гарантує, що програма завжди матиме актуальний стан. Це досягається двома шляхами: оновленням вірусної бази даних і оновленням системних компонентів.

```
Step 8/9 : RUN freshclam
 ---> Running in c2601f7e33b2
Thu Jan 21 21:14:08 2021 -> ClamAV update process started at Thu Jan 21 21:14:08 2021
Thu Jan 21 21:14:08 2021 -> daily database available for download (remote version: 26056)
Thu Jan 21 21:15:07 2021 -> Testing database: '/var/lib/clamav/tmp.f0dc09612f/clamav-4b7e23ebc892fdf8e86bb02a222d9d2f.tmp-daily.cvd' ...
Thu Jan 21 21:15:17 2021 -> Database test passed.
Thu Jan 21 21:15:17 2021 -> daily.cvd updated (version: 26056, sigs: 4199611, f-level: 63, builder: raynman)
Thu Jan 21 21:15:17 2021 -> main database available for download (remote version: 59)
Thu Jan 21 21:16:23 2021 -> Testing database: '/var/lib/clamav/tmp.f0dc09612f/clamav-87404a8c7b3e2999b248acbbaf2abc8a.tmp-main.cvd' ...
Thu Jan 21 21:16:27 2021 -> Database test passed.
Thu Jan 21 21:16:27 2021 -> main.cvd updated (version: 59, sigs: 4564902, f-level: 60, builder: sigmgr)
Thu Jan 21 21:16:27 2021 -> bytecode database available for download (remote version: 331)
Thu Jan 21 21:16:28 2021 -> Testing database: '/var/lib/clamav/tmp.f0dc09612f/clamav-0a70fc4ed4a8c4466be314d4b49b3fed.tmp-bytecode.cvd' ...
Thu Jan 21 21:16:28 2021 -> Database test passed.
Thu Jan 21 21:16:28 2021 -> bytecode.cvd updated (version: 331, sigs: 94, f-level: 63, builder: anvilleg)
Thu Jan 21 21:16:28 2021 -> ^Clamd was NOT notified: Can't connect to clamd through /var/run/clamav/clamd.ctl: No such file or directory
Removing intermediate container c2601f7e33b2
 ---> 9053bc7ab86e
```

Рис. 2.18. Оновлення вірусної бази даних

Після оновлення вірусної бази даних ми можемо перейти до етапу сканування файлу (рис. 2.19).

```
pavlo@lwo1-lhp-f70493:~/Diplome/Main$ clamscan SafeBot.sh
/home/pavlo/Diplome/Main/SafeBot.sh: OK

----------- SCAN SUMMARY -----------
Known viruses: 8698643
Engine version: 0.102.4
Scanned directories: 0
Scanned files: 1
Infected files: 0
Data scanned: 0.00 MB
Data read: 0.00 MB (ratio 1.00:1)
Time: 18.707 sec (0 m 18 s)
```

Рис. 2.19. Приклад сканування файлу

Як видно з рис. 2.19, антивірус надає стислу, лаконічну, проте чітку інформацію про сканований файл. Варто звернути увагу на перший рядок, у якому OK інформує про те, що файл є безпечним та не містить вірусів. Ефективність антивірусу підкреслює кількість відомих вірусів в вірусній базі ClamAV, а саме 8 698 643 вірусів.

Для перевірки антивірусу був використаний тестовий файл EICAR.

EICAR антивірус Test File – це комп'ютерний файл, який був розроблений Європейським інститутом комп'ютерних антивірусних досліджень (EICAR)

і антивірусних досліджень організації Computer (Caro), щоб перевірити реакцію комп'ютера антивірус (AV) програми [14]. Замість використання справжнього шкідливого програмного забезпечення, яке може спричинити реальну шкоду, цей тестовий файл дозволяє людям тестувати антивірусне програмне забезпечення без необхідності використовувати справжній комп’ютерний вірус.

Антивірусні програмісти встановлюють рядок EICAR як перевірений вірус, подібний до інших ідентифікованих підписів. Відповідний антивірусний сканер, виявляючи файл, реагуватиме більш-менш так само, як якщо виявив шкідливий вірус. Не всі антивірусні сканери сумісні і можуть не виявити файл, навіть якщо вони правильно налаштовані. Ні спосіб виявлення файлу, ні формулювання, з яким він позначений, не стандартизовані та можуть відрізнятися від способу позначення справжнього шкідливого програмного забезпечення, але повинні перешкоджати його виконанню до тих пір, поки він відповідає суворим специфікаціям, встановленим Європейським Союзом. Інститут комп'ютерних антивірусних досліджень.

Приклад тестового файлу EICAR:

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

```
pavlo@lwo1-lhp-f70493:~/Diplome/Main$ clamscan viruse.txt
/home/pavlo/Diplome/Main/viruse.txt: Eicar-Signature FOUND

----------- SCAN SUMMARY -----------
Known viruses: 8698643
Engine version: 0.102.4
Scanned directories: 0
Scanned files: 1
Infected files: 1
Data scanned: 0.00 MB
Data read: 0.00 MB (ratio 0.00:1)
Time: 17.748 sec (0 m 17 s)
```

Рис. 2.20. Приклад сканування вірусного файлу

У випадку сканування тестового файлу EICAR, ми бачимо, що в першому рядку вказаний тип вірусу Eicar-Signature FOUND. Це дає нам розуміння того, що антивірус робочий та надає нам потрібну інформацію.

Для імплементації процесу сканування файлів антивірусом ClamAV в віртуальну машину, нам потрібно дописати наступні команди в Docker-file, який виконає їх після розгортання віртуальної машини та встановлення антивірусу на неї, або за допомогою команди docker exec інтерактивно виконувати команди сканування безпосередньо з самого контейнера.

РОЗДІЛ 3. ТЕСТУВАННЯ РОЗРОБЛЕНОЇ ПЛАТФОРМИ

3.1.Симуляція процесу з безпечним файлом

Від користувача **Pavlo** надійшло повідомлення з вкладеним файлом (рис. 3.1), який потрібно перевірити на наявність вірусного програмного забезпечення.

У користуванні не обов'язково надсилати вкладений файл, тим більше, це порушує мету створення бота, а саме – можливість перевіряти файл до встановлення на комп'ютер. Тому варто переслати файл від особи, яка вам його надала, або надати посилання на цей файл.

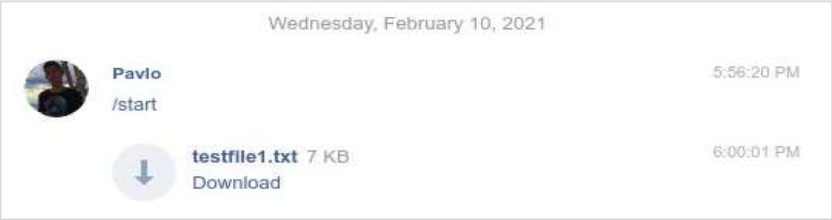

Рис. 3.1. Надсилання файла боту

Після отримання файлу відбувається зчитування ботом файлу та розгортання віртуальної машини kalilinux/kali-rolling.

Для того, щоб упевнитись в достовірності перевірки, ми розгортаємо нову віртуальну машину для кожного скачаного файла. Оскільки є ймовірність, що один з файлів буде вірусний і може змінити роботу Docker Container, що призведе до невірних даних після перевірки файлу. Проте варто зазначити, що час на розгортання одної віртуальної машини займає близько 1.5 хв, що в умовах реального часу є затратним. Тому було прийнято рішення залишати контейнер в активному стані для кожного користувача, а розгортати нову віртуальну машину лише у випадку виявлення вірусного файлу. В такому випадку віртуальну машину буде видалено та очищено кеш хост-машини, на котрій працюватиме бот. Ця дія унеможливить потрапляння файлу в систему сервера.

Після цього віртуальна машина скачує файл, який є повністю ізольований від сервера (рис. 3.2). Максимальний розмір скачаного ботом файла 20MB (у своїй документації Telegram зазначив, що працює над питанням збільшити обсяг

скачуваних файлів).

```
Step 7/9 : RUN wget -O 1.docx ${qqq}
 ---> Running in 455c7eb960ab
--2021-01-21 21:14:06--  https://api.telegram.org/file/bot1495647488:AAEzCQzGZCuI_0ghVqvyp0X5qyw5lEqIDJg/documents/file_13.txt
Resolving api.telegram.org (api.telegram.org)... 149.154.167.220, 2001:67c:4e8:f004::9
Connecting to api.telegram.org (api.telegram.org)|149.154.167.220|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 69 [application/octet-stream]
Saving to: '1.docx'

     0K                                                      100% 57.0M=0s

2021-01-21 21:14:07 (57.0 MB/s) - '1.docx' saved [69/69]

Removing intermediate container 455c7eb960ab
 ---> c66bbde59458
```

Рис. 3.2. Встановлення файлу в віртуальну машину

Текст процесу скачування файлу, зазначено на скріншоті вище червоного кольору через специфіку кодування середовища для програмування, а саме Atom. У даному випадку це не є помилкою, про що нам каже повідомлення **OK**. Отримані файли зберігаються у Віртуальній машині під одною назвою - “1.docx”. Цей процес є фоновим, результат наведено на рис. 3.3.

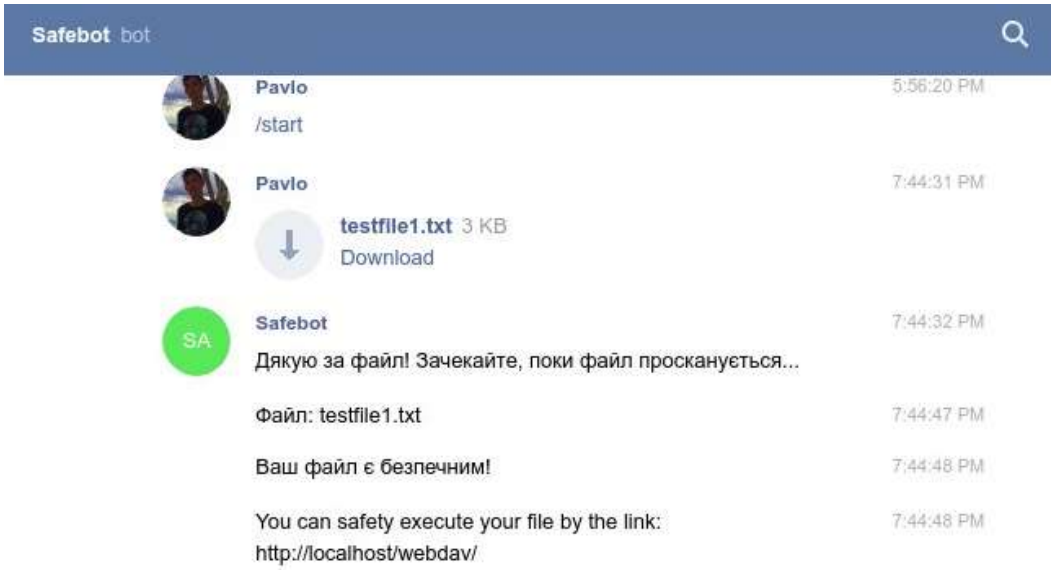

Рис. 3.3. Повідомлення бота з результатом сканування

Після переходу по запропонованій адресі, побачимо веб-сторінку з списком файлів, які ми надіслали боту (рис. 3.4). Варто зазначити, що “Parent Directory” це можливість переходу в попередню директорію. В даному випадку ми не можемо зробити цю дію, так як користувачеві наданий доступ лише до директорії з його файлами. Проте ця функція буде корисною, у випадку надсилання архіву з папками та файлами.

Рис. 3.4 Веб-сторінка з надісланими боту файлами

При натисканні на наш файл, а саме “testfile1.txt” бачимо вміст нашого файлу, не скачуючи його (рис. 3.5).

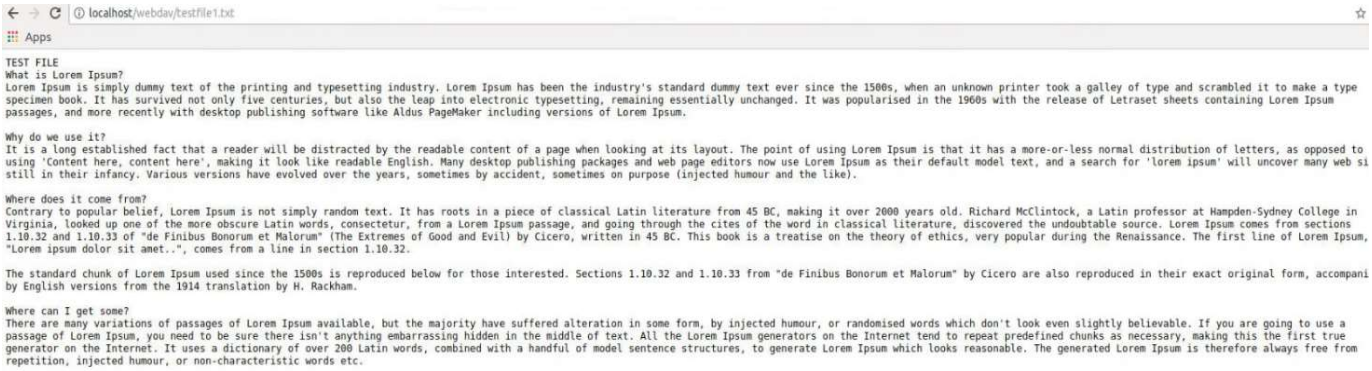

Рис. 3.5. Веб-сторінка з вмістом файла

У Safebot можна файли переадресовувати та змінювати фон. На 3.6 показано, тестування пересланого посилання на зображення з розширенням .gif.

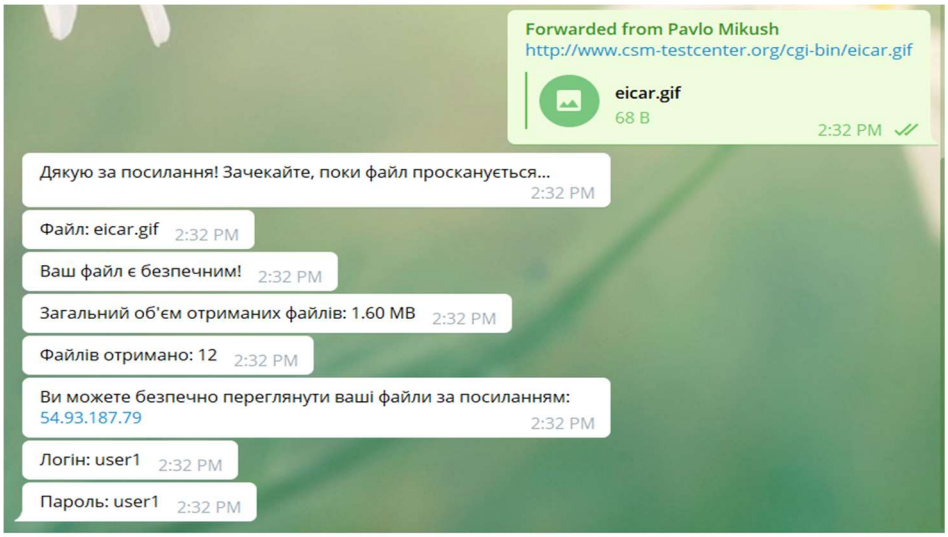

Рис. 3.6. Тестування файла *.gif

3.2. Симуляція процесу з небезпечним файлом

Цей етап є аналогічним попередньому, різниця полягає лише в тому, що файл заражений вірусом (рис. 3.7).

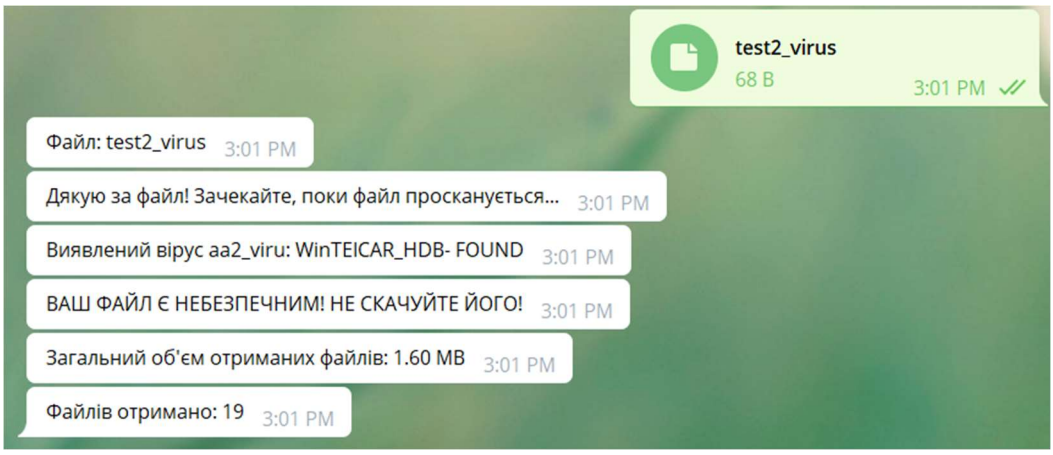

Рис. 3.7. Тестування боту вірусного файлу

Safebot надає інформацію про виявлений тип вірусу, застереження щодо скачування файла та посилання на безпечний перегляд файла. За допомогою секундоміра визначено приблизний час отримання відповіді – 16 секунд.

Якщо цей файл завантажити чи відкрити, пристрій з якого це здійнюється заразиться вірусом. На рис. 3.8 відображається вмість зараженого файла, який відкритий через безпечне середовище, яке ми розробили у цій роботі.

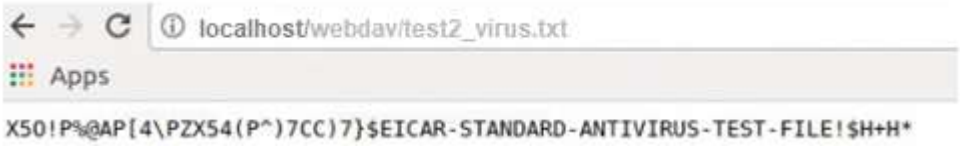

Рис. 3.8. Безпечний перегляд вірусного файлу

Safebot може працювати з файлами різних форматів. Наведемо приклад тестування архіву (рис. 3.9), оскільки часто, з архівом завантажується та встановлюється на комп’ютер вірус.

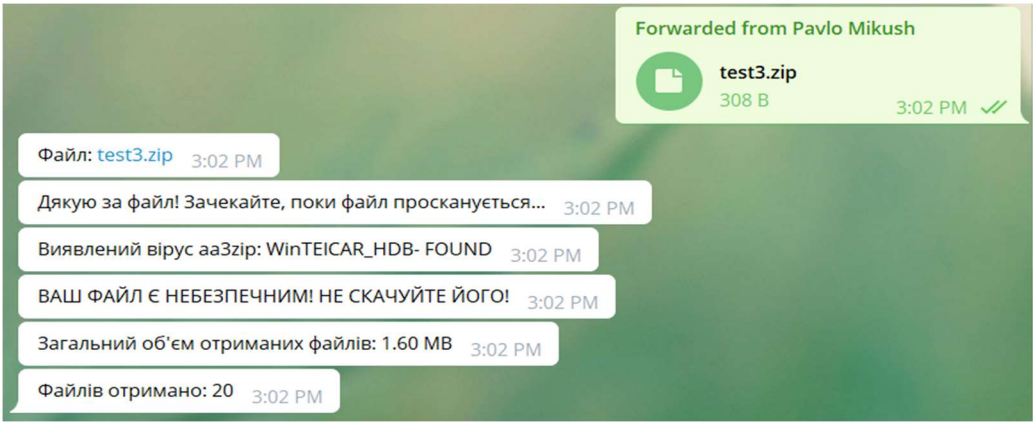

Рис. 3.9. Тестування вірусного архіву

ВИСНОВКИ

У процесі виконання наукової роботи було розроблено автоматизовану платформу для перевірки файлів на наявність шкідливого програмного забезпечення, які передаються у месенджері Telegram

На даний час розробкою користуються викладачі кафедри та студенти для перевірки підозрілих файлів. Платформа працює цілодобово. Працює як Telegram бот, який відслідковує повідомлення, виявляє файли та оперативно реагує на них. При отриманні файлу створює дві віртуальні машини. Одна з встановленим антивірусом, сканує отримані файли. Інша з встановленим веб-сервером, транслює файли з віртуальної машини у веб-браузер, що дає змогу користувачеві переглядати файли, які не мають прямого доступу. Віртуальні машини створюються для кожного користувача окремо, залишають кеш і відновлюють збереженні дані при наступному зверненні до них.

Ця розробка є унікальним методом перевірки файлів надісланих через Telegram, оскільки дає змогу перевірити файл перед скачуванням, не ризикуючи встановити вірусне програмне забезпечення на свій пристрій. Можливість відображати файл в браузері дає можливість зберегти фізичну пам’ять комп’ютера, а також оперативну пам’ять, якщо локально розгортати віртуальну машину для власного користування.

Для користування дана програма працює централізовано, тому не вимагає технічних характеристик з боку користувача, окрім доступу до мережі Інтернет.

Встановлена програма на сервері в робочому стані витрачає:

- 14 Гб вітуальної пам’яті;
- 3.7 Гб вбудованої пам’яті (залежить від навантаження);
- 3 Гб оперативної пам’яті (залежить від навантаження).

Також варто врахувати потенційну кількість користувачів. Адже чим більше користувачів, тим більше ресурсів програма буде витрачати ресурсів.

Програма розпізнає від 8 700 000 вірусів. Час на відповідь 16 секунд.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рейтинг найбільш популярних месенджерів за рівнем надійності. [Електронний ресурс]. Доступно з: https://falcongaze.com/ua/pressroom/publications/research/2.html
2. Number of monthly active Telegram users worldwide from March 2014 to April 2020. [Електронний ресурс]. – Доступно з: https://www.statista.com/statistics/234038/telegram-messenger-mau-users/
3. What is a Telegram virus. [Електронний ресурс]. – Доступно з: https://www.wipersoft.com/telegram-virus/
4. Що таке прихований майнер криптовалюти? [Електронний ресурс]. – Доступно з: https://spravka.co.ua/posts/virus/uk/virus-virus-majner-ak-viaviti-i-vidaliti-povne-kerivnictvo/
5. Вірусна активність в Україні: троянські програми атакують частіше. [Електронний ресурс]. – Доступно з: https://zillya.ua/virusna-aktivnist-v-ukrani-troyanski-programi-atakuyut-chastishe-0?page=1
6. Пісочниця (комп'ютерна безпека). [Електронний ресурс]. – Доступно з: https://uk.wikipedia.org/wiki/Пісочниця
7. **Мікуш. П.Ю. Пісочниці комп'ютерних систем як механізм захисту від вірусів / П.Ю. Мікуш, М.М. Шабатура // збірник тез доповідей IV Всеукраїнської науково-практичної конференції молодих учених, студентів і курсантів «Інформаційна безпека та інформаційні технології», 27.11. 2020. – Львів, ЛДУ БЖД, 2020, с. 45-47.**
8. Announcing Docker Hub and Offical Repositories. [Електронний ресурс]. – Доступно з: https://www.docker.com/blog/announcing-docker-hub-and-official-repositories/
9. Docker. [Електронний ресурс]. – Доступно з: https://habr.com/ru/post/353238/
10. Docker reference. [Електронний ресурс]. – Доступно з: https://docs.docker.com/engine/reference/builder/
11. Kali Linux. [Електронний ресурс]. – Доступно з: https://www.kali.org/
12. [Електронний ресурс]. – Доступно з: https://xakep.ru/2014/09/09/webdav/]
13. Офіційний сайт Clamav. [Електронний ресурс]. – Доступно з: https://www.clamav.net/

14. EICAR. [Електронний ресурс]. – Доступно: https://www.eicar.org/?page_id=3950

ДОДАТКИ

Лістинг програми

```
VERSION="1.0"
TELEGRAM_BOT_ID="1636896105:AAECEmMgCzCh1akOoUFASWcVOY_N22DzCRA"
TELEGRAM_API="https://api.telegram.org/bot";
TELEGRAM_GET_FILE="https://api.telegram.org/file/bot";
TELEGRAM_SEND_MSG=$TELEGRAM_API$TELEGRAM_BOT_ID"/sendMessage";
TELEGRAM_GET_MSG=$TELEGRAM_API$TELEGRAM_BOT_ID"/getUpdates";
ABOUTME=`curl -s "$TELEGRAM_API$TELEGRAM_BOT_ID/getMe"`
TELEGRAM_GET_FILE_PATH=$TELEGRAM_API$TELEGRAM_BOT_ID/getFile?file_id=
TELEGRAM_CHAT_IDS=(391827007)
CHECKNEWMSG=3;
FIRSTTIME=0;
BOTPATH="`dirname \"$0\"`";


sendmessage(){
    msg="$@"
    for i in "${TELEGRAM_CHAT_IDS[@]}"; do
        curl -s $TELEGRAM_SEND_MSG -d chat_id=$i -d text="$msg" > /dev/null 2>&1
    Done
}


if [ -e "${BOTPATH}/${BOTID}.lastmsg" ]; then
          FIRSTTIME=0;
else
    touch ${BOTPATH}/${BOTID}.lastmsg;
         FIRSTTIME=1;
fi


sudo docker build -t clamav .  > /dev/null 2>&1


while true; do


    LASTMSGID=$(cat "${BOTID}.lastmsg");
    MSGID=0;
         TEXT=0;
         FIRSTNAME="";
         LASTNAME="";
    MSGOUTPUT=$(curl -s $TELEGRAM_GET_MSG);
```

```
    echo "${MSGOUTPUT}" | while read -r line ; do


        if [[ "$line" =~ \"chat\"\:\{\"id\"\:([\-0-9]+)\, ]]; then
                        CHATID=${BASH_REMATCH[1]};
              fi


              if [[ "$line" =~ \"message\_id\"\:([0-9]+)\, ]]; then
                        MSGID=${BASH_REMATCH[1]};
              fi


              if [[ "$line" =~ \"text\"\:\"(.+)\"\}\} ]]; then
             TEXT=$(echo $MSGOUTPUT | jq ".result[].message | select(.message_id >
$LASTMSGID) .text" | tr -d '"')
        fi


              if [[ "$line" =~ \"username\"\:\"([^\"]+)\" ]]; then
                        USERNAME=${BASH_REMATCH[1]};
              fi


              if [[ "$line" =~ \"first_name\"\:\"([^\"]+)\" ]]; then
                     FIRSTNAME=${BASH_REMATCH[1]};
              fi


              if [[ "$line" =~ \"last_name\"\:\"([^\"]+)\" ]]; then
                     LASTNAME=${BASH_REMATCH[1]};
              fi


              if [[ "$line" =~ \"from\"\:\{\"id\"\:([0-9\-]+), ]]; then
                     FROMID=" ${BASH_REMATCH[1]}";
              fi


              if [[ $MSGID -ne 0 && $CHATID -ne 0 ]]; then
                  LASTMSGID=$(cat "${BOTID}.lastmsg");
            if [[ $MSGID -gt $LASTMSGID ]]; then
              DOCUMENTNAME=$(echo $MSGOUTPUT | jq ".result[].message |
select(.message_id > $LASTMSGID) .document.file_name" | tr -d '"')
              FILEID=$(echo $MSGOUTPUT | jq ".result[].message  | select(.message_id >
$LASTMSGID) .document.file_id" | tr -d '"')

                                echo $MSGID > "${BOTID}.lastmsg";
```

```
                if [ "${FILEID}" == "null" ]; then
                 echo "[chat ${CHATID}, from ${FROMID}] <${USERNAME} - ${FIRSTNAME}
${LASTNAME}> ${TEXT}";
                  sendmessage Дякую за повідомлення!
                  else
                  echo "[chat ${CHATID}, from ${FROMID}] <${USERNAME} - ${FIRSTNAME}
${LASTNAME}> ${DOCUMENTNAME}";
                  sendmessage Дякую за файл! Зачекайте, поки файл просканується...
                  FILEPATH=$(curl -s $TELEGRAM_GET_FILE_PATH$FILEID | jq
'.result.file_path' | tr -d '"')
                  FILEURL=$TELEGRAM_GET_FILE${TELEGRAM_BOT_ID}/$FILEPATH
                  [[ $(docker ps -f "name=cont$USERNAME" --format '{{.Names}}') ==
cont$USERNAME ]] || docker run -d -t --name cont$USERNAME clamav:latest > /dev/null
2>&1
                  docker exec -e FILEURL=$FILEURL cont$USERNAME wget -O 1.docx $FILEURL
> /dev/null 2>&1
                  docker exec cont$USERNAME clamscan 1.docx > log.txt


                  #[[ $(docker ps -f "name=webdav$USERNAME" --format '{{.Names}}') ==
webdav$USERNAME ]] || docker run --restart always -v /srv/dav:/var/lib/dav -e
AUTH_TYPE=Digest -e USERNAME=alice -e PASSWORD=secret1234  --publish 80:80 --name
webdav$USERNAME -e LOCATION=/webdav -d bytemark/webdav
                  #docker exec webdav$USERNAME wget $FILEURL -P /var/lib/dav/data/ #-O
$DOCUMENTNAME #> /dev/null 2>&1

                  output_safe=$(cat log.txt | grep 'Infected files: 0')
                  output_unsafe=$(cat log.txt | grep 'Infected files: 1')
                  type_of_virus=$(cat log.txt | grep  '/test/1.docx:' | sed
"s/['/test/1.docx']//g")
                  sendmessage Файл: $DOCUMENTNAME
                  if  [ -z "$output_unsafe" ]; then
                      sendmessage Ваш файл є безпечним!
                    else
                      sendmessage Виявлений вірус $type_of_virus
                      sendmessage ВАШ ФАЙЛ Є НЕБЕЗПЕЧНИМ! НЕ СКАЧУЙТЕ ЙОГО!
                      docker rm $(docker stop cont$USERNAME) > /dev/null 2>&1
                  fi
                  sendmessage You can safety execute your file by the link:
http://localhost/webdav/
                  cat log.txt >> all_logs.txt
                  rm log.txt
                fi


              fi

                    fi
```

```
        done

        FIRSTTIME=0;

Done

exit 0
```