Pavlo MIKUSH[1]

Opiekun naukowy: Mariia SHABATURA[2]

DOI: https://doi.org/10.53052/9788366249868.15

ZAUTOMATYZOWANA PLATFORMA DO SPRAWDZANIA PODEJRZEWANYCH PLIKÓW

Streszczenie: Artykuł przedstawia opracowaną zautomatyzowaną platformę, która działa jak „piaskownica” i zdalnie sprawdza podejrzane pliki przesyłane w komunikatorze Telegram. Zostanie wdrożony serwer sieciowy, który umożliwi przeglądanie plików w izolacji za pomocą przeglądarki internetowej bez bezpośredniego dostępu do nich. Rozwój ten jest w pełni zautomatyzowany, nie wymaga dodatkowych działań i oszczędza własne zasoby urządzeń użytkowników.

Słowa kluczowe: piaskownica, bot Telegram, ochrona przed złośliwym oprogramowaniem

AUTOMATED PLATFORM FOR CHECKING SUSPICIOUS FILES

Summary: The paper shows the developed automated platform, which acts as a "sandbox" and remotely checks suspicious files transmitted in the messenger Telegram. A web server will be deployed that will allow viewing files in isolation through a web browser without direct access to them. This development is fully automated, does not require additional actions, and saves users' own devices resources.

Keywords: sandbox, Telegram bot, malware protection

1. Introduction

Every year the popularity of messengers increases, they become an integral attribute of modern life, the number of regular users grows. At the same time, this new form of communication is becoming one of the current research areas.

In our time, as a result of the Covid-19 pandemic, humanity has faced one of the most significant reforms - digitalization. This process has existed for a long

[1] Student of Department of Information Technology Security, Lviv Polytechnic National University, pavlo.mikush.mkbasz.2021@lpnu.ua

[2] PhD, Associate Professor of Department of Information Technology Security, Lviv Polytechnic National University, mariia.m.mandrona@lpnu.ua

time, but it has accelerated many times due to the pandemic, instantly transferring most of the business, education, and even entertainment online. Of course, it provides many new opportunities and is an undisputed step forward. However, with the sharp increase in online activity, the activity of attackers increases, who use all possible methods to steal other people's data or destroy the efficiency of systems. The simplest method is to spread the virus software in the form of regular files. Moreover, the distribution of files is through file sharers and messengers.

This **work aims** to create an automated platform that will act as a "sandbox" and remotely check the files transmitted in the Telegram messenger for malware.

2. Telegram

Telegram is software for smartphones, tablets, and computers that allows exchanging text messages, graphics, and video files and calling other program users for free. In general, Telegram is more reliable than others, according to global cybersecurity companies such as San, Isaca, Enisa and others. In particular, the analytical center Falcongaze, a company developer of software solutions in the field of data leakage prevention through various channels (e-mail, social networks, messengers, USB, etc.), ranked the most popular messengers depending on their level of security (Fig. 1) [1]. In the first place, Telegram.

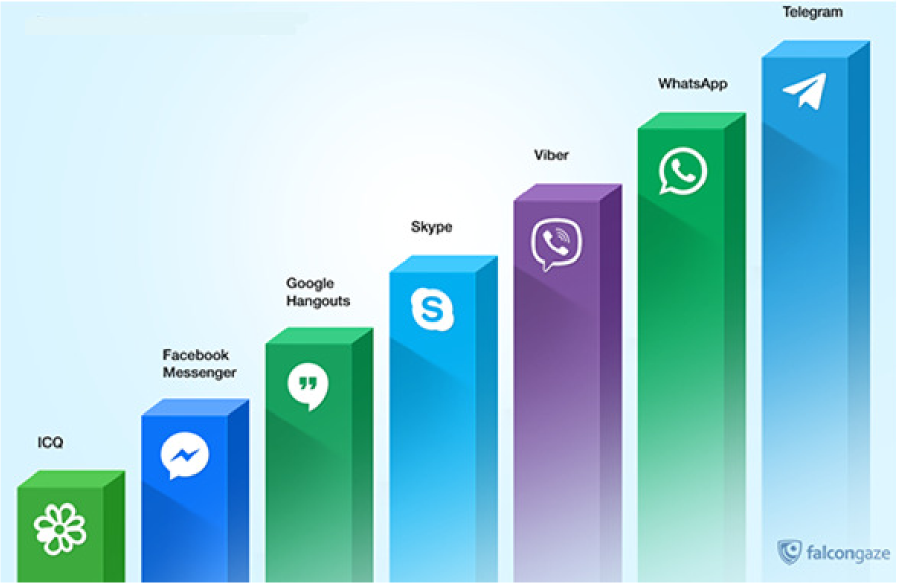


Figure 1. Rating of the most secure popular messengers

According to the global portal statista.com, this platform is used by a significant number of users. In fig. Fig. 2 shows the growth chart of the number of users from 2014 to 2020 [2].

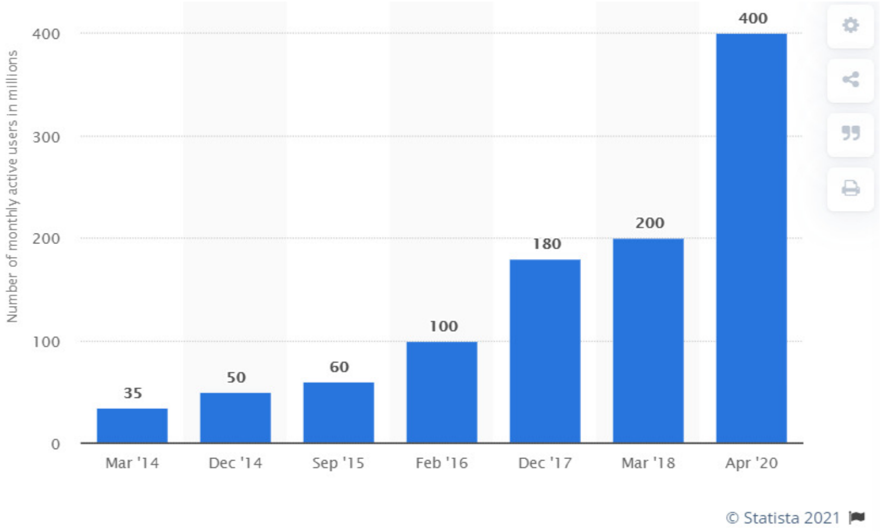


Figure 2. Active users Telegram Statistics

Telegram is known as a private, and most importantly, secure messaging service because it automatically encrypts all messages and offers features such as self-destruction of the account if users are not delayed for a certain period.

However, cybercriminals are always finding ways to use, even seemingly the most secure Telegram platform, to harm users. We are talking about sending files with viruses.

A virus in Telegram is a possible threat that can be spread through a messaging program or even presented as the program itself. This term is used to describe a new cyber threat that is being created for many purposes. Hackers distribute malicious software using completely legitimate cloud-based messaging software [3].

The purpose of this infection is to spread the cryptocurrency miner [4] or to penetrate the system with spyware and other potentially unwanted programs and gain remote control of the device. Unfortunately, it is difficult to identify the virus because it hides its presence in the system. It is clear that the Telegram resource itself has nothing to do with these scams and malware; it is just a convenient platform used to spread infections. As soon as malicious software enters the system, it settles in the deepest places on the smartphone or computer and starts running in the background.

Based on the analysis of literature sources, one of the most common malicious programs is the distribution of cryptocurrency miners through Telegram [3-4]. These infections are not particularly serious, as they do not usually harm a computer. They mainly focus on using computer resources to extract cryptocurrencies such as Monero, ZCash, etc. In addition to using computer resources, this will cause long-term damage to the system. However, mining causes short-term problems, especially with the computer. The system will become sluggish, programs will constantly load and crash, and strange processes with high CPU usage will appear in Task Manager. These are all signs of cryptocurrency malware. Fortunately, it is so noticeable that the user will know it right away if the computer gets infected.

Other dangerous malware users can get on platforms like Telegram are Trojans that can spy on and steal private information. Trojans can also act as backdoors and allow other malicious software to install. Most antivirus programs can detect and remove most Telegram viruses, so as long as users have a security program installed, it should protect the computer [3].

3. Platform development process

We decided to develop a platform that allows checking the submitted files for malware and call it "Safebot". To implement an isolated mechanism for safe execution of programs called "Sandboxes" or "Sandbox" [5, 6], we need to deploy a virtual machine, install an antivirus on it and configure the ability to download files. The popular Telegram messenger was chosen to upload files to the virtual machine, which is a quite popular today (see Fig. 1). Since Telegram is an anonymous file sharer, there is a high risk of downloading an infected file sent by an attacker. The ability of Telegram to interact with the "Sandbox" allows you to check the received files by sending them to the bot without opening them [5]. An intermediary between the user and the "Sandbox" will be a platform, which will redirect files directly to the virtual machine and provide the user with the final information about the scan results.

Stages of our development:
Step 1. The beginning of the bot will be considered to receive a file in a message from the user. It should be noted that the bot responds to send messages with a file similar to the attached message directly from the user. This nuance is very important, because it allows sending messages from the bot attacker without downloading the file or following the link. Then the bot will receive this message, determine the type of message (text, media, audio, file, etc.). This step is necessary because the bot needs to know what type of data it will get to work. The next step will be to form a direct link to the file. This step is required to download the file sent by the user.
Step 2. Having a direct link to file, the bot performs the deployment of the Docker Container with the installation of the necessary software, which will be used to check the file for viruses.
Step 3. After scanning, we write a log file (file with data on the execution of previous commands), from which we form a response to the client about the status of the file. It takes about 16 seconds. The answer will be sent to the user immediately after the scan.
Step 4. After scanning the file, we proceed to the deployment of the WebDav web server [7]. We install user files on it, and we do it too in the virtual machine to isolate a file from a working network of the server. Then we form a link and provide it to the user for interactive access to the file through a web browser.
Step 5. Send a response with the scan result. The described algorithm is fully automated, which means that the user will not see the process of deploying the virtual machine and scanning the file, but only the bot's response.
Summarizing the above, Fig. 3 shows the architecture of the working directory of the project.

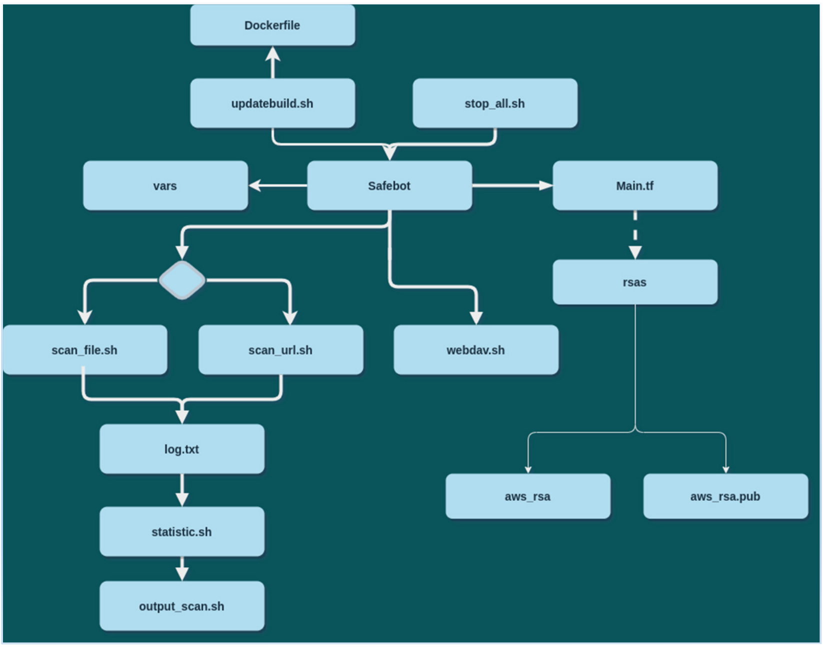


Figure 3. Working directory architecture

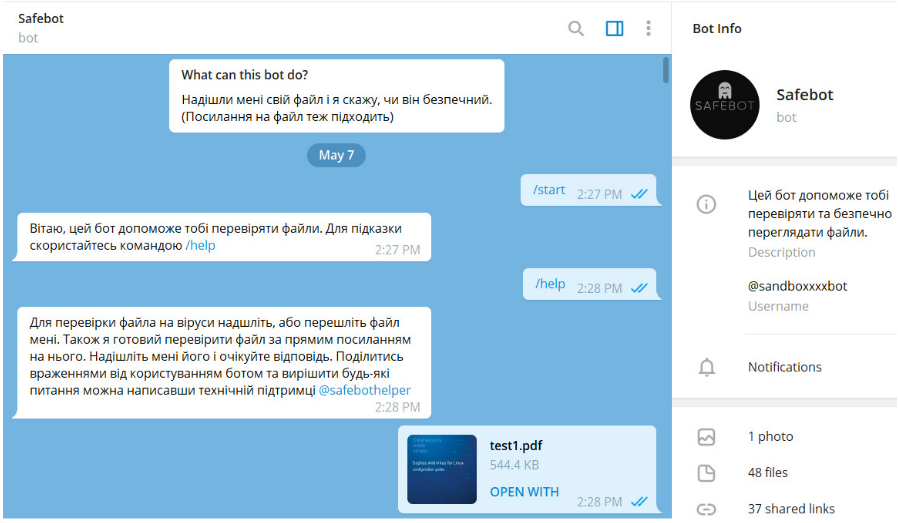
Safebot.sh – the main file that controls all processes launches the program, generates a direct link to the file received by the bot from the user, manages all the logic of our bot: collects all the necessary information about the file from which we received a message; sends messages to the user with information about the status of the file; writes user information to the database. For the bot to work, we only need to run this file, and it, in turn, will run all the others if necessary.

Main.tf – Terraform file. Written in YUML, it builds the infrastructure on the side of Amazon Web Service. Deploys a server with a direct connection to our server and describes security and network routing rules.

Scan_file.sh, Scan_url.sh – Depending on the type of file representation in the message (attachment, or file link), a file is launched. The logic of scanning files is one create a container with antivirus installed and scan. The only difference is that when receiving an attachment, the bot needs to generate a direct link to access the file from the Telegram server. Generate a Log.txt file; run Statistic.sh and output_scan.sh

4. Testing

Simulate the process with a secure file. The user unknown received a message with an attached file from unknown user, which must be checked for virus software. Below is the process of working with the bot. To start, use the /start command, which will run the bot for the user also have the opportunity to use the help command. This command gives us additional information about what exactly we need to do to check our files.



What can this bot do?

Надішли мені свій файл і я скажу, чи він безпечний. (Посилання на файл теж підходить)

/start 2:27 PM

Вітаю, цей бот допоможе тобі перевіряти файли. Для підказки скористайтесь командою /help 2:27 PM

/help 2:28 PM

Для перевірки файла на віруси надшліть, або перешліть файл мені. Також я готовий перевірити файл за прямим посиланням на нього. Надішліть мені його і очікуйте відповідь. Поділитись враженнями від користуванням ботом та вирішити будь-які питання можна написавши технічній підтримці @safebothelper 2:28 PM

test1.pdf 2:28 PM

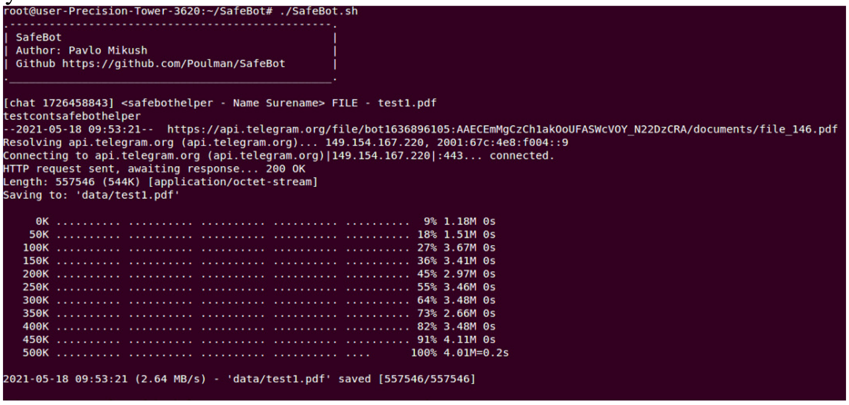
Цей бот допоможе тобі перевіряти та безпечно переглядати файли.
Description

@sandboxxxxbot
Username

Figure 4. Sending a file to Safebot

After receiving the file, Safebot reads the file and deploys the virtual machine Kali Linux / kali-rolling. To ensure that the validation is valid, we deploy a new virtual machine for each downloaded file. Because it is likely that one of the files will be viral and may change the Docker Container, which will lead to incorrect data after checking the file. However, it should be noted that the deployment time of one virtual machine takes about 1.5 minutes, which in real-time is costly. Therefore, it decided to leave the container active for each user and deploy a new virtual machine only if a virus file is detected. In this case, the virtual machine will delete, and the cache of the host machine on which the bot will run will clear. It will prevent the file from entering the server system. After that, the virtual machine downloads file that is completely isolated from the server (Fig. 5). The maximum size of the file downloaded by the bot is 20MB (Telegram documentation stated that it is working on the issue of increasing the number of downloaded files).

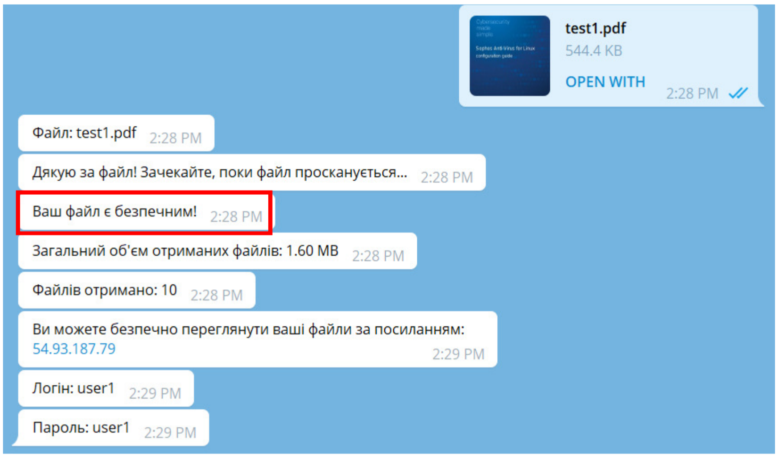
Fig. 5 shows the operation of the bot on the server. This process is server-only, so only the server administrator can see it.



```
root@user-Precision-Tower-3620:~/SafeBot# ./SafeBot.sh
.-----------------------------------------------.
| SafeBot                                       |
| Author: Pavlo Mikush                          |
| Github https://github.com/Poulman/SafeBot     |
._______________________________________________.

[chat 1726458843] <safebothelper - Name Surename> FILE - test1.pdf
testcontsafebothelper
--2021-05-18 09:53:21--  https://api.telegram.org/file/bot1636896105:AAECEmMgCzCh1akOoUFASWcVOY_N22DzCRA/documents/file_146.pdf
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: 557546 (544K) [application/octet-stream]
Saving to: 'data/test1.pdf'

     0K .......... .......... .......... .......... ..........  9% 1.18M 0s
    50K .......... .......... .......... .......... .......... 18% 1.51M 0s
   100K .......... .......... .......... .......... .......... 27% 3.67M 0s
   150K .......... .......... .......... .......... .......... 36% 3.41M 0s
   200K .......... .......... .......... .......... .......... 45% 2.97M 0s
   250K .......... .......... .......... .......... .......... 55% 3.46M 0s
   300K .......... .......... .......... .......... .......... 64% 3.48M 0s
   350K .......... .......... .......... .......... .......... 73% 2.66M 0s
   400K .......... .......... .......... .......... .......... 82% 3.48M 0s
   450K .......... .......... .......... .......... .......... 91% 4.11M 0s
   500K .......... .......... .......... .......... ....      100% 4.01M=0.2s

2021-05-18 09:53:21 (2.64 MB/s) - 'data/test1.pdf' saved [557546/557546]
```

Figure 5. Display of Safebot operation from the server side

Fig. 6 shows Safebot response. As we can see, the file is secure and can be downloaded and opened.



test1.pdf
544.4 KB
OPEN WITH 2:28 PM

Файл: test1.pdf 2:28 PM

Дякую за файл! Зачекайте, поки файл проскануеться... 2:28 PM

Ваш файл є безпечним! 2:28 PM

Загальний об'єм отриманих файлів: 1.60 MB 2:28 PM

Файлів отримано: 10 2:28 PM

Ви можете безпечно переглянути ваші файли за посиланням:
54.93.187.79 2:29 PM

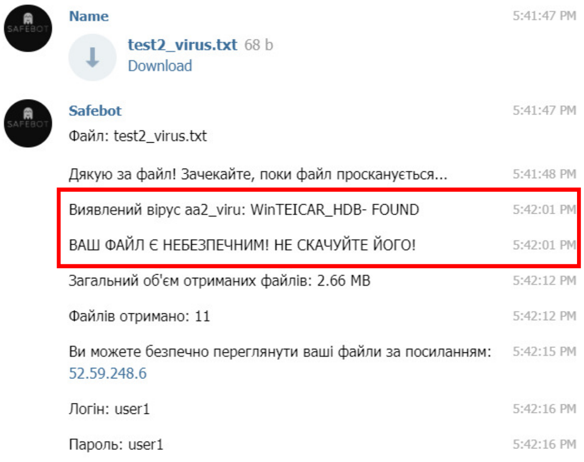
Логін: user1 2:29 PM

Пароль: user1 2:29 PM

Figure 6. Bot message with the result of the scan

The advantage of Safebot is the ability to work seamlessly with any type of data and in any format.

Simulation with a dangerous file. This stage is similar to the previous one. The only difference is that the file is infected with a virus (Fig. 7).



Name 5:41:47 PM
test2_virus.txt 68 b
Download

Safebot 5:41:47 PM
Файл: test2_virus.txt

Дякую за файл! Зачекайте, поки файл проскануеться... 5:41:48 PM

Виявлений вірус aa2_viru: WinTEICAR_HDB- FOUND 5:42:01 PM

ВАШ ФАЙЛ Є НЕБЕЗПЕЧНИМ! НЕ СКАЧУЙТЕ ЙОГО! 5:42:01 PM

Загальний об'єм отриманих файлів: 2.66 MB 5:42:12 PM

Файлів отримано: 11 5:42:12 PM

Ви можете безпечно переглянути ваші файли за посиланням: 5:42:15 PM
52.59.248.6

Логін: user1 5:42:16 PM

Пароль: user1 5:42:16 PM

Figure 7. Safebot work with a virus file

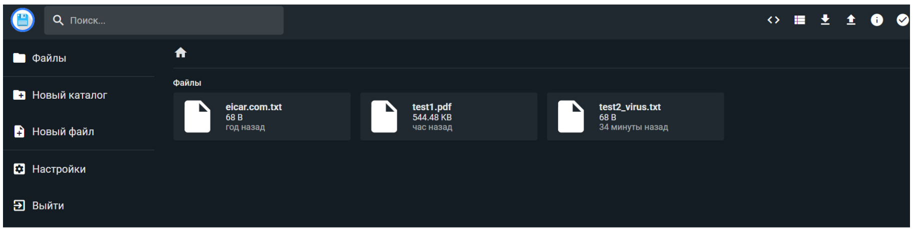
It should be noted how much time Safebot spends performing the scan. The web version of telegram was used for verification accuracy, allowing to make sure that the bot will work in any version of the messenger. So, as we see in Fig. 7, the file was sent at 5:41:47. The file's response was received and taken into operation was sent at 5:41:47, which indicates that the bot immediately recognizes the received messages.

The result of the program execution was sent at 5:42:01. From this, we can get the approximate execution time of the program, namely 14 seconds. Statistics were sent at 5:42:12, 11 seconds later. This delay is due to the use of a database and several data checks.

Summarize:

- scanning the file takes 14 seconds;
- calculation of statistics takes 11 seconds;
- uploading a file to Amazon Web Service and sending a link with access data to the user takes 3 seconds;

In addition, from the provided message, we see the detected type of virus, warnings about downloading the file, and a link to a secure browser (Fig. 8).



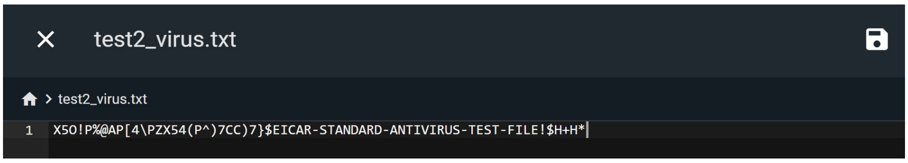
eicar.com.txt
68 B
год назад

test1.pdf
544.48 KB
час назад

test2_virus.txt
68 B
34 минуты назад

Figure 8. List of received files

If the file is downloaded or opened, the device will be infected with a virus. In fig. 9 displays an infected file's contents that are opened through a secure environment that we have developed in this paper.

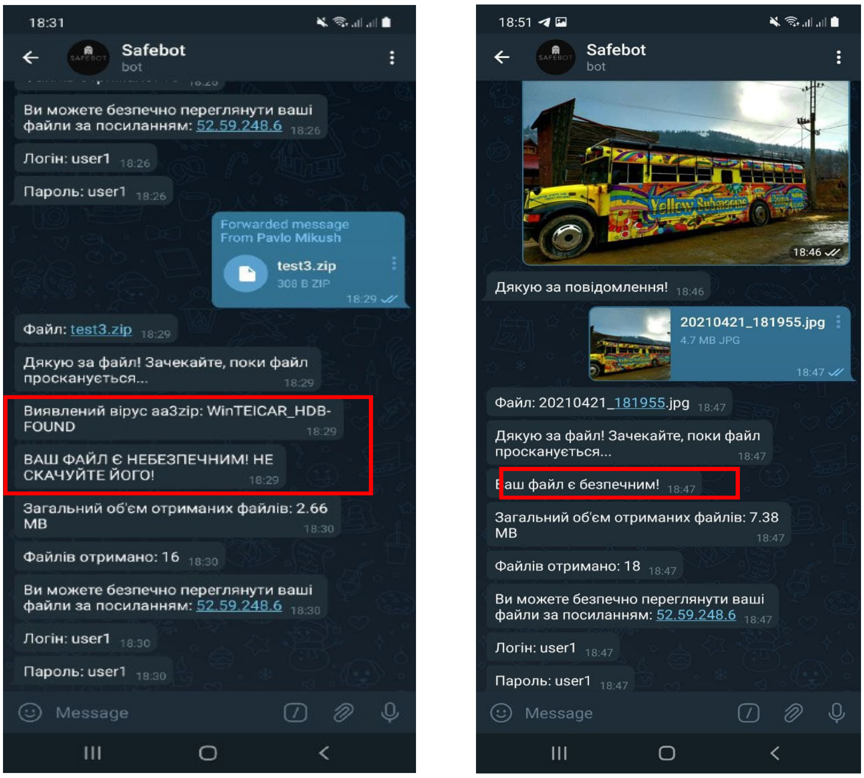


```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

Figure 9. Safe browsing of a virus file

The advantage of Safebot is the ability to work seamlessly with any type of data and in any format.

Simulation from mobile version. Mobile phones are an integral part of our lives today, and testing could not be done without the mobile version. Therefore, at this stage, we will try to send the archive via Fig. 10 *a* and picture Fig. 10 *b* in mobile device. In the mobile version it is also possible to safely view files



18:31

Safebot
bot

Ви можете безпечно переглянути ваші файли за посиланням: 52.59.248.6 18:26

Логін: user1 18:26

Пароль: user1 18:26

Forwarded message
From Pavlo Mikush
test3.zip
308 B ZIP
18:29

Файл: test3.zip 18:29

Дякую за файл! Зачекайте, поки файл проскануеться... 18:29

Виявлений вірус аа3zip: WinTEICAR_HDB-FOUND 18:29

ВАШ ФАЙЛ Є НЕБЕЗПЕЧНИМ! НЕ СКАЧУЙТЕ ЙОГО! 18:29

Загальний об'єм отриманих файлів: 2.66 MB 18:30

Файлів отримано: 16 18:30

Ви можете безпечно переглянути ваші файли за посиланням: 52.59.248.6 18:30

Логін: user1 18:30

Пароль: user1 18:30

a

18:51

Safebot
bot

18:46

Дякую за повідомлення! 18:46

20210421_181955.jpg
4.7 MB JPG
18:47

Файл: 20210421_181955.jpg 18:47

Дякую за файл! Зачекайте, поки файл просканується... 18:47

Ваш файл є безпечним! 18:47

Загальний об'єм отриманих файлів: 7.38 MB 18:47

Файлів отримано: 18 18:47

Ви можете безпечно переглянути ваші файли за посиланням: 52.59.248.6 18:47

Логін: user1 18:47

Пароль: user1 18:47

b

Figure 10. Testing a virus archive by mobile version Safebot: a) malware archive zip; b) picture jpg

In addition to sending images directly, it is possible to re-send photos to Safebot.

5. Conclusion

This paper demonstrates a developed automated platform that checks for malware files transmitted in the Telegram messenger works as a Telegram bot that tracks messages, detects files, and responds quickly to them. When Safebot receives file, it creates two virtual machines: one with the antivirus installed, scans the received files, another, with a web server installed, translates files from a virtual machine to a web browser, allowing the user to view files that are not directly accessible. Virtual machines are created for each user separately. Leave the cache and restore the saved data the next time you access them. This development is a unique method of scanning files sent via Telegram, as it allows to scan the file before downloading, without the risk of installing virus software on device.

LITERATURA

1. Convenience and security: ranking of the most popular messengers by the degree of their reliability Availaible to web: *https://falcongaze.com/ua/pressroom/publications/research/2.html*
2. Number of monthly active Telegram users worldwide from March 2014 to April 2020. Availaible to web: https://www.statista.com/statistics/ 234038/telegram-messenger-mau-users
3. What is a Telegram virus. Availaible to web: *https://www.wipersoft.com/telegram-virus*
4. What is a hidden cryptocurrency mining? Availaible to web: – Доступно з: *https://spravka.co.ua/posts/virus/uk/virus-virus-majner-ak-viaviti-i-vidaliti-povne-kerivnictvo/*
5. MIKUSH. P.Yu., SHABATURA M.M.: Sandboxes of computer systems as a mechanism of protection against viruses / Proceeding IV All-Ukrainian scientific-practical conference of young scientists, students and cadets "Information Security and Information Technologies", 27.11. 2020. - Lviv, 2020, p. 45-47.
6. SHABATURA M. BALATSKA V.: Exploration Of Computer Network By Vulnerability Scanner Nessus / Bulletin of Lviv State University of Life Safety, 2019, Vol 20, P.6-11. Availaible to web: *https://journal.ldubgd.edu.ua/index.php/Visnuk/article/view/1581*
7. Clamav. Availaible to web: *https://www.clamav.net*