

Forum:[Hack Forums](#)

Thread: [Telegram Bot for searching Telegram accounts](#)

DarkG8 - May 12, 2025, 7:47 am

This code is a Telegram bot that searches for phone numbers and Telegram accounts. It uses telebot library to interact with Telegram API and requests to send requests.

P.S. The code supports only Russian numbers, but you can add localization or make specific for your country

Code:

```
Code import os
import re
import time
import logging
from dotenv import load_dotenv
import telebot
import requests
from requests.adapters import HTTPAdapter
from urllib3.util.retry import Retry

# Configuring logging
logging.basicConfig(
    filename='parser.log',
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)

# Token download from .env
load_dotenv()
BOT_TOKEN = os.getenv("BOT_TOKEN")

# Security Configuration
MAX_RANGE = 1000 # Maximum number range per request
REQUEST_DELAY = 1 # Delay between requests (in seconds)
RETRY_ATTEMPTS = 3 # Number of retries for errors

# Setting up a session with repeats
session = requests.Session()
retry = Retry(
    total=RETRY_ATTEMPTS,
    backoff_factor=0.3,
    status_forcelist=[500, 502, 503, 504]
)
adapter = HTTPAdapter(max_retries=retry)
session.mount("http://", adapter)
session.mount("https://", adapter)
```

```
bot = telebot.TeleBot(BOT_TOKEN)
```

```
def validate_phone(number: int) -> bool:
```

```
    """Checks the number for compliance with the RF format (79XXXXXXXXXX)."""
    return len(str(number)) == 11 and str(number).startswith("79")
```

```
@bot.message_handler(commands=['start'])
```

```
def start_handler(message):
```

```
    """Handler for /start command with validation."""
```

```
    args = message.text.split()
```

```
    if len(args) != 3:
```

```
        bot.reply_to(message, "Format: /start <start_number> <end_number>")
```

```
        return
```

```
    try:
```

```
        start = int(args[1])
```

```
        end = int(args[2])
```

```
    except ValueError:
```

```
        bot.reply_to(message, "Numbers must be integers.")
```

```
        return
```

```
    if not (validate_phone(start) and validate_phone(end)):
```

```
        bot.reply_to(message, "Numbers should be in the format 79XXXXXXXXXX (11 digits).")
```

```
        return
```

```
    if start >= end:
```

```
        bot.reply_to(message, "The starting number must be less than the ending number.")
```

```
        return
```

```
    if (end - start) > MAX_RANGE:
```

```
        bot.reply_to(message, f"Maximum range: {MAX_RANGE} numbers.")
```

```
        return
```

```
    bot.reply_to(message, f"I'm starting to search {end - start + 1} numbers...")
```

```
    for number in range(start, end + 1):
```

```
        try:
```

```
            url = f"https://t.me/{number}"
```

```
            response = session.get(url, timeout=10)
```

```
            if response.status_code == 200:
```

```
                # Use a combination pattern to reduce the risk of errors
```

```
                match = re.search(
```

```
                    r'<a href="/(?P<username>\w+)".*?meta property="og:description"
```

```
content="(?P<bio>.*?)"',
```

```
                    response.text,
```

```
                    re.DOTALL
```

```
                )
```

```
                if match:
```

```
                    username = match.group("username")
```

```
                    bio = match.group("bio").replace("\n", " ")[:100]
```

```
                    bot.send_message(
```

```

        message.chat.id,
        f"Found an account:\n"
        f"Number: `{number}`\n"
        f"Username: @{username}\n"
        f"Bio: {bio}...",
        parse_mode="Markdown"
    )
    time.sleep(REQUEST_DELAY)

```

```

except Exception as e:
    logging.error(f"Error for number {number}: {str(e)}")
    continue

```

```

bot.reply_to(message, "The search is complete. Error log: parser.log")

```

```

if __name__ == "__main__":
    if not BOT_TOKEN:
        logging.critical("Bot token not found in .env!")
        exit(1)
    bot.infinity_polling()

```

Requirements to run:

Create an .env file with the contents:

Code BOT_TOKEN=your_token_bot

Install dependencies:

Code pip install python-dotenv telebot requests

P.S. I'm back in glue and I have so many projects that I absolutely must post here.

???????????? - September 17, 2025, 2:39 am

(May 12, 2025 07:47 AM)DarkG8 Wrote: [\[CITE\]](#) This code is a Telegram bot that searches for phone numbers and Telegram accounts. It uses telebot library to interact with Telegram API and requests to send requests.

P.S. The code supports only Russian numbers, but you can add localization or make specific for your country

Code:

```

Code import os
import re
import time
import logging
from dotenv import load_dotenv
import telebot
import requests
from requests.adapters import HTTPAdapter
from urllib3.util.retry import Retry

```

[h1]Configuring logging[/h1]

```

logging.basicConfig(
    filename='parser.log',
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)

```

)

```
[h1]Token download from .env[/h1]
load_dotenv()
BOT_TOKEN = os.getenv("BOT_TOKEN")
```

```
[h1]Security Configuration[/h1]
MAX_RANGE = 1000 # Maximum number range per request
REQUEST_DELAY = 1 # Delay between requests (in seconds)
RETRY_ATTEMPTS = 3 # Number of retries for errors
```

```
[h1]Setting up a session with repeats[/h1]
```

```
session = requests.Session()
retry = Retry(
    total=RETRY_ATTEMPTS,
    backoff_factor=0.3,
    status_forcelist=[500, 502, 503, 504]
)
adapter = HTTPAdapter(max_retries=retry)
session.mount("http://", adapter)
session.mount("https://", adapter)
```

```
bot = telebot.TeleBot(BOT_TOKEN)
```

```
def validate_phone(number: int) -> bool:
    """Checks the number for compliance with the RF format (79XXXXXXXXXX)."""
    return len(str(number)) == 11 and str(number).startswith("79")
```

```
@bot.message_handler(commands=['start'])
```

```
def start_handler(message):
    """Handler for /start command with validation."""
    args = message.text.split()
```

```
    if len(args) != 3:
        bot.reply_to(message, "Format: /start <start_number> <end_number>")
        return
```

```
    try:
        start = int(args[1])
        end = int(args[2])
    except ValueError:
        bot.reply_to(message, "Numbers must be integers.")
        return
```

```
    if not (validate_phone(start) and validate_phone(end)):
        bot.reply_to(message, "Numbers should be in the format 79XXXXXXXXXX (11 digits).")
        return
```

```
    if start >= end:
        bot.reply_to(message, "The starting number must be less than the ending number.")
        return
```

```
    if (end - start) > MAX_RANGE:
```

```

bot.reply_to(message, f"Maximum range: {MAX_RANGE} numbers.")
return

bot.reply_to(message, f"I'm starting to search {end - start + 1} numbers...")

for number in range(start, end + 1):
    try:
        url = f"https://t.me/{number}"
        response = session.get(url, timeout=10)

        if response.status_code == 200:
            # Use a combination pattern to reduce the risk of errors
            match = re.search(
                r'<a href="(P<username>\w+)".*?meta property="og:description"
content="(P<bio>.*?)"',
                response.text,
                re.DOTALL
            )
            if match:
                username = match.group("username")
                bio = match.group("bio").replace("\n", " ")[:100]
                bot.send_message(
                    message.chat.id,
                    f"Found an account:\n"
                    f"Number: `{number}`\n"
                    f"Username: @{username}\n"
                    f"Bio: {bio}...",
                    parse_mode="Markdown"
                )
                time.sleep(REQUEST_DELAY)

    except Exception as e:
        logging.error(f"Error for number {number}: {str(e)}")
        continue

```

```

bot.reply_to(message, "The search is complete. Error log: parser.log")

```

```

if __name__ == "__main__":
    if not BOT_TOKEN:
        logging.critical("Bot token not found in .env!")
        exit(1)
    bot.infinity_polling()

```

Requirements to run:

Create an .env file with the contents:

Code BOT_TOKEN=your_token_bot

Install dependencies:

Code pip install python-dotenv telebot requests

P.S. I'm back in glue and I have so many projects that I absolutely must post here.

Your Telegram bot is a creative attempt at automating account discovery by scanning phone numbers and checking for associated Telegram profiles. However, technically speaking, the

approach is flawed because the URLs like https://t.me/<phone_number>; do not resolve to actual user profiles unless a user has manually set their username to match their phone number, which is highly unlikely and not a reliable method for enumeration. This kind of tool could be valuable in a simulated phishing or OSINT operation. Telegram's actual phone number to account correlation is only accessible through their internal API (MTProto), and accessing it requires a logged-in user session, which introduces both operational and compliance risks. If you're serious about building this tool, consider switching to a more robust method using libraries like Telethon, while ensuring you have proper authorization.

michaelgagne - *October 26, 2025, 12:43 pm*

that is great