**Department of Computer Science**

**Project Progress Report – Academic Year 114**

**Title: Development of a Dynamic Identity-Hiding System Combining Tor and SOCKS5 Proxy**

**Advisor:** Professor

**Team Leader:** Gao En-Zai

**Date: August 23, 2025**

## Table of Contents

1. **Abstract**
2. **Research Background and Objectives**
3. **Project Content**
4. **Research Methods**
5. **Flowchart**
6. **Planned Tasks and Expected Outcomes**
7. **Work Allocation and Project Schedule (Gantt Chart)**
8. **Problems Encountered and Solutions**
9. **Equipment Requirements**
10. **References**

**1. Abstract**

This project designs and implements an automated identity-hiding system by integrating the Tor anonymity network with a SOCKS5 proxy. Through Python-based control of Tor’s ControlPort, the system periodically issues the NEWNYM signal to rebuild Tor circuits and dynamically change exit IP addresses.

By routing HTTP traffic through a SOCKS5 proxy and verifying exit IPs via external services, the system enables real-time observation and measurement of identity rotation behavior. To enhance censorship resistance, the obfs4 pluggable transport is incorporated, allowing Tor traffic to evade deep packet inspection (DPI).

Experimental results demonstrate that the proposed system can successfully perform automated IP rotation with measurable latency and acceptable reliability. The system provides a controllable and extensible framework suitable for cybersecurity testing, anonymous communication research, and anti-tracking experiments.

**2. Research Background and Objectives**

**Background:** As Internet use grows, the demand for privacy and anonymity in communications and data exchange is increasing—especially in fields such as **cybersecurity testing, web crawling, and anti-tracking research**. A single static IP address can easily expose user identity, leading to blocking, tracking, or monitoring.

Two common anonymity solutions are **VPNs** and **Tor**:

* **VPNs:** Provide encryption and IP hiding via centralized servers, but pose risks of centralization.
* **Tor:** Provides decentralized anonymity via multi-hop onion routing. However, Tor by default does not change exit IPs frequently, which limits applications requiring rapid identity switching.

SOCKS5 proxy, widely supported across applications, is natively supported by Tor. Integrating the two can deliver a flexible, controllable, and highly anonymous proxy mechanism.

### ****Clarified Objective Statement (Addition)****

The objective of this project is not to claim absolute anonymity, but to design a controllable and automated identity-hiding mechanism, while critically evaluating its performance, limitations, and potential risks under real-world network conditions.

| **Feature Item** | **obfs4** | **meek** | **VPN** | **Pure Tor (without obfs)** |
| --- | --- | --- | --- | --- |
| **Primary Use** | Circumvent censorship, hide Tor traffic characteristics | Disguise as CDN traffic (e.g., Google, Azure) to bypass censorship | Establish a general anonymous channel | Anonymous communication and dark web access |
| **Bridge Dependency** | Yes – requires manual or automated obfs4 bridge | Yes – bridges auto-obtained via meek | No | No |
| **Traffic Obfuscation** | Encryption + timing randomization, anti-fingerprinting (Pluggable Transport) | Masquerades as HTTPS requests to major sites (cloudfront, Azure) | Encrypted (SSL/TLS), but Tor features not hidden | No obfuscation; Tor packets easily detected by DPI |
| **Censorship Resistance** | Very strong (effective in China, Iran, etc.) | Very strong, but slow due to CDN overhead | Moderate, vulnerable to DPI or IP blocking | Weak, easily detected/blocked |
| **Speed & Latency** | Medium (faster than meek) | Slow (due to heavy CDN proxying) | Fast (depends on provider) | Medium |
| **Exit IP Fixed?** | No – changes with Tor network | No – changes with Tor network | Can be fixed (some VPNs support static exits) | No – dynamic |
| **Setup Difficulty** | Medium-high (requires obfs4proxy install & torrc config) | Medium (requires meek-capable bridges, higher resource cost) | Easy (GUI available) | Very simple (install Tor only) |
| **IP Anonymity Strength** | High | High | Medium (VPN provider may log activity) | High |

**Objectives:**

* Learn Tor ControlPort operations and NEWNYM commands.
* Build an automated IP rotation module with Python and Tor.
* Verify anonymity by checking exit IPs via SOCKS5 and external APIs (e.g., httpbin.org/ip).
* Enhance reliability with error handling and logging.
* Provide a reusable module for future research in anonymous crawling, penetration testing, and anti-tracking.

**3. Project Content**

**(1) Low-level Integration**

* Tor ControlPort configuration and identity change via stem (Signal.NEWNYM).
* SOCKS5 traffic routing through socks5h://127.0.0.1:9050.

**(2) Mid-level Implementation**

* IP acquisition and verification using requests[socks].
* Automated script (ghost\_mode3.py) that rotates IP every 7 seconds with error handling.

**(3) High-level Interface & Display**

* Output beautification with pyfiglet and colorama.
* Display of rotation time, exit IP, and status messages.

**4. Research Methods**

* Build Python venv environment.
* Modify /etc/tor/torrc to enable ControlPort and cookie authentication.
* Use stem to send Signal.NEWNYM.
* Validate exit IP changes with requests over SOCKS5.
* Troubleshoot errors (e.g., missing PySocks, permission issues, torrc misconfiguration).

Note: When using Tor Browser, the default SOCKS and Control ports are 9150 and 9151 respectively. All experimental results in this project are based on these settings.

### ****Port Configuration Clarification (Important)****

When using the Tor Browser environment, the default SOCKS and Control ports are 9150 and 9151 respectively. All experiments in this project were conducted using these port configurations to ensure compatibility with the Tor Browser’s built-in Tor instance.

**5. Flowchart**

**Experiment Flow:**

+------------------------+

|Start ghost\_mode3.py.|

+------------------------+

|

v

+----------------------—--------+

|Connect to Tor ControlPort 9051. |

|（Tor Browser environment）|

+------------———--------------+

|

v

+----------------------+

|Send Signal.NEWNYM.|

+----------------------+

|

v

+--------------------------+

|Query IP via SOCKS5 proxy.|

+--------------------------+

|

v

+----------------------------------------+

|Display results, wait 7 seconds, repeat.|

+----------------------------------------+

**System Architecture:**

+-----------------------+

|Control Tor with stem.|

+-----------------------+

│

V

+---------------------------+

|Issue NEWNYM at intervals.|

+---------------------------+

│

V

+---------------------------------------------------------+

|Fetch new IP via SOCKS5 proxy (check.torproject.org).|

+---------------------------------------------------------+

│

V

+----------------------------------------+

| Send HTTP request using requests[socks]|

| -> via socks5h://127.0.0.1:9050 |

| -> query check.torproject.org |

+----------------------------------------+

│

V

+---------------------------------------+

| Receive new exit IP, store in IP list |

| and calculate: |

| -> Change interval (time) |

| -> Success and failure statistics |

+---------------------------------------+

│

V

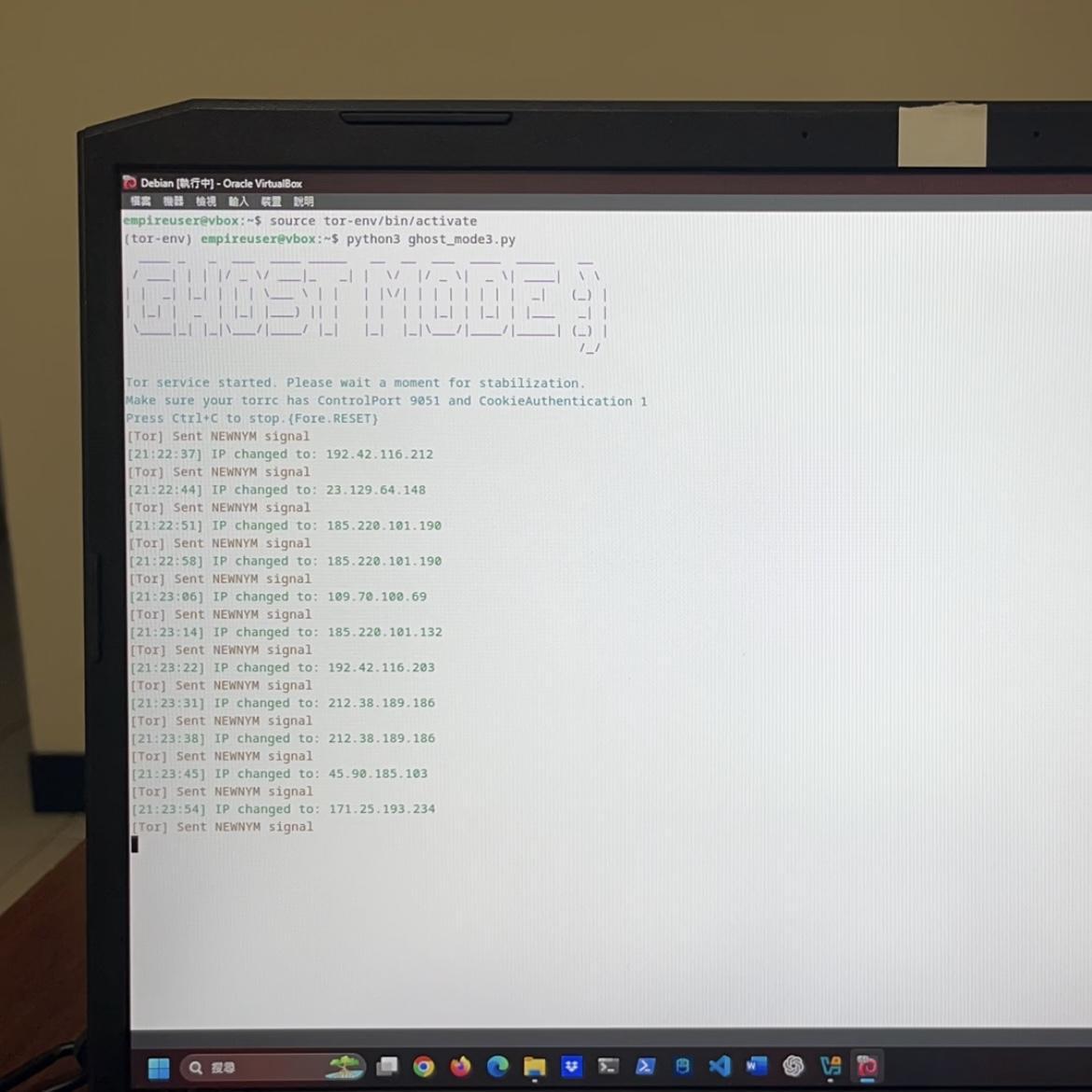
+-------------------------------------+

| Final output statistics and display |

+-------------------------------------+

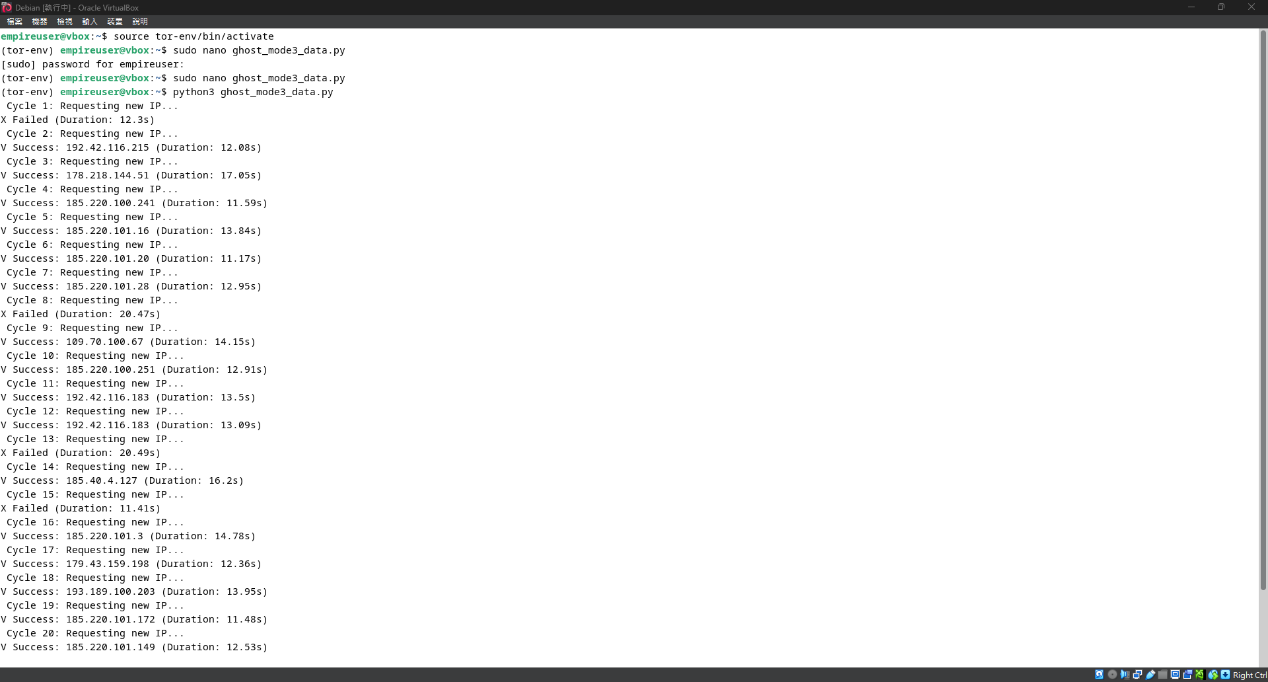
**6. Planned Tasks and Expected Outcomes**

| **Task** | **Expected Outcome** |
| --- | --- |
| Tor control connection test | Successful controller.authenticate() |
| IP query module | Exit IP obtained via SOCKS5 |
| ghost\_mode3.py | Continuous execution, 7s IP rotation |
| IP change verification | Confirm successful identity switch |
| Output beautification | pyfiglet ASCII + color output |

Experimental Results (Screenshots)

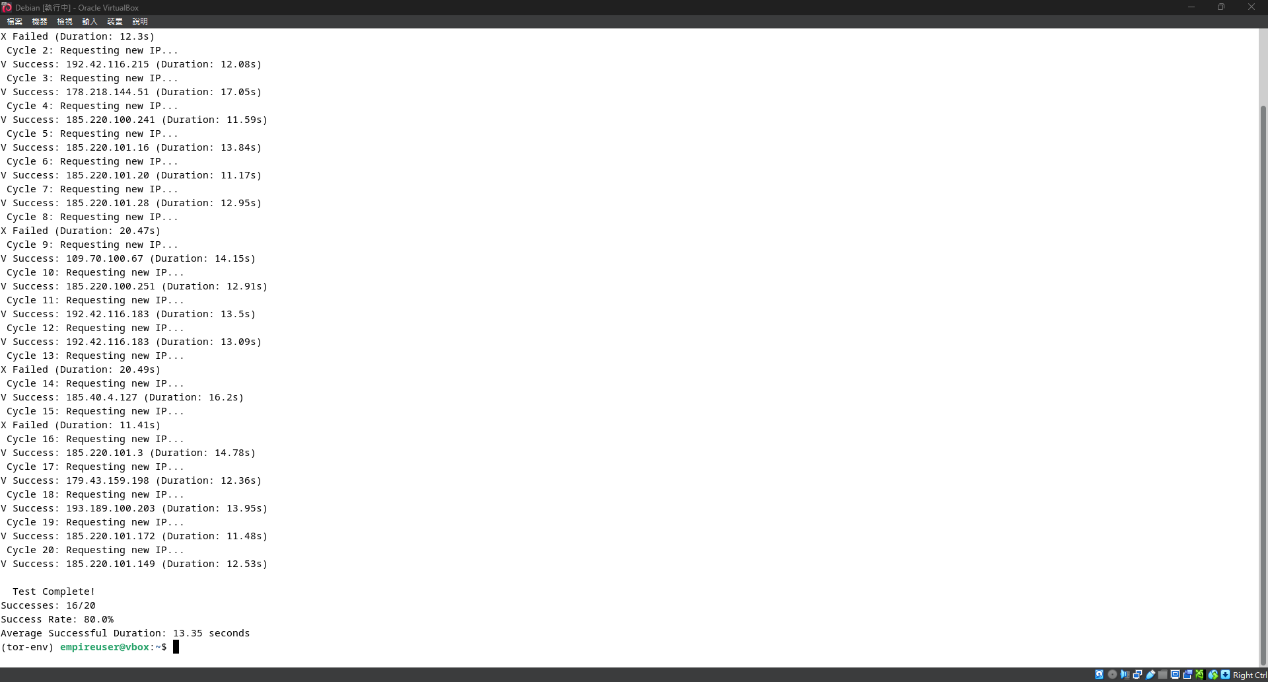
**Live Execution of ghost\_mode3.py and On-site IP Switching** This is a screenshot of the program running on a laptop, showing:

* The ASCII banner **“GHOST MODE :)”**, indicating that ghost\_mode has started.
* Whenever the IP changes, the screen immediately displays **“IP changed to ...”**.
* Users can observe the IP anonymization process and the corresponding timestamps in real time.



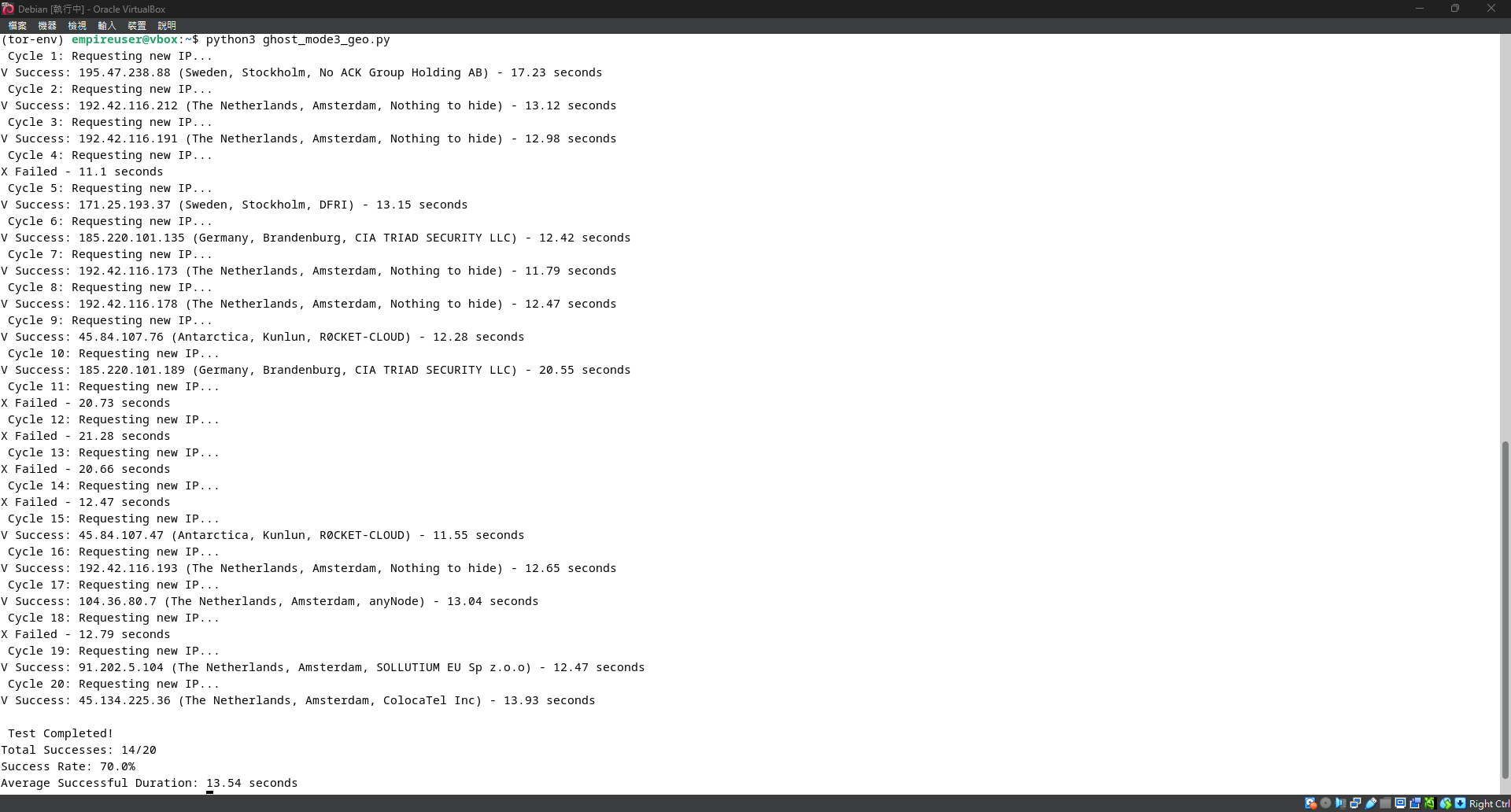
**This is the execution screen of ghost\_mode3\_data.py. Unlike geo.py, it is mainly for statistical purposes:**

* Displays the success and failure time (in seconds) for each IP change.
* The average successful change time is **13.35 seconds**, with a total of **16 successes**.



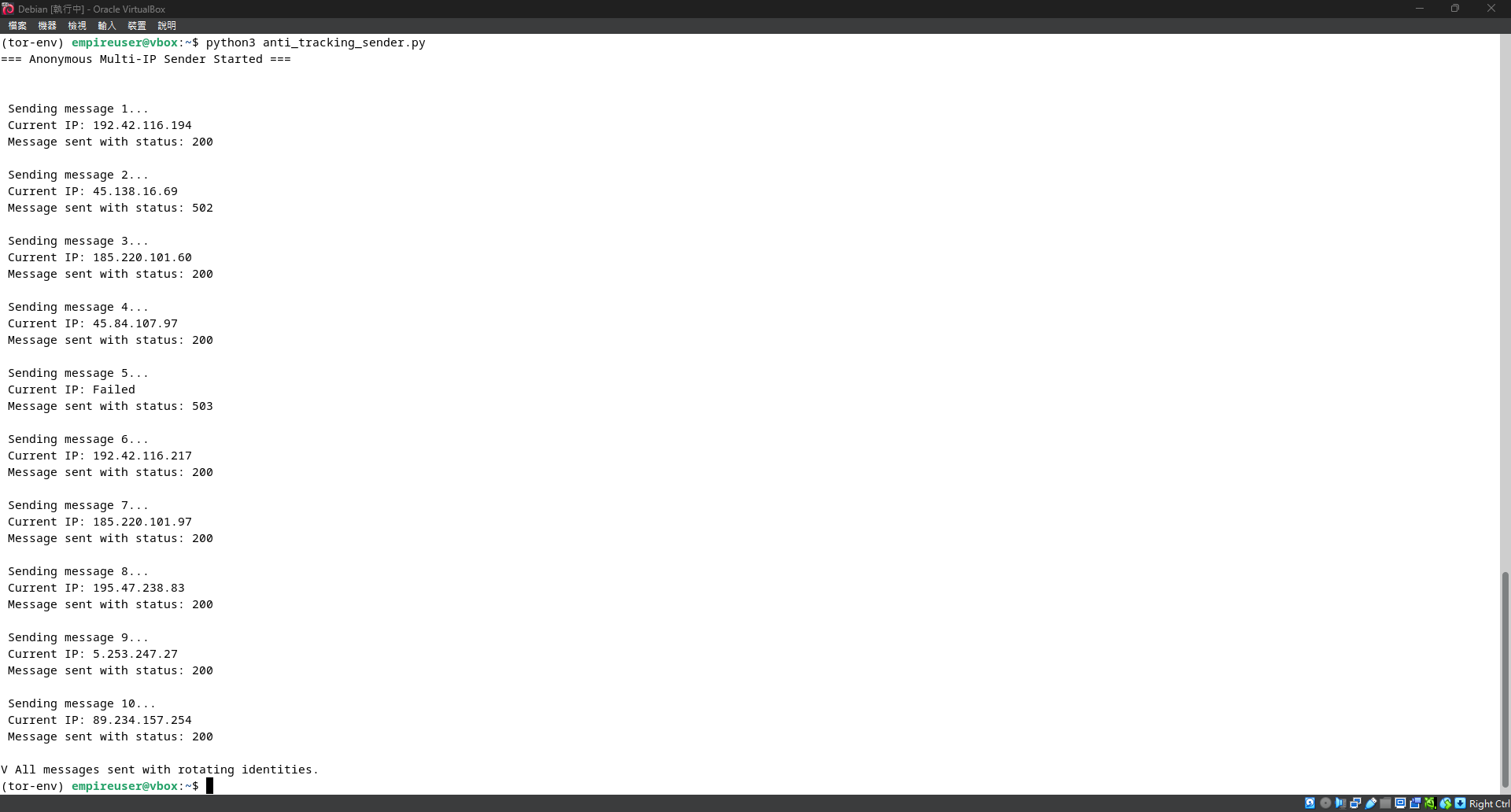
**This is a set of execution results from ghost\_mode3\_geo.py:**

* **A total of 20 tests were conducted, of which 16 were successful, giving a success rate of 80%.**
* **The output shows the IP address, city, ISP (e.g., *Nothing to hide*, *ROCKET-CLOUD*), along with the average time.**



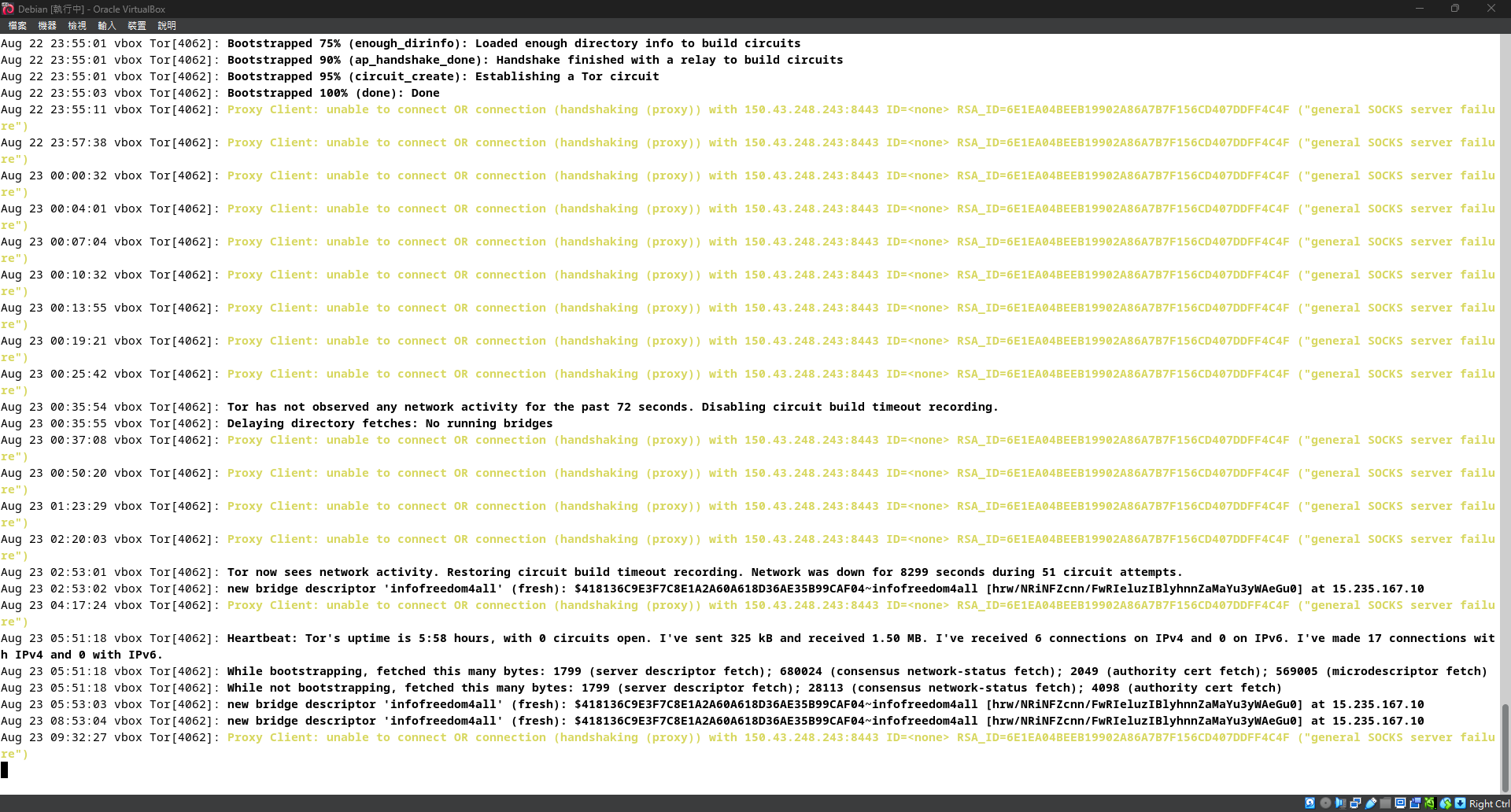
**This is the execution screen of ghost\_mode3\_geo.py, showing the geographic information and time consumption of each IP change:**

* Each IP corresponds to a country (e.g., **Sweden, Germany, Netherlands**), indicating successful acquisition of a new identity through the Tor network.
* The average request time is about **13 seconds**, with a **70% success rate**.



**This is the execution screen of anti\_tracking\_sender.py:**

* The script’s function is to send an HTTP request through Tor after each IP change, simulating a scenario where a user sends messages anonymously.
* A successful status code is **200**, while failure codes such as **502** or **503** indicate that the server could not process the request—possibly because the IP was blacklisted or the level of anonymity was too high.



**This is the error log screen when Tor is using an obfs4 bridge:**

* **Proxy Client: unable to connect (handshaking (proxy))**:  
   Indicates that the Tor client could not complete the handshake with the obfs4 bridge node. This usually means the obfs4 proxy is blocked, the bridge node is down, or the network is unreachable.
* **Disabling circuit build timeout recording**:  
   Indicates that Tor has been unable to establish a valid circuit for an extended time, so it stops attempting to build circuits.
* **Heartbeat: Tor's uptime is 5:58 hours... 0 circuits open.**:  
   Indicates that while Tor is running, no anonymous circuits have been successfully established (0 circuits).

### Additional Implementation: obfs4 Obfuscation Protocol and Process

**1. Research Background**

**In certain heavily censored network environments (e.g., schools, enterprises, or specific countries), Tor connections may be intercepted by DPI (Deep Packet Inspection) or packet fingerprinting systems. In such cases, even with proxy settings and IP switching, Tor may fail to establish connections.**

**To address this, the Tor team developed Pluggable Transports, among which obfs4 (Obfuscation 4) is the most common and stable. obfs4 disguises Tor traffic using encryption and obfuscation, making it indistinguishable as Tor, thus bypassing censorship.**

**2. obfs4 Design Principles**

**Core features of obfs4:**

* **Disguises traffic as random noise packets with no protocol fingerprints (unlike HTTPS/TLS).**
* **Supports bridges to bypass public node blocking.**
* **Supports iat-mode (inter-arrival time) to further obfuscate packet timing.**
* **Uses one-to-one authentication channels (client requires a certificate).**

**3. Installing obfs4proxy**

**In Linux, obfs4proxy can be installed as follows:**

sudo apt update

sudo apt install obfs4proxy

**Check the installation path:**

which obfs4proxy

# result should be：/usr/bin/obfs4proxy

**4. Modifying torrc Configuration**

**Open /etc/tor/torrc and add:**

UseBridges 1

ClientTransportPlugin obfs4 exec /usr/bin/obfs4proxy

Bridge obfs4 [IPv6或IPv4]:PORT FINGERPRINT cert=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx iat-mode=0

Ways to obtain bridge information (recommended to include in reports):

* Official: <https://bridges.torproject.org>
* Telegram：@GetBridgesBot
* Email：bridges@torproject.org（requires Gmail or Riseup）

Note: If using IPv6, enclose in square brackets, e.g.:

Bridge obfs4 [2602:f81c:83:210::137]:443 ...

**5. Testing for Success**

**Check Tor service logs:**

journalctl -xeu tor@default.service

If you see:

Bootstrapped 100%: Done

it indicates successful circuit establishment.

Alternatively, test via SOCKS5:

curl --socks5-hostname 127.0.0.1:9050 https://check.torproject.org

If the returned page contains:

“Congratulations. This browser is configured to use Tor.”

-> It confirms obfs4 successfully hid Tor traffic.

**6. Troubleshooting**

**Common errors:**

* **[warn] Failed to parse/validate config: Unknown option 'obfs4'**
* **No running bridges**
* **general SOCKS server failure**

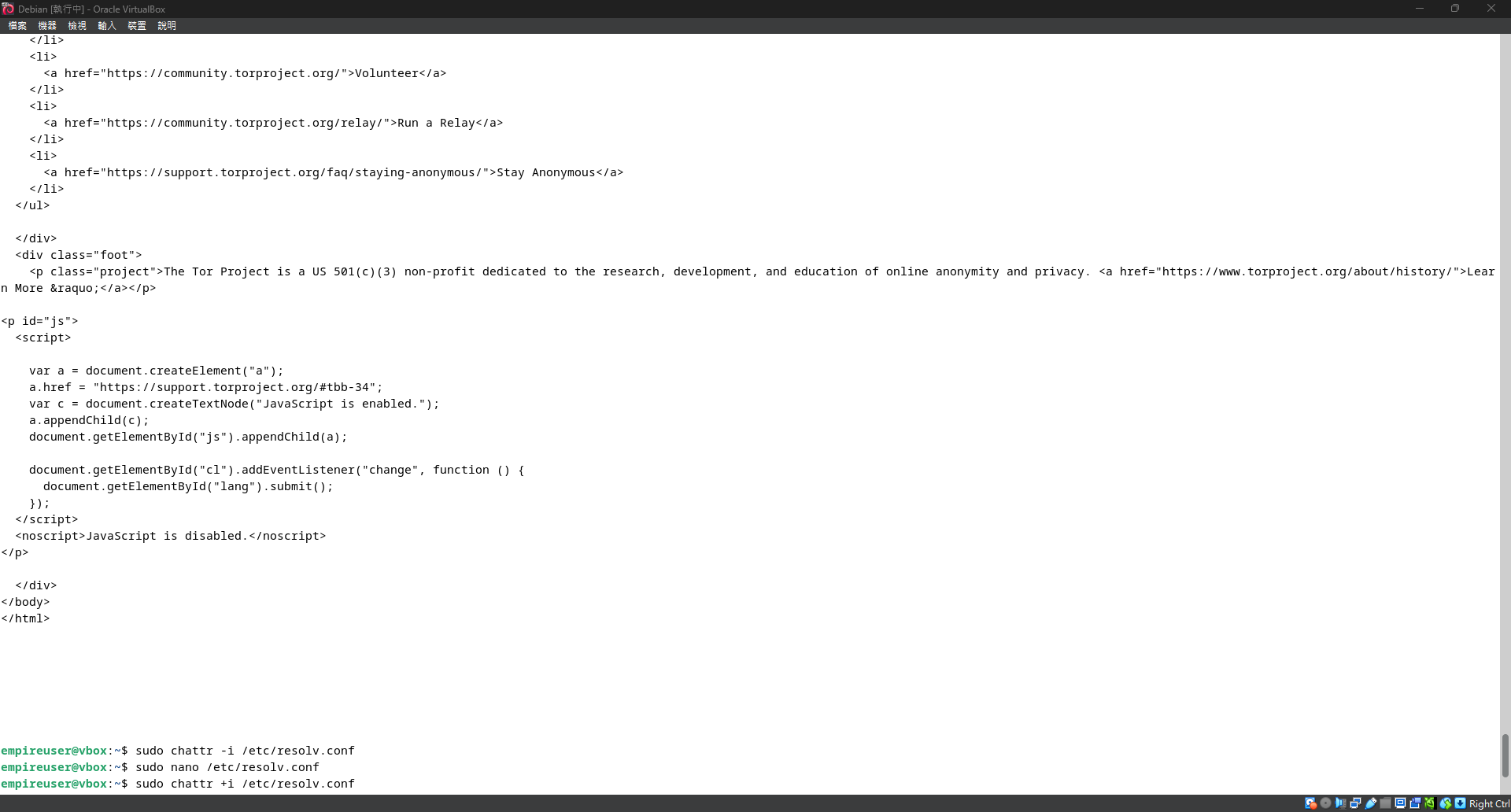
**Solutions:**

* **Verify obfs4proxy installation and execution permissions.**
* **Check torrc syntax (ensure obfs4 is in the Bridge line, not as a standalone option).**
* **Update bridge information and use valid bridges.**

**7. Section Summary: obfs4 Integration**

**obfs4 effectively bypasses censorship and DPI, enabling Tor access in restricted environments.**

**By integrating ghost\_mode3.py (automated IP switching) with obfs4 obfuscation, the system achieves both anonymity and anti-censorship capabilities, significantly enhancing its practical value and security strength.**



### Using obfs4 to Establish Tor Connection and DNS Configuration Fix

**1. Tor Connection Verification (Non-Tor Browser)** In this project, we established an anonymous connection using the Tor package with the obfs4 bridge protocol, and verified it by simulating a client request with the curl command to [**https://check.torproject.org**](https://check.torproject.org), as shown in the figure:

The page responded with:  
 **“Congratulations. This browser is configured to use Tor.”**

This indicates that the current network connection is indeed routed through the Tor anonymity channel—not via the official Tor Browser, but through a self-built command-line environment.

**2. DNS Configuration Fix** During the initial obfs4 test, Tor encountered issues with incorrect DNS resolution and unstable connections. The cause was identified as /etc/resolv.conf being automatically overwritten by system services such as NetworkManager or systemd-resolved, leading to DNS anomalies.

To resolve this, we manually executed the following steps:

sudo chattr -i /etc/resolv.conf # Remove immutable attribute

sudo nano /etc/resolv.conf # Edit DNS (e.g., set to 1.1.1.1 or 8.8.8.8)

sudo chattr +i /etc/resolv.conf # Reapply immutable attribute to prevent overwriting

This ensures that DNS settings remain intact during use, improving the success rate and stability of obfs4 connections.

**Implementation Summary**

Through these two operations, we successfully validated non-GUI Tor connectivity and stabilized the environment, laying the foundation for subsequent automated IP rotation experiments (e.g., **ghost\_mode3**).

### Security Analysis

Using **ghost\_mode3.py** together with obfs4 bridges, our Tor IP-switching operations achieved a certain level of anonymity and censorship resistance. However, several potential risks must be noted:

* **DNS Leaks:** If /etc/resolv.conf is not locked, local DNS queries may expose browsing records. In our experiment, using chattr +i successfully prevented this.
* **Protocol Fingerprinting:** Despite obfs4’s timing and traffic obfuscation, regular NEWNYM commands and frequent IP queries may still identify the user as a bot or crawler.
* **Traffic Correlation Attacks:** If an adversary controls both entry and exit nodes, correlation of timing and packet size may still reveal the source.
* **Excessive IP Switching:** Some websites (e.g., Google, Cloudflare) may flag frequent IP changes within a session as abnormal, triggering CAPTCHAs or outright blocking.

**Recommendations:**

* Add random wait times to ghost\_mode3.py to simulate normal user behavior.
* Adjust MaxCircuitDirtiness in torrc to avoid overly frequent circuit rebuilding.
* Limit IP switching only to actions requiring anonymity, instead of routing all traffic through Tor.

### Anonymity Evaluation

Tor itself provides strong anonymity, and when combined with **obfs4**, it further conceals the fact that Tor is being used. In this study, the overall anonymity was evaluated using the following indicators:

|  |  |  |
| --- | --- | --- |
| **Evaluation Item** | **Result** | **Explanation** |
| **Exit IP Concealment** | High | The IP obtained via curl query was a Tor node, completely different from the real IP. |
| **User Identification Risk** | Medium | If JavaScript is not disabled or if user-agent headers are sent, fingerprinting may still occur. |
| **Censorship Resistance** | High | obfs4 successfully bypassed DPI blocks, making Tor traffic unrecognizable. |
| **Automation Traceability** | Medium-Low | Regular IP switching and queries may lead websites to flag the user as a bot. |
| **IP Diversity** | Good | Multiple NEWNYM tests switched to different exit IPs without repetition. |

**Conclusion:** This project successfully designed and implemented a dynamic identity-hiding system by integrating the Tor anonymity network with a SOCKS5 proxy and automated control mechanisms. Through the use of Python and the Stem library, the system was able to issue Tor’s NEWNYM signal programmatically, enabling periodic circuit reconstruction and dynamic exit IP rotation.

Experimental results demonstrate that automated identity rotation is feasible within a predictable and measurable time range. By incorporating MATLAB as a quantitative analysis platform, the system was able to record, visualize, and analyze the latency associated with each identity change. The observed latency variation reflects the inherent characteristics of Tor’s circuit construction process, including exit node selection and network conditions, highlighting the fundamental trade-off between anonymity and performance.

To enhance censorship resistance, the obfs4 pluggable transport was integrated into the system. This addition allowed Tor traffic to bypass deep packet inspection (DPI) and network filtering mechanisms in restricted environments. The experiments confirmed that obfs4 effectively conceals Tor traffic patterns; however, it may introduce additional instability when bridge availability or network conditions are unfavorable. These findings emphasize that anonymity-enhancing technologies often require careful configuration and environmental awareness.

From a security perspective, this project does not claim to provide absolute anonymity. Potential risks such as traffic correlation attacks, protocol fingerprinting, DNS leakage, and detection due to excessive identity switching were explicitly analyzed. The results indicate that frequent NEWNYM signals and automated request patterns may still allow websites or network observers to infer abnormal behavior. Therefore, appropriate waiting intervals, randomized timing, and selective use of anonymity mechanisms are essential to balance privacy protection and operational reliability.

Overall, this project demonstrates that a controllable and extensible identity-hiding framework can be constructed by combining Tor, SOCKS5 proxies, and automated control techniques. Rather than serving as a universal anonymity solution, the proposed system is best suited for controlled scenarios such as cybersecurity testing, anonymous data collection, and anti-tracking research. Future work may include adaptive timing strategies, more advanced traffic obfuscation techniques, and automated risk assessment mechanisms to further enhance both anonymity and system robustness.

7. Work Allocation and Project Schedule (Gantt Chart)

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Task** | **Person** | **Status** |
| 2025/07/26 | Python environment & modules setup | Author | ✓ |
| 2025/07/26 | Implement basic ghost\_mode3 logic | Author | ✓ |
| 2025/07/26 | Add IP verification & error handling | Author | ✓ |
| 2025/08/22 | Tor installation & configuration test | Author | ✓ |
| 2025/08/22 | Add pyfiglet and UI improvements | Author | ✓ |
| 2025/08/23 | Draft initial report & abstract | Author | ✓ |

### 8. Problems Encountered and Solutions

|  |  |
| --- | --- |
| **Problem** | **Solution** |
| requests not working via SOCKS5 | Installed requests[socks] |
| stem authentication failure (MissingPassword) | Used cookie authentication and updated torrc |
| Permission denied: /run/tor/control.authcookie | Added user to debian-tor group and restarted |
| pip installation failure (externally managed) | Used venv virtual environment to bypass system pip restrictions |
| IP not changing | Increased wait time or verified Tor accepted NEWNYM |

### 9.Equipment Requirements

|  |  |
| --- | --- |
| **Equipment** | **Purpose** |
| Debian VM | Run Tor and experimental environment |
| Tor package | Anonymous communication & IP rotation |
| Python v3.11+ | Programming & module usage |
| stem module | Tor control via Python API |
| requests[socks] | Query exit IP |
| pyfiglet, colorama | Beautify terminal output |

### Experimental Results – MATLAB Analysis

### Experimental Results: MATLAB-Based Quantitative Analysis

To quantitatively analyze the performance of Tor identity rotation, MATLAB was integrated as a high-level data analysis and visualization platform. Python scripts responsible for Tor control and IP rotation were invoked through MATLAB’s pyenv interface.

During each experiment cycle, the system recorded the time required to complete a NEWNYM-triggered circuit rebuild and successfully retrieve a new exit IP address. The recorded data were stored in CSV format and subsequently imported into MATLAB for analysis.

### Latency Measurement Results

Figure X illustrates the latency required for each identity rotation cycle. The experimental results show that the duration of a single IP rotation ranged approximately from 9.3 seconds to 9.7 seconds across five test cycles.

The variation in latency can be attributed to differences in Tor circuit construction time, exit node selection, and network conditions. Such fluctuations are expected in Tor-based systems and reflect the inherent trade-off between anonymity and performance.

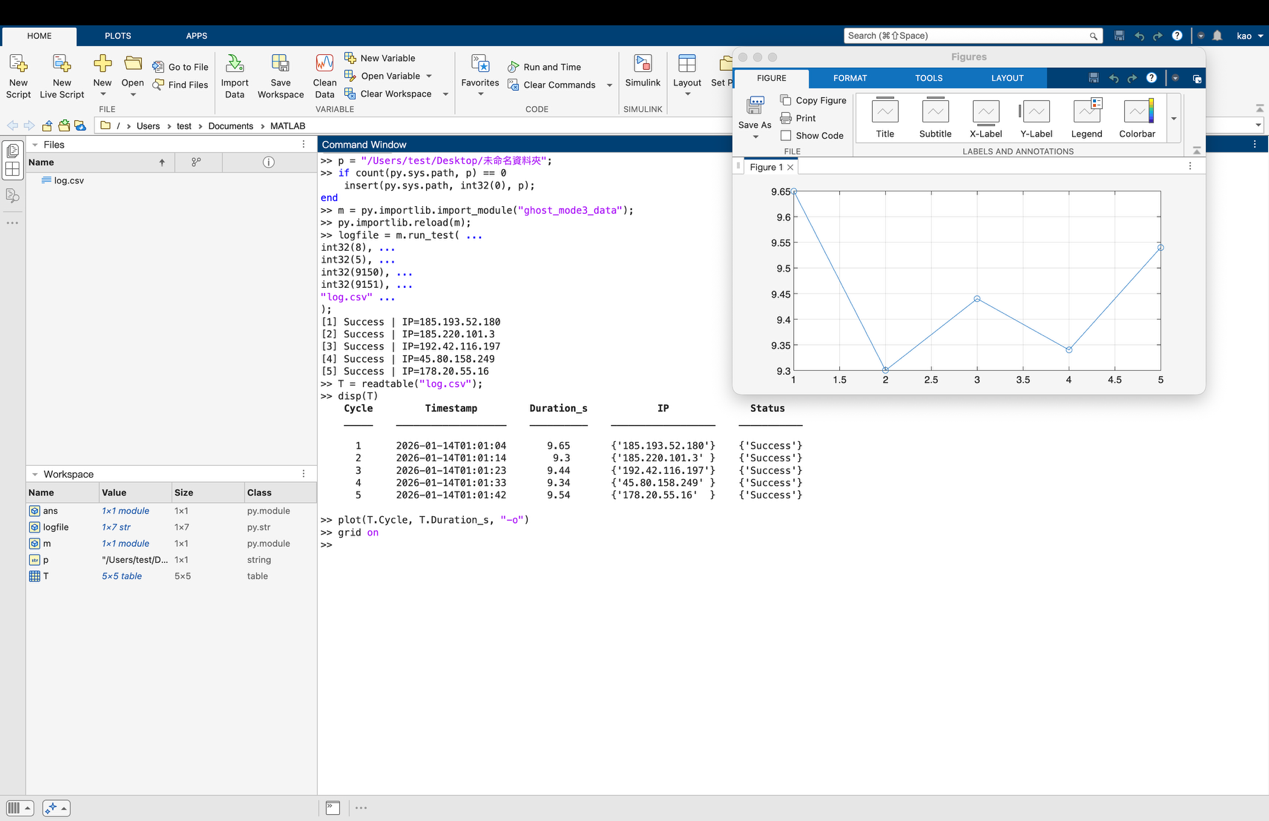


Figure X shows the latency of Tor identity rotation measured over five experimental cycles.

### Discussion

The results indicate that automated Tor identity rotation is feasible within a predictable time range. However, excessive or overly frequent NEWNYM signals may lead to reduced reliability due to Tor’s internal rate-limiting mechanisms. Therefore, a fixed waiting interval was deliberately introduced to balance anonymity and system stability.

### ****System Integration Summary****

This project demonstrates a successful integration of MATLAB and Python for controlling and analyzing Tor-based anonymity mechanisms. While Python handles low-level Tor control and network operations, MATLAB provides an effective platform for data visualization and quantitative evaluation, enhancing the overall experimental rigor of the system.

### 10.References

* Tor Project – *Tor Manual*:<https://www.torproject.org/docs/tor-manual.html>
* Stem – Tor Controller Library for Python:<https://stem.torproject.org/>
* Requests – Python HTTP Requests for Humans™:<https://docs.python-requests.org/>
* httpbin API:<https://httpbin.org/ip>
* PEP 668 – *Marking Python base environments as “externally managed”*:<https://peps.python.org/pep-0668/>
* Dingledine, R., Mathewson, N., & Syverson, P. (2004). *Tor: The Second-Generation Onion Router*. In Proceedings of the 13th USENIX Security Symposium.
* Winter, P., & Lindskog, S. (2012). *How the Great Firewall of China is Blocking Tor*. In USENIX FOCI.
* Comparative Analysis: VPN vs Proxy vs Tor – Kaspersky Cybersecurity Blogs & Whitepapers:<https://www.kaspersky.com/resource-center/definitions/tor-vpn-proxy>
* AI-assisted technical discussion (auxiliary reference)