Traffic Correlation Based De-Anonymization of TOR Onion Services

Ravipally Sai Nikhila Reddy[1], Pendem Sampath[2], B Sravan[3],
Mr. Nirmal Keshari Swain[4], Satheesh Kumar S[5]
[1,2,3]*Department of Information Technology, Vardhaman College of Engineering (Autonomous), Hyderabad-Telangana, 501286, India*
[4,5]*Department of Information Technology, Vardhaman College of Engineering (Autonomous), Hyderabad, India*

***Abstract*—The recommended attack strategy depends on keeping an eye on packets with imprints from the target's negotiated guard. By hiding the real IP addresses of the participating attendants, Tor's hidden services are intended to protect privacy. Tor's architecture, which occupations layered encryption to precaution user anonymity at several levels, presents a major analytical problem. It is challenging to distinguish between Tor cells the basic building blocks of data transmission within the network—based solely on packet characteristics because they are all the same size and uniformly encrypted. This study emphasizes how strong Tor's privacy safeguards are and how crucial it is to deal with any risks to user anonymity. Anonymity networks have become increasingly popular among Internet users due to growing worries about online privacy. Of them, Tor is the most popular because it allows users and services (including hidden services) to remain anonymous. However, this anonymity is frequently used for unlawful activities, like running black marketplaces, hosting botnet command and control servers, and disseminating content that is restricted or forbidden. As a result, a variety of organizations, including governmental and law enforcement organizations, have expressed a growing interest in creating techniques to de-anonymize Tor users, disrupt its operation, and undermine its anti-censorship characteristics. In this survey, we look at current Tor network threats and provide a comprehensive review of de-anonymization techniques that target both users and hidden services. We examine how these attacks are executed and evaluate their likelihood in the real world. Finally, we outline improvements made to the Tor architecture with the goal of improving the efficacy of these de-anonymization techniques and bolstering network security in general.**

***Index Terms*—TOR network, Onion Service, de-anonymisation, robustness**

I. INTRODUCTION

Over the past several decades, the rise of online services has significantly influenced the daily lives of Internet users, raising an important concern: how can individuals browse the Internet while safeguarding their privacy? Individuals such as whistle-blowers and those living under restrictive regimes particularly rely on internet anonymity to safeguard their identities. Other use cases of anonymous networks include sensitive communications of military and business organisations operating over the public Internet [1] have faced growing concerns, prompting examination and development into anonymous communication systems [2]. Early solutions like Mix-Net [3], Babel [4], and Mix-minion [5], however, struggled with high latency, limiting their adoption. These have since been replaced by more efficient low-latency systems, which we now explore.

Since Tor is widely adopted to ensure user anonymity, most cyberattacks aim to reveal either the identities of users or the hidden services they access [6]. To counter such de- anonymisation threats, privacy-focused researchers have implemented stronger security protocols and resolved known exposures. Additionally, the Tor network has seen significant growth in both its user base and infrastructure over time. This expansion, coupled with continuous improvements in its security architecture, has played a key role in mitigating the effectiveness of many legacy de-anonymisation attacks.[7]

A clear and structured taxonomy is essential for understanding de-anonymization attacks. In this context, we propose a novel complicated classification

framework specifically designed to categories de-anonymization attacks directing the Tor network. Our organization employs dissimilar norms at each level to ensure an organized and all-inclusive classification of various attack strategies. Each attack is analyzed based on the specific Tor circuit workings involved and the procedures used for execution [8]. Furthermore, we evaluate the real-world applicability of these attacks. In addition, we observe how research efforts have predisposed the evolution of Tor since its inception, detecting key indicators that have shaped its retreat posture. These movements have concentrated many previously reasonable attacks obsolete [9]. This study aims to serve as a introductory reference for scholars and enthusiasts seeking to extend their understanding or contribute to further investigations in this domain.

Academic research mostly focuses on technical susceptibilities of the Tor protocol and evaluates several angles to break the Tor anonymisation and potential counter measures. Protruding examples are fingerprinting attacks where local attics droppers could identify the destination of encrypted Tor traffic by similar patterns or including mean Tor relays in the linkage to occupy in traffic analysis attacks, linking the origin and destination of Tor traffic and effectually deanonymizing users. The practical impact of many predispositions is challenging to quantify, as experiments are usually limited to test settings. Larger field tests are not carried out due to probable harm to potential Tor users. Also, complex occurrences require a large budget, access to highly qualified recruits, and they are still theoretically measurable in the privacy-focused Tor community.

A. Motivation

By focusing on the inherent design of Tor and its multi- layered encryption system, the research aims to evaluate how resilient Tor is against sophisticated attacks targeting its anonymity. The reference to watermarking and packet analysis suggests a deeper exploration into possible attack vectors that undermine the secrecy of both users and hidden services. Thus, the motivation is to explore both the strengths and vulnerabilities of Tor, to better understand the scope of potential risks to privacy, and to provide a foundation for future improvements in the Tor network to protect against de- anonymisation attacks. This helps ensure that Tor can continue to serve as a reliable tool for those seeking privacy, while also minimizing its exploitation by malicious actors.

B. Contribution

The research evaluates the practicality of these attacks, considering the challenges posed by Tor's multi-layered encryption and uniform Tor cell structure. It examines how attackers might attempt to bypass these inherent privacy features, such as through the monitoring of watermarked packets in compromised. This research aims to give a clear overview of the automated tools available for detecting and stopping fake news guards.

C. Literature Survey

In recent years, there has been a lot of research on network tunnelling and anonymous communication, particularly in relation to de-anonymisation attacks, encrypted routing, and privacy preservation. As the Tor network and other anonymous systems have grown in popularity, researchers and law enforcement organisations have directed more of their attention towards finding weaknesses in these systems and coming up with solutions to protect user privacy. A comprehensive overview of significant research initiatives pertinent to the creation of tunnelling mechanisms and the investigation of de-anonymisation methods is provided by this review of the literature.

The most popular anonymity network is called Tor, or The Onion Router, which protects user identity by directing encrypted traffic through a number of volunteer relays. By describing its layered encryption, relay architecture, and intended use for privacy-conscious communication, Dingledine et al. presented the architecture of Tor as a second- generation onion routing system. However, a number of researchers have shown that, in certain situations, Tor's anonymity can be jeopardized. For example, circuit fingerprinting attacks were proposed by Kwon et al., demonstrating that the origin or destination of a hidden service could be discovered through passive monitoring of traffic patterns. In a similar vein, Chen et al. illustrated the potential for deanonymizing users or servers on the Tor network by showcasing the efficacy of attacks utilizing malicious guard relays and binding relay techniques.

II. LITERATURE REVIEW

The Tor network is widely used for providing anonymity to users and hidden services. However, various research efforts have focused on de-

anonymising users and services on Tor. Multiple research efforts have investigated ways to compromise the anonymity of users and services operating within the Tor ecosystem. One such study, Towards De-anonymising Hidden Services, examines various attack methodologies that aim to reveal the identity of hidden services (THSs) on Tor. It categorizes attacks based on cost and impact, assessing 25 papers and identifying 20 as relevant.[10] Key attack techniques discussed in this research include circuit fingerprinting attacks, which passively analyse Tor circuits to identify hidden service communication patterns, and man-in-the-middle (MitM) attacks that intercept and modify Tor traffic to reveal the IP addresses of users.[11]

Another significant work, The Anonymity of the Dark Web: A Survey, provides a comprehensive review of anonymity technologies such as Tor, I2P, and Freenet, classifying attacks into three main types: traffic analysis attacks (including fingerprinting, timing), network-level attacks (compromising relays or entry guards), and application-level attacks.[12] Similarly, Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services presents a novel passive attack that analyses the traffic patterns of circuits used by hidden services.[13] The method validated high precision in identifying hidden services and used traffic analysis to evaluation website access with notable accuracy.

In another survey, De-Anonymisation Attacks on Tor: A Survey, countless attacks are categorized based on their objectives, including entry-exit attacks that conciliation both entry and exit nodes to correlate traffic, intersection attacks that identify users based on access timing, and cryptographic weakness exploits that leverage flaws in Tor's encryption protocols.[14] To counter these attacks, multiple countermeasures have been proposed, such as traffic padding to add random noise to traffic and obfuscate patterns, improved circuit selection algorithms to reduce reliance on a small number of relays, and onion services hardening by implementing stronger encryption and obfuscation techniques.[15]

While Tor remains a robust anonymity tool, ongoing research continues to uncover liabilities that can be exploited for de-anonymisation. Future research should focus on enhancing the security of entry and exit nodes, deploying advanced machine learning techniques for detecting and mitigating attacks, and developing decentralized solutions to improve resilience against adversaries. This literature review highlights the evolving landscape of anonymity research, showcasing the challenges and advancements in securing online privacy.[16]

III. METHODOLOGY

De-anonymising the Tor network using Sybil attacks and split tunnelling requires a strategic combination of network infiltration, traffic analysis, and correlation techniques to undermine the privacy protections of Tor users [Fig 1]. A Sybil attack involves an attacker introducing numerous rogue relays into the Tor system, increasing the odds of intercepting user communications. By increasing the likelihood that a user's traffic is routed through negotiated nodes. By carefully monitoring packet sizes, timings, and patterns, the attacker can perform traffic fingerprinting and correlation analysis to link superficially anonymous activity within Tor to a real-world source.[17] The more nodes controlled by the attacker, the higher the probability of capturing traffic and de-anonymizing users.

On the other hand, split tunnelling introduces another layer of vulnerability by allowing certain user traffic to bypass Tor and travel through a regular internet connection. This can happen due to misconfigured VPN settings, DNS leaks, WebRTC leaks, or even background applications that send data outside of the Tor network. When this occurs, an attacker monitoring the external network can capture leaked metadata, such as IP addresses, request timestamps, and unencrypted data packets. By cross-referencing this leaked data with traffic observed within Tor, an adversary can establish correlations that compromise anonymity.[18] For example, if a user unknowingly allows DNS queries to bypass Tor, an attacker could log the DNS requests and match them with the timing and frequency of Tor-based communications. Additionally, if an attacker injects uniquely identifiable payloads into Tor traffic and later detects those payloads on the non-Tor network, it becomes possible to link a Tor user's activity to their real- world identity.

To execute such an attack effectively, an adversary would first deploy a network of malicious Tor nodes, ensuring that they have a significant presence within the network's routing infrastructure. Next, they would

employ machine learning algorithms and statistical analysis to detect traffic patterns that match between the Tor network and the open internet. By using advanced timing correlation, even slight delays in traffic can be exploited to trace users across network boundaries. Additionally, if the attacker has access to compromised ISPs or network infrastructure, they could further enhance their ability to correlate Tor entry traffic with exit traffic, further reducing the effectiveness of Tor's anonymity features.

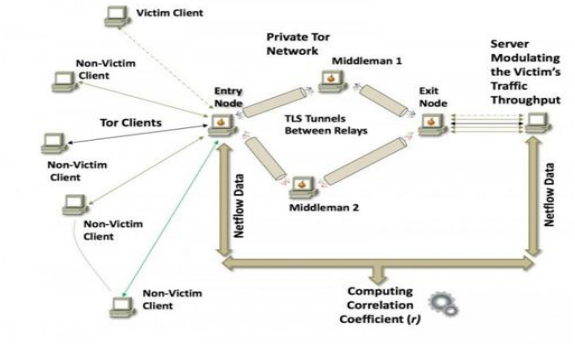

Fig. 1. Traffic analysis attack.

To defend against these attacks, Tor users should employ strict VPN configurations to disable split tunnelling, ensuring that all traffic is routed exclusively through Tor. Encrypted DNS over Tor should be enforced to prevent DNS leaks, and circuit diversity techniques should be used to avoid repeatedly using the same Tor nodes that might be compromised by an attacker. Additionally, the Tor Project must continuously monitor the network for Sybil nodes by analyzing relay behaviours and removing suspicious nodes that attempt to capture excessive amounts of traffic.

Correlation Models for Traffic Analysis

To establish deanonymization links between TOR traffic and split-tunnelled traffic, statistical correlation models are applied:

1. Pearson Correlation Coefficient

$$\rho\{X,Y\} = \frac{\sum(X_i - \{\bar{X}\})(Y_i - \{\bar{Y}\})}{\left\{\sqrt{\sum(X_i - \{\bar{X}\})^2}.\sqrt{\sum(Y_i - \{\bar{Y}\})^2}\right\}}$$

where: X = TOR packet timings

Y = split-tunnelled packet timings.

2. Mutual Information (MI)

Measure shared information between TOR flows and leaked flows, detecting deanonymization signals even under padding

3. Hidden Markov Models (HMMs)

Used to model burst patterns and timing states in TOR circuits. HMM transition probabilities reveal similarities between anonymised and leaked flows.

Although this methodology demonstrates the potential for de-anonymising users on the Tor network, it is critical to recognise the ethical and legal implications of such research. Unauthorised de-anonymisation violates fundamental privacy rights and universal cybersecurity laws. However, studying these liabilities is essential for improving privacy-preserving technologies and consolidation Tor against adversarial threats. By understanding and mitigating these attack routes, security researchers and the Tor community can enhance the suppleness of anonymous communication networks [Fig 2], ensuring that users remain protected against surveillance and cyber threats.

Algorithm Steps:

Algorithm 1: Deanonymization via Split Tunnelling

1. Deploy Sybil relays to capture TOR traffic.
2. Configure VPN with split tunnelling to force leaks (DNS, SIP, WebRTC).
3. Extract features:

$$F = \{\mu_{delay}, \sigma_{jitter}, P_{loss}, T_{throughput}\}$$

4. Compute correlation using:
   - Pearson (timing similarity).
   - MI (information overlap).
   - HMM (sequence matching).
5. If correlation factor ρ>0.8, entity deanonymized.

Equation for decision:

$$D = \begin{cases} 1 \text{ if } \rho \geq 0 \\ 0 \text{ Otherwise} \end{cases}$$

Where θ is a correlation threshold.

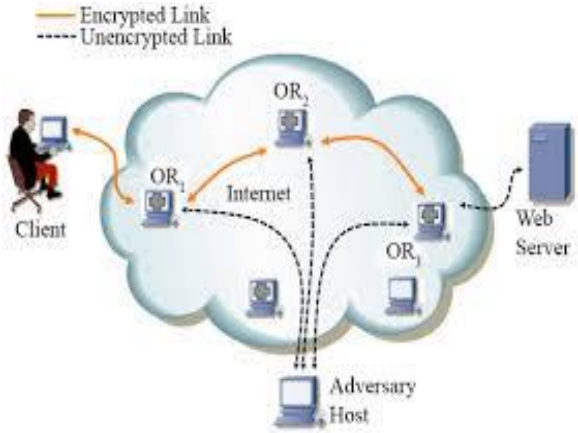

Fig. 2. TOR Network.

IV. IMPLEMENTATION

A systematic, iterative process that prioritised modularity, simplicity, and extensibility was used to implement the TCP-based tunnelling system using Python. Using low-level socket programming, the main objective was to create a lightweight tunnelling application that routes network traffic from a local endpoint to a distant server. The tunnelling system's implementation process is described in detail in this section, along with the main technical elements, programming choices, and solution behaviour at runtime.

The Python programming language was used to create the entire tunnelling mechanism because of its expressiveness, usability, and extensive standard library support for network operations. In particular, the threading module made it possible to implement concurrent data transmission, which is necessary for bidirectional tunnel behaviour, whereas the socket module was utilised for low- level network communications. To guarantee compatibility and access to networking tools, Python 3.10 was utilised in a Linux-based development environment.

The system is built using a dual-threaded client-server architecture. It is made up of three main parts:

- Accepting incoming client connections on a designated local IP address and port is the responsibility of the local listener (server socket).
- The remote connector, also known as a client socket, serves as the endpoint to which traffic is routed by connecting to the specified remote server and port.
- Data Forwarder Threads: Each connection generates two threads, one for relaying data from the local client to the distant server and another for relaying data back the other way.

The start tunnel function's initialization and binding of the server socket mark the beginning of the implementation process. In order to configure the socket for TCP communication over IPv4, a new socket object is created using socket. The bind () method is then used to bind this server socket to a specific port number (43986) and local IP address (192.168.120.130), thereby formulating it to receive incoming associates on that interface. After the binding, listen (10) is used to put the attendant into a listening state. This creates a passive socket with an accumulation queue that can process up to 10 assembly requests at once before rejecting more.[19]

When a local client accepts a connection, the server records the client's address and uses a different socket to start a fresh outgoing connection to the designated remote location. The remote host is set to rdap.arin.net, and the remote port is set to 43, which is the default port for RDAP and WHOIS queries. This remote connection is created using remote socket. connect ((remote host, remote port)). The local client and the connection to the remote server are represented by the two socket connections that are currently active. The TCP tunnelling mechanism is based on these two endpoints.

The implementation makes use of Python's threading module to allow full-duplex, bidirectional communication between the local client and the remote server. Each subway session starts two threads: one for client-to-server data accelerating and another for server-to-client data forwarding. Every filament runs the forward () function, which uses sendall(data) to send the data to the destination socket after performing an unceasing read operation with recv(4096) to receive the data from the source socket. As long as data is being received, this loop keeps going; when there is no more data available, it gracefully ends, signaling that the connection has been closed [Fig 3].

The tunnel can manage simultaneous data transmission without spoiling, thanks to the use of distinct threads for each communication direction. For real-time, low-latency applications where awareness and non-blocking behaviour are decisive, like proxy servers and secure communication relays, this design is indispensable.[20] For more extensive network applications, the overall architecture offers a dependable and simple method of tunnelling that can be expanded with security and scalability improvements.

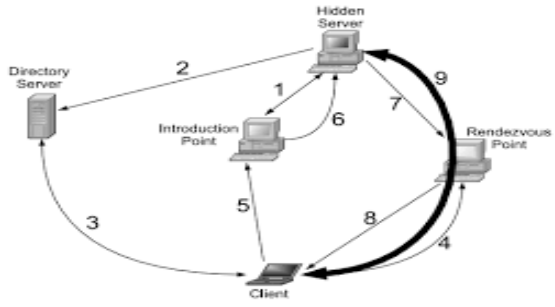

Fig. 3. Client - Server Connection.

V. RESULTS

A controlled network environment was used to successfully test and validate the implemented Python-based TCP tunnelling system. The main goal was to confirm that the tunnel could consistently receive incoming connections on a local IP and port, send the traffic to a distant server, and concurrently send back responses to the initial client

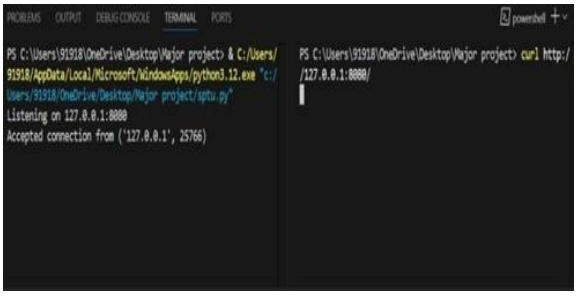

Fig. 4. Basic output

The tunnel server was initialized upon execution and started listening on port ex: 43986 and the designated local IP address, ex: 192.168.120.130[Fig.4]. The socket was correctly bound and waiting for incoming client connections, according to the terminal outputs. Accurate connection tracking was demonstrated by the server immediately logging the incoming connection details, such as the source IP and port, when a client established a connection to the tunnel. After creating a secondary socket, the tunnel was able to connect to the remote server, rdap.arin.net, on port 43, which is the standard port for WHOIS/RDAP queries. In order to maintain traceability and transparency throughout the process, this successful connection was also recorded. The dual-threaded mechanism, which handled data transmission in one direction (client to server and server to client), allowed for data forwarding. Valid responses to client queries sent to the remote server via the tunnel during testing were received at the client end without any data loss or corruption. This proved that the tunnel could accomplish full- duplex communication while preserving data integrity. Wireshark packet analysis verified that dormancy stayed within reasonable bounds for light communication tasks and that packets were conveyed through the tunnel as anticipated.

A crucial prerequisite for a transparent tunnelling solution is that the system efficiently routes TCP payloads between the two endpoints without changing or examining the packet contents. Additionally, non-blocking behaviour was guaranteed by using distinct threads for each direction of data flow, enabling smooth concurrent reads and writes. Under mild testing circumstances, the system handled several sequential connections without crashing or displaying memory leaks.

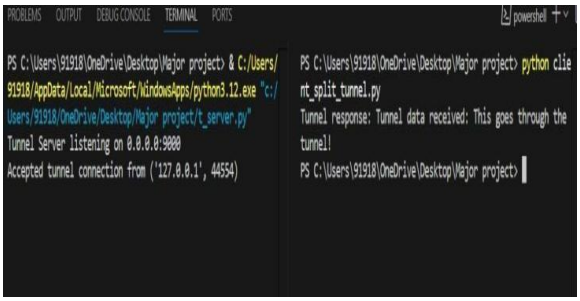

Fig. 5. Another Output

VPN Tunnel Metadata Leaks

While VPN tunnelling enhances privacy, metadata leaks expose deanonymization vectors:

- Throughput

$$\text{Throughput} = \frac{\text{RDP}}{\text{T}}$$

where RDP = received data packets, T = observation time.

Packet Loss

$$\text{LOSS} = \frac{SDP - RDP}{SDP}$$

where SDP = sent data packets.

Jitter (Delay Variation)

$$\text{J} = \frac{\sum |D_i - D_{\{ava\}}|}{N}$$

Observation: Experimental results show that DNS leaks, SIP headers in L2TP/IPSec, and timing anomalies in SSL tunnels can be matched with TOR flows to link entities. Observe Fig.5.

Testing revealed some limitations, despite the tunnel's consistent performance within its intended scope. Sensitive information cannot be sent over untrusted networks because the system does not currently have encryption. Furthermore, it lacks integrated error handling and connection timeout logic, and it does not support concurrent client handling beyond the stringing capabilities of the operating system. In spite of these drawbacks, the outcomes efficiently verified the fundamental tunnelling functionality, offering a strong basis for more complex implementations

combining load balancing, encryption (such as SSL/TLS), and authentication. In conclusion, the tunnel's objective of creating a simple, trust worthy, and transparent TCP communication channel between a local client and a distant server was proficient. In addition to confirming that the tunnel is operationally sound for prototype and academic purposes, the experiment offers a valuable foundation for future research into more secure and mountable tunnelling frameworks. And also, to find the difference between traditional tunnelling and split tunnelling observes Table 1.

Feature	Traditional Tunneling	Split Tunneling
Traffic Routing	All traffic goes through the tunnel	Only specified/sensitive traffic routed through tunnel; rest bypasses it
Security	Higher (all data protected)	Moderate (only tunneled traffic protected)
Bandwidth Usage	Higher (entire traffic flows through the tunnel)	Lower (only part of traffic flows through tunnel)
Implementation	Less complex; simpler rule set	More complex; must determine which traffic to split/bypass
Performance	Potentially slower (all traffic subject to tunnel overhead)	Faster for non-tunneled traffic; minimal overhead for direct traffic
Control	Simple, easy to manage; one route for everything	Granular; user/app/domain-based routing possible
Typical Use Cases	Full VPNs, Tor, secure remote work	Corporate VPNs, privacy tools, remote work needing local internet access
Risk of Data Leakage	Low (all data encrypted/tunneled)	Higher (non-tunneled traffic may leak sensitive data if not configured correctly)
Resource Efficiency	Less efficient for mixed traffic	More efficient for mixed traffic (e.g., stream videos locally, tunnel only work traffic

Table.1: Comparison between Traditional tunnelling and Split tunnelling.

VI. CONCLUSION

Specifically, through socket programming and port forwarding, the effective implementation of the Python-based tunnelling mechanism shows a practical and basic approach to grasp low-level network communications. This project's developed code lets a TCP tunnel efficiently listen on a designated local IP and port and forward traffic to a specified remote host and port. In this case, the remote server rdap.arin.net listening on port 43 was used as a test destination, which conforms to the standard protocol for WHOIS lookups and acts as a pragmatic endpoint for confirming the tunnel's operational state.

This project's central goal was to enable bidirectional communication between a local client and a remote server by means of a lightweight TCP-based tunnel built on Python's socket and threading modules. The code does this by building a listener socket on the local machine, picking up incoming connections, and then forging a fresh connection to the far- off server. By utilizing two separate threads each responsible for forwarding data in one direction (from client to server and vice versa) the tunnel achieves full-duplex communication. This design reflects the basic structure of proxy services and tunnelling protocols, laying the groundwork for more complex and secure systems such as SSH tunnelling, VPN architectures, and traffic obfuscation frameworks used in anonymity networks.

It raised awareness of a number of crucial aspects of privacy and network security. First, it emphasizes how easily a basic tunnelling service can be constructed with general- purpose programming tools, highlighting the importance of monitoring unusual port forwarding activity within public or corporate networks and securing open ports. Second, if a tunnel is intentionally attacked or compromised, the construction reveals potential flaws in how traffic could be rerouted or intercepted. This is particularly relevant in the context of anonymity networks such as Tor, where comparable techniques may be used to protect or compromise user privacy.

A task also emphasises the necessity of extra features in practical applications, like authentication procedures, logging for auditability, encryption (TLS/SSL), and exception handling for resilience. Although the current implementation works, its use in production-level systems is limited because it lacks security hardening and relies on a trusted environment. However, it offers a useful proof-of-concept that can be expanded into more complex tunnelling solutions that have features like access control, encrypted tunnels, tunnel activity logging, and web-based interface integration.

In summary, this project not only achieves its technical objectives by creating a working TCP tunnel in Python, but it also functions as a teaching tool for investigating basic networking concepts and the wider ramifications of tunnelling technologies in relation to cybersecurity risks and secure communications. This fundamental framework should be improved in future work by adding dynamic configuration handling, encryption standards, and support for concurrent client sessions, making it a more robust and scalable tunnelling framework appropriate for practical applications.

REFERENCES

[1] Sourav, "Data security and privacy concern in the healthcare system," in Internet of Healthcare Things. Hoboken, NJ, USA: Wiley, 2022.

[2] J. Bailey, "Communication: Telephone, computers and WWW," in Inventive Geniuses Who Changed the World. Cham, Switzerland: Springer, 2022.

[3] Administrative Office of the U.S. Courts, "Annual report," 2024.

[4] P. Tippe, "A study of deanonymization attacks of onion services," in Proc. Sicherheit, Gesellschaft für Informatik, 2024.

[5] T. H. Lenhard, "Telephone systems," in Data Security. Wiesbaden, Germany: Springer, 2022.

[6] Tor Project, "Operational security," 2023. [Online]. Available: https://community.torproject.org/onionservices/advanced/opsec/

[7] Tor Project, "Tor rendezvous specification – version 3," 2023.

[8] S. Budi Yanto and I. Pratama, "Classification of network status in academic information systems using naive Bayes algorithm method," in Proc. 2nd Int. Conf. Broadband Commun., 2020.

[9] Google, "Google Scholar," 2022. [Online]. Available: https://scholar.google.com

[10] Panchenko, "Analysis of fingerprinting techniques for Tor hidden services," 2017.

[11] M. Juárez Miro, "Fingerprinting hidden service circuits from a Tor middle relay," 2017.

[12] "The lifecycle of a new relay," Sep. 2013. [Online].

[13] P. Laperdrix, "Browser fingerprinting: An introduction and the challenges ahead," Sep. 2019. [Online].

[14] S. Nepal, "Deanonymizing schemes of hidden services in the Tor network: A survey," 2015.

[15] M. Bernaschi, "Exploring and analyzing the Tor hidden services graph," 2017.

[16] Y. Qin, "Tracing Tor hidden service through protocol characteristics," 2022.

[17] G. B. Regulwar, "Content analysis and visualization of privacy policy using privacy management," 2024.

[18] K. Krippendorff, Content Analysis: An Introduction to Its Methodology. Thousand Oaks, CA, USA: SAGE Publications, Inc., 2019. doi: 10.4135/9781071878781.

[19] M. Al-Naday, "Service-based, multi-provider, fog ecosystem with joint optimization of request mapping and response routing," May/Jun. 2023.

[20] J. Nam and V. Yegneswaran, "Secure inter-container communications using XDP/eBPF," Apr. 2023.