SNITCH: Leveraging IP Geolocation for Active VPN Detection

Tomer Schwartz
Data and Security Laboratory
Fujitsu Research of Europe Ltd.
Tel Aviv-Yaffo, Israel
Email: tomer.scwhartz@fujitsu.com

Ofir Manor
Data and Security Laboratory
Fujitsu Research of Europe Ltd.
Tel Aviv-Yaffo, Israel
Email: ofir.manor@fujitsu.com

Andikan Otung
Data and Security Laboratory
Fujitsu Research of Europe Ltd.
Slough, United Kingdom
Email: andikan.otung@fujitsu.com

***Abstract*—Cyber attacks and fraud pose significant risks to online platforms, with malicious actors who often employ VPN servers to conceal their identities and bypass geolocation-based security measures. Current passive VPN detection methods identify VPN connections with more than 95% accuracy, but depend on prior knowledge, such as known VPN to IP mappings and predefined communication patterns. This makes them ineffective against sophisticated attackers who leverage compromised machines as VPN servers. On the other hand, current active detection methods are effective in detecting proxy usage but are mostly ineffective in VPN detection.**

This paper introduces SNITCH (Server-side Non-intrusive Identification of Tunneled CHaracteristics), a novel approach designed to enhance web security by identifying VPN use without prior data collection on known VPN servers or utilizing intrusive client-side software. SNITCH combines IP geolocation, ground-truth landmarks, and communication delay measurements to detect VPN activity in real time and seamlessly integrates into the authentication process, preserving user experience while enhancing security. We measured 130 thousand connections from over 24 thousand globally distributed VPN servers and client nodes to validate the feasibility of our solution in the real world. Our experiments revealed that in scenarios where the State of the Art fails to detect, SNITCH achieves a detection accuracy of up to 93%, depending on the geographical region.

I. Introduction

Virtual Private Network (VPN) servers run software that forwards network traffic through an encrypted tunnel while obfuscating the source of the communication. They have multiple uses, such as enabling secure remote access to private networks, source IP spoofing (which enables geolocation spoofing), and providing an additional layer of encryption for users in insecure networks [1] [2]. Consequently, not all VPN usage is malicious, but through the use of IP spoofing, malicious actors can hide their identity and avoid detection. Specifically, malicious actors can use privately owned or hacked machines or routers as proxy/VPN servers [3] [4] to avoid detection mechanisms, particularly those used by state-of-the-art (SOTA) passive detection systems. As an example, state-level malicious actors have been seen to use hacked devices as proxies in multiple campaigns [5] [6], ranging from intrusion vectors to botnets used in DDoS attacks. Detection of these threats is crucial for web service security.

VPN detection is an important security mechanism used to detect bots, scrapers, and other malicious actors, but it is especially important for services that use IP geolocation as a security factor, such as zero-trust infrastructure and fraud prevention services [7] [8]. The global Zero Trust security market was valued at 26.45 billion dollars and is projected to reach 162.9 billion dollars by 2032 [9], similarly, the size of the fraud detection and prevention market was valued at 36.89 billion dollars in 2022 and is projected to grow to 182.66 billion dollars by 2030 [10].

Passive VPN detection methods are the current state of the art and can detect VPN usage with an accuracy of up to 95-99% [11] [12] [13]. However, passive detection methods require prior knowledge of VPN to IP address mappings and will fail to detect new private VPNs until updated (the limitations are discussed in Section II-B2). Furthermore, multiple server-side active detection methods have been tried over the years [14], but they have been circumvented by VPN services, which are actively trying to avoid detection.

To address this, we propose SNITCH, a server-side VPN detection solution/framework with real-time delay measurement of client and (approximate) VPN communication. Our approach to VPN detection uses direct delay measurements of client-side communication, IP geolocation, and the utilization of trusted landmarks. Our method does not require prior known-VPN knowledge for detection and communicates with trusted landmarks in proximity to the perceived location of the client, thus avoiding the need to communicate with the VPN server and overcoming the limitations of previous solutions. We have shown the viability of SNITCH through extensive experimentation on leading VPN services, achieving up to 93% detection accuracy in regions with modern network infrastructure. It can be seamlessly integrated into the authentication process, enabling web services to detect new or private VPN servers in cases where existing solutions are unable to do so due to the aforementioned reasons.

Our paper is structured as follows. Section II presents the background of IP geolocation, VPN technologies, and known detection methods with their limitations. Section III

Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb) 2025
28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-9-0
https://dx.doi.org/10.14722/madweb.2025.23008
www.ndss-symposium.org

describes the detection method in detail and discusses the reasoning behind the methods used. Section IV describes the components and structure of the experiment. Section V provides the obtained results. Section VI discusses the efficacy of the proposed method, as well as possible future work to address the limitations. Section VII concludes the article.

II. Background

In this section, we discuss IP geolocation methods and their security use cases. Followed by existing VPN detection methods and their limitations.

A. IP Geolocation

IP Geolocation [15] is the process of locating a device based on information obtained from its IP address. It is used in Internet security both to profile potential targets and as a security measure, especially in the fraud prevention and Zero Trust markets [8] [16]. IP geolocation techniques can be categorized as: *passive*, *active*, and *hybrid*.

- **Passive techniques** utilize data analyzable purely from the IP address without direct device communication. Various passive methods demonstrate notable results. [15] [17] [18] [19]
- **Active techniques** typically employ probes to ascertain the delay to the target, which, combined with delay-to-distance models, assists in estimating the target device's location. [20] [21]
- **Hybrid techniques** integrate two methodologies, utilizing prior data and machine learning algorithms to enhance active detection results. [17] [22] [23]

1) Technology applications: Zero Trust security models rely on IP geolocation as an additional security factor. Specifically, Multi-Factor Authentication (MFA) providers utilize IP Geolocation to permit and deny access to certain users based on their location or implement further security restrictions when a user attempts to log in from an anomalous location. Microsoft Azure's Conditional Access policy [24] allows organizations to block users from accessing the services from countries and regions that the organization never operates from. Furthermore, the proliferation of national and multinational legislatures on data and user protection (e.x. GDPR) requires certain data to remain within certain borders. [25]

2) IP Geolocation Limitations: Passive geolocation methods require extensive ground truth data to maintain adequate accuracy [17]. In contrast, active methods need much less data but are more vulnerable to network failures, which can lead to major inaccuracies [20]. The essential premise of IP Geolocation is that the IP address presented by the target device is the one allocated to the user's device or network. The use of VPNs and proxies to hide the client's IP renders geolocation security ineffective. This limitation applies to all IP Geolocation methods, as both passive and active methods depend on this core assumption. To address this, some IP geolocation services employ server-side VPN/proxy detection as a countermeasure [15].

B. VPN Detection

1) VPNs vs Proxies: Virtual Private Networks (VPNs) enable secure connections across public networks [26]. This security is achieved through the encapsulation and encryption of outbound packet data from the client, which is subsequently unpacked at the VPN server and forwarded as though originating from the server itself. Upon receiving a response, it is encapsulated, encrypted, and retransmitted to the client. This technique conceals the client's original IP address from the accessed service or facilitates access to internal networks, simultaneously encrypting sensitive data across public networks. Separately, HTTPS proxies [14] establish secure connections on the transport layer. When a client connects to a web server through a proxy, all HTTPS requests pass through the proxy server, obfuscating the client's IP address. Client requests are transmitted to the proxy over TCP, necessitating distinct TCP handshakes for each session by both the client and the web server, effectively "breaking" the TCP connection at the proxy. On the other hand, VPNs function at the network layer, maintaining a continuous TCP connection with the destination server. Figure 1 illustrates the differences in handshake protocols between direct connections, proxies, and VPNs.

The core VPN server implementations, such as OpenVPN [27], used by PIA [28], encapsulate and transport all client-generated packets through a tunnel without partitioning server-bound handshakes. The established tunnel explains the inability of the previous solutions to detect VPN usage. Certain VPN services function similarly to proxies, where TCP handshakes are broken, but TLS handshakes are tunneled. Our measurements have observed this technique in NordVPN [29] and ExpressVPN [30]. In contrast, VPNs with SSL decryption capabilities (typically used in corporate settings, such as Palo Alto's GlobalProtect) interrupt the TLS handshake, resulting in a TLS round-trip time (RTT) shorter than TCP RTT. VPNs incorporating these features are identifiable through standard proxy detection methodologies, such as BADPASS [31] which measures the difference between the TCP and TLS handshake.

2) Detection Methods: There are multiple solutions for Server-side Proxy/VPN detection, each with their own strengths and weaknesses. For the purpose of our paper, we will discuss only detection methods that can be implemented from the server side. The service being accessed is doing the detection and only has access to the communication from the client **after** it has passed through the VPN / proxy. These solutions can be divided into two main categories. The first is the passive approach, which is commonly based on known IP databases (DBs). DBs can include behavioral analysis data and reputation scores based on reports and passive data collection. Passive data can include raw packet capture at key junctures, offline port scanning, source analysis, etc. The main players include Maxmind, SEON, IPQualityScore, etc. [11] [12] [13].

The second approach is the active detection approach, which has multiple techniques.

- **Active port and service scanning:** common VPN solu-

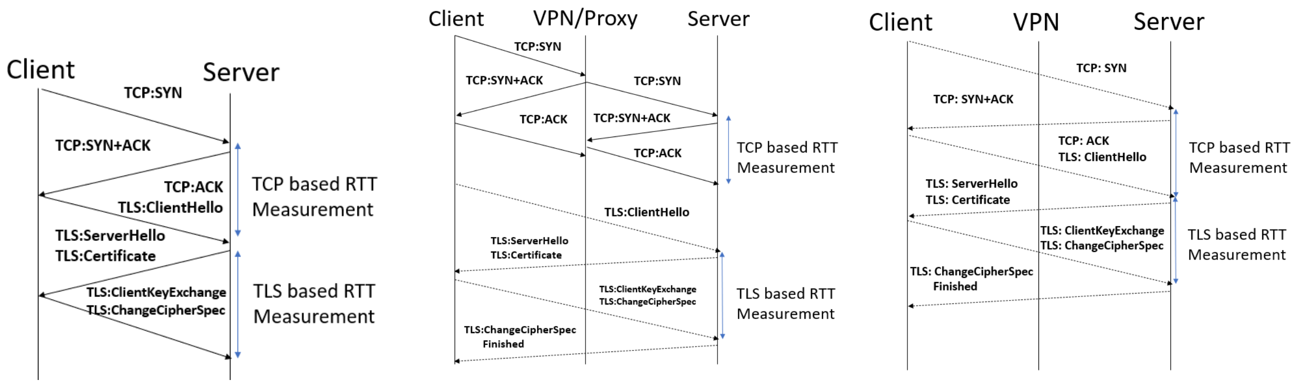


Figure 1. Initial handshake between client and web server with either Direct,Proxy, or VPN communication

tions/services have default ports, protocols, and patterns that can be scanned to determine if a VPN service is running on a server using a method called fingerprinting. When a connection is initiated on the server, ideally a scan of the source IP can determine if the client is using a VPN. New fingerprinting methods are constantly developed [32] in a constant game of cat and mouse between VPN and detection services. Fingerprinting methods have been shown to not be robust because the ports and protocols used can be easily switched with non-default configurations and cause false negatives. [14] However, more sophisticated methods such as VPNChecker [33] that use a graphical representation of the ports probed, topological analysis, and communication information, have shown impressive results. VPNChecker achieved an accuracy of up to 89.3%, but requires probing access to the VPN server (which is not a given) and node connection information, which are significant limitations.

- **Deep Packet Inspection:** VPN usage causes full encapsulation of the original packet; this fact can change the available size for data in the packet and cause a few detectable changes in the TCP headers. Specifically, the MTU/MSS fields in the header can shift from the norm. This limitation also causes more variance in the data size and other characteristics of the received packets over time. Deep packet inspection can be implemented on the server using any sniffing tool. However, there are multiple issues with this approach. The TCP field anomalies can be (and have been) solved with simple configuration changes. The packet content anomalies are not consistent and require a significant amount of packets and complex machine learning models to identify with sufficient accuracy. Prominent examples of such models are “Detection of VPN Network Traffic” [34] which achieved an accuracy of 93.8%, and “Detection of Virtual Private Network Traffic using Machine Learning” [35] which achieved an accuracy of up to 97.82% on its dataset. It is important to note that most methods using packet inspection and machine learning try to identify the encrypted communication between the client and the VPN server, and are therefore not analyzed here.
- **Delay and RTT measurement:** Delay anomaly measurements have been implemented in multiple papers to detect proxies, using RTT measurement of the TCP handshake [36] compared to either the TLS Handshake [37] or application layer communication, [14] [38]. BADPASS, by Elisa Chiapponi et al. [31] specifically achieves excellent results for proxy detection (and is utilized in our solution as a first step to detect proxy usage). This technique has proven effective at detecting proxies, but has failed to consistently detect VPNs. This is caused by the fact that in a VPN service without extensions (as discussed previously), the entire communication is encapsulated, and does not “break” at the VPN server, and thus does not have significant differences in the RTT measurement of different protocols or applications. A newer technique called Calculatency [39] by Reethika Ramesh et al. integrates both ICMP ping requests and 0trace measurements to achieve impressive detection results for VPN connections as well as proxy usage. However, a significant limitation in their detection method is the requirement for the VPN server to respond to ICMP pings or send out an ICMP TTL exceeded response, which are both easily blocked by the service. Furthermore, machine learning-based methods [40] require significant initial training, data collection, and a significant amount of data to generate a detection. Other more intrusive methods can include running JavaScript code that generates new connections to the server (or multiple servers).

For the purpose of using VPN detection as a security factor, a solution is required that could be incorporated into the authentication process. Practically, the user experience should not be (significantly) affected via loading times or false positive errors. Most importantly, security standards must be maintained, which requires very high accuracy as well as fast detection speeds. The issue with the passive detection approach is that even though it is extremely fast and accurate, it requires prior knowledge. Meaning that it is very good at detecting known, well-established VPN services (which will tend to have a benign use), but when encountering new non-mainstream VPN servers (higher risk of malicious activity), the passive approach will essentially always get a false negative, which is unacceptable from a security perspective. Port scanning is unreliable and easily circumvented, as we discussed. Deep packet inspection paired with machine learning approaches shows promise, but thus far have required a lot of communication flow data to be effective, which we do not have access to in a log-in process. Based on these known limitations, we decided to take a deeper look at delay measurement. In the following section, we will go over the method itself, with a focus on the thought process and assumptions it is based on.

III. Method Description

Our detection method utilizes geolocated landmarks to generate approximate RTT measurements to the VPN. In the following subsections, we will explain the detection method in depth as well as the underlying assumptions.

A. Delay vs Geolocation

When looking at a client's communication to a web server (which we will call the server for this explanation) through a VPN or a proxy, we can divide the communication into 3 sections (as illustrated in Figure 2):

CS_{RTT} is the full RTT (round trip time) from the client to the server.

CV_{RTT} is the RTT from the client to the VPN.

VS_{RTT} is the RTT from the VPN to the server, where essentially $CS_{RTT} = CV_{RTT} + VS_{RTT}$.

From the servers' perspective, VS_{RTT} is the RTT measured from the server to the clients' perceived IP, which is the VPN servers' IP when one is used and the clients' IP otherwise. When no VPN is used $CV_{RTT} = 0$ and in an attempt to measure VS_{RTT} we should get approximately the same result as CS_{RTT}. As a server, we have no access to CV_{RTT} and in the detection process we essentially attempt to prove that it has a nonzero value. We can do this with measurements of CS_{RTT} and CV_{RTT}.

To measure CS_{RTT}, the TCP and TLS handshake RTTs are measured. We then use the established BADPASS [31] method to check if there is a significant difference between the TCP and TLS RTTs and detect proxy usage. The novel aspect of our method lies in the measurement of VS_{RTT} when no such difference is present. The naive approach would be to simply ping the VPN. However, VPNs try to hide or obfuscate their presence as much as possible. There are multiple ways in which this is done, but a simple method is that the VPN servers just ignore all unrecognized incoming communication.

To solve the issue of intentional obfuscation by the VPN, we implemented a method to measure the expected delay difference indirectly using trusted probes and landmarks. Our baseline assumption is that with modern network infrastructure and endpoint hardware, the network delay is mostly dictated by the physical distance between the communicating network endpoints (in our case, these are the client, VPN server and web server). When measuring a significant difference in the RTT between CS_{RTT} and VS_{RTT}, we are essentially detecting that the client is farther away from the server than is indicated by its IP address (IP Geolocation spoofing). Therefore, if this assumption holds, the measurement of the delay between landmarks geographically close to the VPN and the detection server will produce very similar results to the expected VS_{RTT} value. This assumption fails in unstable connections, which can be detected using standard deviation analysis and other relevant metrics. With current passive geolocation technologies, we can expect city-level accuracy for most urban areas and country / region-level accuracy for the rest [41] [42]. Specifically, for the IP geolocation databases analyzed by Callejo et al. [41], more than 80% of the IPs had a geolocation error of 100 km or less. Using passive IP geolocation on the perceived clients' IP will return a good approximation of either the clients' or VPNs' geolocation. An error of 100-200km is negligible in its effect on delay measurement on an international scale, with an expected propagation speed between half and 2/3 the speed of light over short distances (up to 100 km per ms RTT) [43][1]. With the returned geolocation, we can use a ground truth service to find geographically close landmarks. The trusted landmarks can be pinged for direct RTT measurement, which will generate a very good approximation of VS_{RTT}.

The difference between VS_{RTT} and CS_{RTT} proves that some sort of IP geolocation spoofing is taking place, but remember that our method activates only when no proxy is detected, meaning that a VPN or another tunneling protocol is used. With that solution in hand, we can now build a full VPN detection solution.

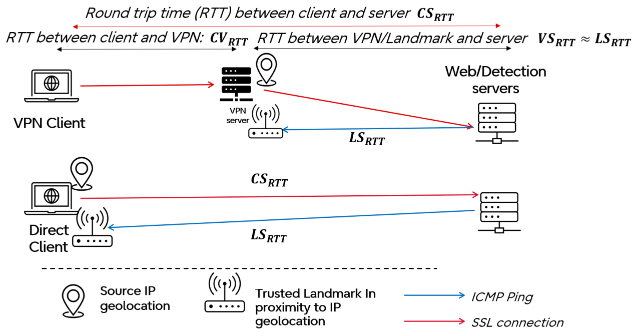

Round trip time (RTT) between client and server CS_{RTT}
RTT between client and VPN: CV_{RTT}
RTT between VPN/Landmark and server $VS_{RTT} \approx LS_{RTT}$
VPN Client
VPN server
Web/Detection servers
LS_{RTT}
CS_{RTT}
Direct Client
LS_{RTT}
Source IP geolocation
Trusted Landmark In proximity to IP geolocation
ICMP Ping
SSL connection

Figure 2. Direct vs VPN landmark connection RTT representation

B. The Detection Method

Our solution runs a detection server alongside any running service and requires permissions to run a sniffing tool. The steps are as follows:

1) The client initializes communication with the web server. That initial communication will be analyzed to calculate the full RTT from the client to the server

[1]Ideally, we would want to use a geolocation service that is as accurate as possible, which would usually be an active geolocation service, but the system has to be non-disruptive for the client, which requires very fast geolocation speeds, so a balance must be reached.

(CS_{RTT}). The RTT is measured using TCP and TLS handshakes (as described in Section IV-B).
2) If the connection has a high jitter and is considered unstable.[2] return an unstable connection error.
3) Using the BADPASS [31] algorithm, if a significant difference is measured between the minimal TLS and TCP RTTs, a proxy is detected. if no significant difference is detected, continue.
4) The client IP is geolocated and nearby landmarks are located in a specified radius. The 3 geographically closest responding landmarks are chosen. If no landmarks are found in that radius, return an error.
5) The RTT to the chosen landmarks is measured using multiple pings and aggregated to generate a good estimation of VS_{RTT}. These will be called the SNITCH landmarks going forward
6) A determination is reached using the measured RTTs with error margins calculated based on the measured CS_{RTT} value.

Both the TCP and TLS handshake RTTs are measured multiple times per client. Multiple requests are triggered when accessing the web server from base image/javascript tags that are expected parts of any website, with the additional assumption that the requested artifacts are uncached. If needed, more requests can be triggered with additional tags (img/javascript/iframe, etc.).

Detection Parameters and Equation:

- CS_{RTT} is the the minimal RTT between the perceived client and our server (units: [ms])
- LS_{RTT} is the minimal RTT from the server to the geolocated landmarks (units: [ms])
- C_{EM} is a relative error margin of detection based on CS_{RTT}. The specific function is specified in subsection IV-C
- LS_{STD} is the standard deviation of the RTT measured between the chosen Landmarks and the server. (units: [ms])
- D_{CL} is the median distance between the geolocated IP and the landmarks (units: [km])
- GE is the expected error in the IP Geolocation service we are using (units: [km])
- ω is the expected propagation speed of communication through the network (units: [km/ms])

Based on these, we conclude the presence of a VPN when the following condition is satisfied:

$$CS_{RTT} > LS_{RTT} \cdot (1 + C_{EM}) + LS_{STD} + \frac{D_{CL} + GE}{\omega} \quad (1)$$

The detection criterion is that the RTT between the client and the server (CS_{RTT}) is longer than the RTT measured from the landmark by an error margin, which represents the expected jitter of the communication and an additional buffer based on the expected geolocation error and the distance of the landmark from the VPN server.

[2]which we measure using an RTT standard deviation of more than 40 ms, derived from acceptable ranges of jitter [44]

To contend with unstable connections, we added the standard deviation of the measured LS_{RTT} to the detection equation. Connections with increased jitter (represented with the STD) are less likely to achieve the true minimum RTT during the connection process.

The detection confidence value is the difference between the RTTs and the buffers divided by the expected jitter buffer, with an upper bound of 1 and a lower bound of -1. The confidence is calculated as:

$$\frac{CS_{RTT} - LS_{RTT} \cdot (1 + C_{EM}) - LS_{STD} - (D_{CL} + GE)/\omega}{LS_{RTT} \cdot C_{EM}} \quad (2)$$

After presenting the detection process in depth. In the following section, we will present the experimental setup, process, and limitations.

IV. Experiment

A. Experimental setting

Through the use of VPN services, one gets access to tens of thousands of unique VPN servers in diversified locations. However, no comparable service exists for direct clients, making it challenging to access thousands of globally distributed to test our solution. To overcome this limitation, we decided to use the residential IP proxy service Bright Data [45]. As discussed previously, we know that the TCP connection is "broken" when a proxy is used. Therefore, we can use the TCP RTT of proxied TCP communication as a direct RTT measurement from the proxy, which we treat as the direct client. An in-depth explanation is given in subsection IV-A2. The efficacy of proxy detection has been shown in previous work. Therefore, for our experiment, we used VPN services that generate a full tunnel and only used the TCP RTT for detection, bypassing proxy detection entirely.

1) Clients and servers: we set up 8 servers and 8 clients using Azure virtual machines in diversified locations. 2 in North America, 3 in Europe, 2 in South East Asia, and 1 in the Middle East (both for clients and servers).

Our detection server includes a simple flask web server with an additional packet sniffing script (described in Section IV-B) that communicates with the detection server. Both are implemented using Python3. Our detection method can run separately from any service running on the server, but will need to run alongside it with packet sniffing privileges.

To accommodate connections to VPN and proxy services from remote machines, as well as to maximize the authenticity of the connection, we created an automation script using Selenium [46] with a Chrome driver and the VPN services Linux cli applications and proxy settings. Every few seconds, the client script connects to a random VPN server from our chosen VPN services or a random residential proxy and accesses our detection servers. The client sends the original IP of the client, as well as the service and protocol as metadata. This information is used for validation of the detection results and further analysis.

2) VPN and Proxy Services: To prove the efficacy of our solution, we tested our detection method on two well-known VPN services that use a full tunnel (PIA and PureVPN) [28] [47]. Each service can be configured to use a subset of modern VPN protocols. Those are OpenVPN [27], IKEv2/IPSec [48], and WireGuard [49]. We tested our solution with each service and each available protocol.

We used bright data residential proxies [45] as "direct" clients. Bright Data's residential network architecture, which was analyzed in the paper by Xianghang et al. [50][3], includes a proxy gateway, which routes client requests using multiple hidden gateways to residential hosts running proxy software. The residential hosts forward the client request to the destination server (with the residential hosts ip visible to the server) and send the answer back through the proxy gateway to the client.

The SSL connections originate from the client and pass through the entire infrastructure, which could add very significant delays to the connection, but the TCP connection breaks at the residential host as it would with any other HTTPS proxy and can therefore be considered a direct connection that will be unaffected by the prior residential proxy infrastructure. Specifically, we will use the TCP handshake RTT from the residential proxy as the RTT measurement from client to server CS_{RTT} and ignore the SSL RTT measurement in this experiment.

Our connection configuration used HTTPS proxies with randomized geolocation targeting to ensure internationally diverse direct client locations.

3) IP Geolocation and Ground Truth Landmarks: The geolocation service that we used for the experiment is the ipinfo geolocation service [52]. It is a passive geolocation service with an API that is easy to integrate. There are other IP geolocation services that might have better results, but we made a choice of convenience for the experiment. Based on past measurements [41] that showed that over 80% of the analyzd IPs have an error of less than 100[km], we set an expected geolocation error (GE) of 100 km.

We chose to use the RIPE Atlas API [53] to access trustworthy geolocated probes. Probe information around a geolocation can be queried using the API. Based on the information provided, we find the closest available probes.

The main cause of RTT is the distance traveled, meaning that the further away a landmark is from the VPN server or the direct client, the less representative it is of the actual machines' RTT, which is what we are trying to approximate and will therefore reduce detection accuracy.

For this experiment, we chose a radius of 200 km to balance between accommodating the coverage limitations of RIPE ATLAS, having a wide margin from the expected city-level accuracy for most IPs worldwide [41], and keeping the RTT approximation valid.

The RTT from the server to the 3 closest available landmarks is measured with direct ICMP ping, and the minimal RTT is chosen because of the stability of direct connections and to filter out slow landmarks.

[3]Bright Data was called Luminati previously and was renamed [51]

B. Client RTT Measurement

The sniffing script measures both the TCP and TLS handshakes, in a similar fashion to the measurement method used in previous delay-based proxy detection systems. First, the TCP RTT is measured using the TCP handshake. The interval between the server sending the SYN-ACK packet and the received client ACK packet is measured. The TLS handshake is then used for an additional RTT measurement. The interval between the SERVER HELLO packet sent from the server to the next TLS packet received from the client, which includes either the CHANGE_CIPHERSPEC or TLS APPDATA FLAG depending on the TLS version [37], these are visualized in figure 1. To mitigate errors/noise, 7 TCP and TLS RTTs are measured from each client and sent to the detection server. The number of handshakes measured can be changed depending on the use case; increasing the number of measured handshakes would improve error mitigation but would slow detection times and could affect the user experience.

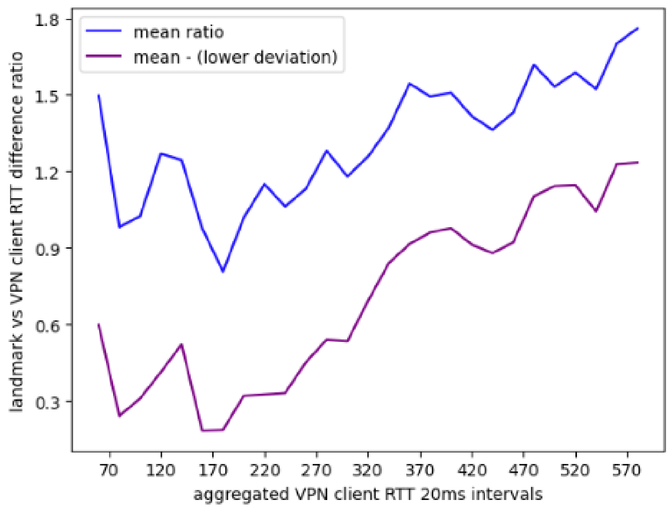
This measurement method works on any successful TLS authentication and exceptions for edge cases can be easily implemented, just through control of the server side. Therefore, control of the client that does not correlate directly with the real-world use case will not cause undue bias.

C. Error margin determination

To develop the error margin, we analyzed the relative difference of the measured RTT between the client and the landmark communication for both direct and VPN communications. It was clear that there is no constant difference between the RTT point of reference and the client RTT, and a relative margin approach was more likely to generate accurate detections. We used positive and negative deviation analysis to find the accurate margin, which is presented in figure 3. We determined that the positive and negative deviation should be calculated separately because RTT has a strict lower bound (limited by the maximal network propagation speed) and network delay can cause very significant positive deviations. Our expectation was that there would be a definitive margin between the negative deviation of VPN communication and the positive deviation of direct communication, but that was not the case. The positive and negative deviations overlap in multiple areas. The cause of this incongruence with our hypothesis will be discussed in later sections.

To maintain security standards, it is more important to limit false negatives than false positives (while keeping the false positive rate at a reasonable rate). Therefore, we fit the error margin to the positive deviation of the direct client RTT. The deviation does not have a consistent trend, so we fit a third-order polynomial to avoid overfitting. The resulting error margin is (where $CS_{RTT} = x$):

$$\mathrm{C}_{EM} = -1.972 \cdot 10^{-8}x^3 + 1.939 \cdot 10^{-5}x^2 - 4.952 \cdot 10^{-3}x + 0.620 \quad (3)$$



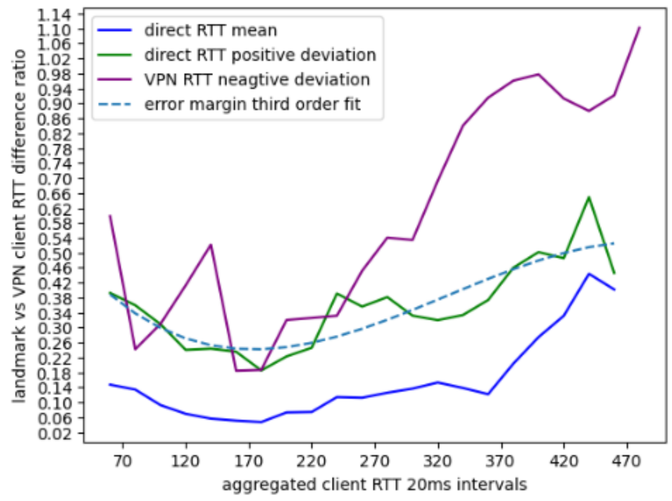


Figure 3. $\frac{CS_{RTT}-LS_{RTT}}{LS_{RTT}}$ ratio between VPN clients, direct clients, and SNITCH landmarks communication, aggregated over 20ms intervals for deviation analysis and clarity

D. Data collection

Each VPN client connects in varying intervals to a random configuration of service, protocol, and available region. We ran multiple iterations of the experiment from 10/09/2024 to 15/10/2024. Over the course of our experiment, we have analyzed over 130k connections with over 24k unique IPs and corresponding locations. The data was collected in a centralized Azure SQL server. Each detection gets a unique record in a detection information table and a client connection information table, which are then joined for analysis (the detection mechanism has no access to client data). Each connection from our clients includes unique fields and only those are added to the client database, joining the tables allows us to filter out detections that were not triggered by our clients. The sniffing tool restarts every 15 minutes, and during that time frame limits each unique IP to one detection. (After acceptance, we will make the data and code publicly available.)

E. Experiment Limitations

RIPE Atlas probe reliability and coverage: Atlas probes are not under our direct control and their setup conditions vary. Therefore, they are not necessarily reliable and might have a significant delay in relation to the expected communication delay of their area. Specific regions or probes that have significant deviations in latency can be detected and addressed over time. However, it is difficult to determine whether a significant deviation is representative of the region or a malfunction of the probe.

Another major limitation is that RIPE Atlas Coverage varies per region and a significant amount of direct clients were in locations that did not have a landmark in a radius of 200 km and therefore could not be detected. We measured 42k unique IPs for direct connections but could only generate detections for 17.3k of them. That is not the case for VPN connections, almost all VPNs had landmarks co-located with them. It seems that almost all data centers have RIPE ATLAS probes and tend to be in business networks. Most significantly, almost half of the connected IPs could not be detected properly due to the lack of coverage. This will be addressed in the discussion and must be a focus of future work.

Geolocation reliability/spoofing: Our method relies on the speed and precision of modern passive IP geolocation technology. For most regions, the expected accuracy is city-level [41] [42], with geolocation errors below 100 km for most IPs (80% in the analyzed databases), but remote regions can have a more significant error with an expected accuracy of region-level or even country level. In the range of a 200 km error the RTT difference between two machines should be insignificant on an international scale, with the expected propagation. However, in cases of more severe errors or even spoofing detection, the results could be severely affected, causing false positives or false negatives depending on the error.

Skewed Dataset: The dataset is slightly skewed due to the probe coverage limitation which affected direct connections more than VPN connections.

Possible future work to address these limitations is discussed in Section VI

Following the experiment setup, in the next section we will present the results. Initially, we will substantiate our assumptions on the basis of our detection method, followed by the detection results performance metrics analysis.

V. Results

A useful correlation to validate our initial assumption is the distance traveled versus the measured client-server RTT visualized in Figure 4. The correlation has a good linear fit (R^2 of 0.95); this result validates our initial assumption that the total distance traveled between the endpoints is the most significant contributor to the RTT of the communication. However, it is worth noting that the standard deviation is quite large.

Based on the same assumption, we expect a significant difference in measured RTT between the client and landmark

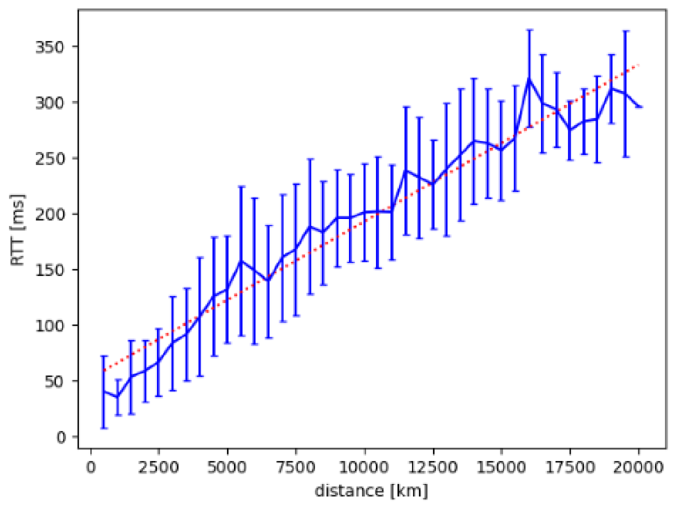


Figure 4. Aggregated endpoint distance (server to VPN to client) vs measured server to client RTT (CS_{RTT}) with standard deviation errors

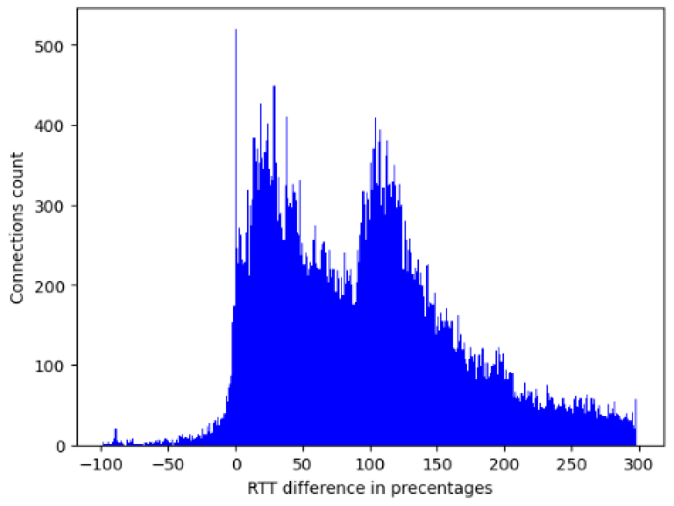


Figure 5. CS_{RTT} vs LS_{RTT} RTT difference histogram for VPN communication

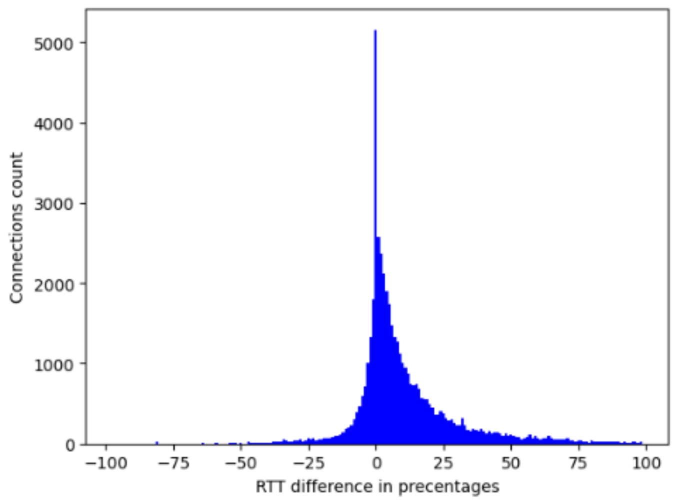


Figure 6. CS_{RTT} vs LS_{RTT} RTT difference histogram for direct client communication

when a VPN is used, and a minimal difference for direct connections. A histogram of the CS_{RTT} versus LS_{RTT} differences of VPN connections is visualized in Figure 5 and the direct connections are shown in Figure 6. The relative difference ratio of RTT is calculated simply as $\frac{CS_{RTT}-LS_{RTT}}{LS_{RTT}}$. A significant difference between VPN and direct connections can be seen. The direct connection distribution is somewhat normal, but the VPN distribution definitely isn't normally distributed, so we used a Mann-Whitney U test to analyze the difference. The results of the U test were a U-value of 274149589 with a P-value of 0 showing that the difference in the distributions is statistically significant. VPN connections have a median RTT difference of 109% compared to direct communication, which has a median RTT difference of 5%. As expected, the RTT difference for direct communications is close to zero and much more significant for VPN communication. However, there are also VPN connections that have low RTT differences, some as low as 0%. We found that for VPN communication, these cases are mostly caused by instances of clients located in the same city or region as the VPN server. For direct connections, we found that some cases are more complicated; we found that residential vs. business networks have a more significant effect on communication delay than expected, and we will discuss these cases in Section VI.

The minimal client RTT below which detection has been proven impossible is 3[ms], and detection reaches a true positive rate of 50% at 25[ms]. When the distance the packet travels is very short, the RTT difference becomes less significant than our minimal detection threshold (equation 1) which is based on standard network jitter, congestion, and IP geolocation errors. The rest of the low and negative RTT difference values are rare cases of extensive network delays for landmark communication, which can be caused by random packet drops or congestion.

Analysis of the detection results with the standard metrics of accuracy, precision, and recall for our detection rate. We observed a significant effect of the region of the perceived IP of the client on our detection method. Therefore, we chose to visualize the detection performance metrics for each continent (see Figure 7). Whether the communication originates from a direct client or passes through a VPN in that region, the communication passes through the infrastructure of the presented region. We believe that the observed regional effect can be correlated with the quality of the network infrastructure. Less developed regions (South America, parts of Asia, and Africa) which tend to have lower quality network infrastructure can be clearly seen in the data to have lower detection rates contrasted by the high reliability observed in more developed regions (North America, Europe, Oceania), where SNITCH achieved an accuracy rate between 89- 93%, which can be seen in Figure 7.

As an additional factor to the binary detection value we have

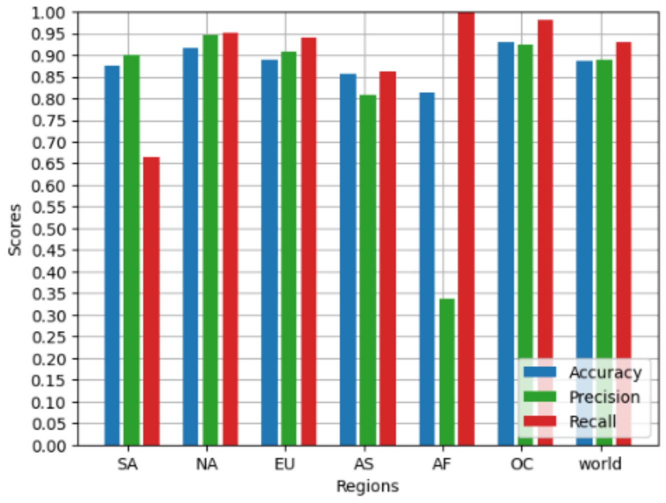


Figure 7. Detection metric scores with 0.5 confidence based on the geographic region of the IP address visible to the web server.

a confidence value that we can use to set our detection thresholds, we generated performance metrics for binary detection, 0.5 confidence, and full confidence, as can be seen in Figure 8. Increasing the confidence threshold improves the detection performance, but detection entries whose confidence values do not meet the specified confidence threshold are discarded. Detection with 0.5 confidence for all communications returns results of: accuracy: 0.891, precision: 0.897, recall: 0.939

We have not found statistically significant differences in detection rates between the tested services or when different protocols are used. These results (with acknowledgment of the limitations of the experiment) show that server-side VPN detection using geolocated landmarks and RTT measurement is effective and a viable option for security services.

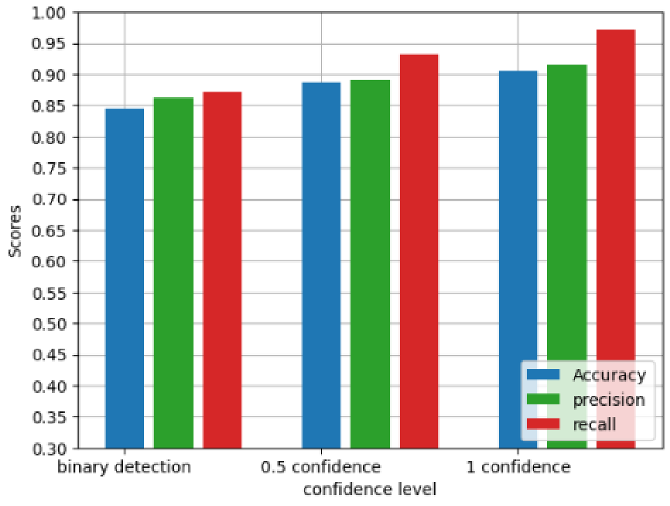


Figure 8. Detection confidence threshold metric score comparison.

VI. Discussion

We are proposing a new method for VPN detection based on RTT measurement and IP geolocation, which is triggered once proxy detection fails. With the detection of both proxies and VPNs, a fully implemented SNITCH system can be used to detect any kind of "proxy" usage (HTTPS Proxy, VPN, SSH forwarding, TOR, etc.).

SNITCH overcomes the limitations of existing passive and active detection solutions. Enabling detection of hijacked machines used as first-time VPNs, and circumventing the need for the TCP connection to break, or the requirement of ICMP responses from the VPN server.

A hybrid approach that uses passive VPN detection alongside SNITCH (the passive VPN detection can usually be paired with the IP geolocation service) can lead to more indicative detections. If a VPN is detected by SNITCH but is not detected by the passive detection system, that could be indicative of private VPN usage which has a higher risk of malicious intent.

Our initial assumptions of direct correlation between distance traveled and RTT and the following significant difference between VPN and direct communication RTT hold true. We have shown consistency in different services, protocols, and modern network infrastructures. We also found some limitations that can be overcome in future work:

1) Distance vs Jitter: As we have shown, the RTT is directly correlated with the distance traveled. Therefore, with an expected jitter of up to 30[ms] [44] plus standard network congestion, in instances where the client and the VPN server are geographically close the jitter becomes very significant relative to the RTT between the machines. This makes our method unreliable in detecting VPN usage within a suitable margin of error that addresses both standard network jitter and congestion.

We can show the effect of jitter vs distance and our effective range, using a distance-to-ratio and confidence analysis presented in Figure 9. Both the ratio of CS_{RTT} versus LS_{RTT} and the detection confidence increase consistently as the distance from the client to the VPN increases, with the detection confidence increasing rapidly starting from 500 [km] where the confidence was effectively 0 to 1000 [km] where the confidence was greater than 0.3. These results show that as the effect of network jitter becomes less significant compared to the increase in RTT, caused by the distance traveled between the VPN server and the client, detection improves as expected. Furthermore, we can see that our effective range starts at a distance of 500 [km] between the client and the VPN server.

Future works can implement geolocation-based DNS entries and diversified server locations to ensure ideal detection conditions for every direct client and improve conditions for VPN clients (the distance between the client and VPN is inaccessible to the detection server) without affecting user experience. However, it is important to note that if the VPN server is geographically close to the client, The IP geolocation will likely generate a reasonably accurate geolocation for current security or regulation purposes.

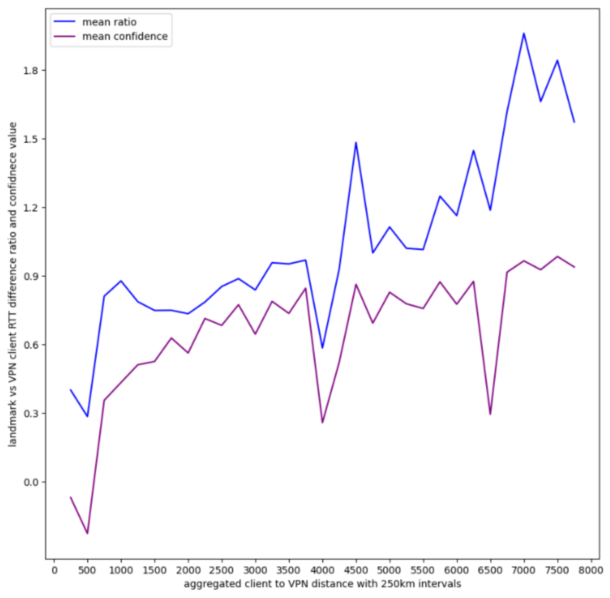


Figure 9. $\frac{CS_{RTT}-LS_{RTT}}{LS_{RTT}}$ ratio and detection confidence measurements for VPN client connections, aggregated over 250[km] intervals.

2) Network Infrastructure: Based on the experiments measurements, we found that regions with outdated network infrastructure (namely Africa and parts of Asia) have significant deviations in network delay, which significantly affects the results of SNITCH. This is probably due to high packet drop rates and low bandwidth. Above a certain threshold of jitter (represented by the RTT STD), detection becomes very unreliable. SNITCH filters out connections with excessive jitter. The filter can be further refined in future iterations with other relevant metrics to increase reliability. Detection of slow/outdated network infrastructure based on prior knowledge or variance measurements can trigger additional measurements, harming the user experience but ensuring security. This method could also be used to detect malfunctioning landmarks.

3) Business vs residential Networks: A broader issue that we identified that affects delays globally is the significant difference in delay between residential and business networks. From our measurements, RIPE Atlas Probes that are geographically close to direct clients but are located in business networks have a consistently shorter RTT than direct clients which are in residential networks. This effect occurs even when the connection is stable and the jitter is very low and causes an increase in false positives. The difference between business and residential networks can be addressed in multiple ways. For example, increasing the number of landmarks in residential networks and differentiating between business and residential networks when determining the error margins needed for detection.

4) Landmark distribution: A significant limiting factor for SNITCH is the distribution of geolocated landmarks worldwide. A large portion of the connections from residential proxies in the experiment could not be detected at all due to the lack of nearby landmarks. To be able to use SNITCH as a robust security solution, landmark coverage needs to be significantly improved worldwide, especially in residential networks. We believe that increasing the radius significantly would make the approximate RTT less representative of the actual VPN server RTT and therefore would reduce detection accuracy. 88% of the landmarks used for detection were less than 50 [km] from the source IP location, making the data for this analysis highly skewed. However, it should be noted that there was a marked decrease in detection accuracy from 91.1% to 88.9% for detections with landmarks less than 50 [km] distance vs. greater than 50 [km], which aligns with our initial analysis but requires more robust verification.

The lack of landmark coverage is a very significant impediment to the implementation of SNITCH as a security product. In future work, the effect of changing the search radius of landmarks should be explored, to check if a better balance between coverage and accuracy can be achieved. Exploration of additional landmark sources besides RIPE Atlas should also be carried out to increase the usable distribution.

5) IP geolocation errors: If a significant error occurs in IP geolocation , it is very likely that the detection accuracy will be negatively affected. To make sure that does not happen, there are a few measures that future implementations can take. Future implementations could cross-reference multiple IP geolocation services to ensure accuracy, and in cases of significant deviations, return an error, or take additional steps to verify the location, including active detection methods.

A. Mitigation methods

An essential question to ask ourselves is what mitigation methods VPN services can apply to avoid detection using our method. A somewhat “simple” mitigation would be breaking both TCP and TLS communication in a similar fashion, as discussed in BADPASS, E. Chiapponi et al. [31]. This method leads to a heavy technical burden, which is very costly for busy servers or residential IP servers which try to limit their presence as much as possible. More importantly, it could cause major privacy issues for clients because the servers will have access to encrypted information passing through them unless a second encryption layer is implemented. If a service does decide to break both the TCP and TLS connections, there are other possible ways to measure the RTT at the application layer or using JavaScript, these are more intrusive but not hard to implement. A mitigation method that affected previous solutions, of artificially increasing the RTT at the VPN server, would not detract from our detection rates and would likely improve them. Another possible mitigation is fully spoofing the VPN server IP addresses to cause errors in geolocation and detection. This is technically difficult and could be mitigated with more sophisticated active IP geolocation techniques, making this method unlikely to become a major issue.

Clients can circumvent our detection by opting to use VPN servers that are geographically close to their location. Effectively hiding their identity without significant geolocation spoofing.

VII. Conclusion

This work presents SNITCH: a server-side VPN detection solution which uses real-time network delay measurements and geolocated landmarks. Unlike existing passive approaches, which rely heavily on known VPN to IP address mappings or active methods that fail if the VPN does not 'break' the TCP connection. SNITCH identifies VPN connections without prior knowledge of specific VPN endpoints or performing intrusive client-side actions.

The method showed strong detection performance, accurately classifying connections from leading VPN services in diverse regions and achieving a detection accuracy between 89-93% in regions with modern network infrastructure and above 84% for all regions. Crucially, SNITCH can be seamlessly integrated into an authentication process to extend the capabilities of existing passive VPN detection solutions. Such a hybrid detection system would enhance the security layer by providing additional mitigation against sophisticated adversaries that attempt to evade geolocation-based defenses using unknown private VPNs.

Our future work should explore refining jitter handling mechanisms, residential vs. business network detection metric analysis, and exploring methods to increase landmark coverage.

References

[1] A. Dutkowska-Żuk, A. Hounsel, A. Morrill, A. Xiong, M. Chetty, and N. Feamster, "How and why people use virtual private networks," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 3451–3465. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/dutkowska-zuk

[2] CyberNews, "What can i do with a vpn?" https://cybernews.com/how-to-use-vpn/what-can-i-do-with-a-vpn/, (accessed Jan. 7, 2025).

[3] M. ATT&CK, "T1090: Proxy," https://attack.mitre.org/techniques/T1090/, (accessed Nov. 17, 2024).

[4] ——, "T1604: Proxy through victim," https://attack.mitre.org/techniques/T1604/, (accessed Nov. 17, 2024).

[5] CISA, "Cybersecurity advisory aa22-158a," https://www.cisa.gov/news-events/cybersecurity-advisories/aa22-158a, (accessed Nov. 17, 2024).

[6] Heimdalsecurity, "Microsoft chinese threat actors botnet credential theft," https://heimdalsecurity.com/blog/microsoft-chinese-threat-actors-botnet-credential-theft/, (accessed Nov. 17, 2024).

[7] P. Identity, "Zero trust security fundamentals," https://www.pingidentity.com/en/resources/identity-fundamentals/zero-trust-security.html, (accessed May 19, 2024).

[8] SEON, "Device intelligence to filter out fraudsters," https://seon.io/resources/device-intelligence-to-filter-out-fraudsters/, (accessed May 19, 2024).

[9] E. Research, "Zero trust security market," https://www.emergenresearch.com/industry-report/zero-trust-security-market, (accessed May 19, 2024).

[10] F. B. Insights, "Fraud detection and prevention market," https://www.fortunebusinessinsights.com/industry-reports/fraud-detection-and-prevention-market-100231, (accessed May 19, 2024).

[11] SEON, "Vpn detection tests," https://seon.io/resources/vpn-detection-tests/, (accessed Mar. 12, 2024).

[12] MaxMind, "Proxy/vpn fraud detection," https://www.maxmind.com/en/solutions/proxy-vpn-fraud-detection, (accessed Mar. 12, 2024).

[13] IPQualityScore, "Vpn/risk rating," https://www.ipqualityscore.com/, (accessed Mar. 12, 2024).

[14] H. Hoogstraaten, "Evaluating server-side internet proxy detection methods," Master's thesis, [University], 2018.

[15] MaxMind, "How accurate is ip geolocation?" https://blog.maxmind.com/2021/07/how-accurate-is-ip-geolocation/, (accessed Mar. 12, 2024).

[16] A. Gulati, B. D. Reddy *et al.*, "Credit card fraud detection using neural network and geolocation," in *IOP Conference Series: Materials Science and Engineering*, vol. 263, 2017, p. 042039.

[17] B. Eriksson, P. Barford, J. Sommers, and R. Nowak, "A learning-based approach for ip geolocation," in *Passive and Active Measurement*, ser. Lecture Notes in Computer Science, A. Krishnamurthy and B. Plattner, Eds., vol. 6032. Springer, Berlin, Heidelberg, 2010, pp. xx–xx.

[18] B. Chandrasekaran *et al.*, "Alidade: Ip geolocation without active probing," Department of Computer Science, Duke University, Tech. Rep. CS-TR-2015.001, 2015.

[19] B. Eriksson, P. Barford, B. Maggs, and R. Nowak, "Posit: a lightweight approach for ip geolocation," *SIGMETRICS Performance Evaluation Review*, vol. 40, no. 2, pp. 2–11, 2012.

[20] B. Du, M. Candela, B. Huffaker, A. C. Snoeren, and K. Claffy, "Ripe ipmap active geolocation: Mechanism and performance evaluation," *ACM SIGCOMM Computer Communication Review*, vol. 50, pp. 3–10, 2020.

[21] S. Laki, P. Mátray, P. Hága, T. Sebők, I. Csabai, and G. Vattay, "Spotter: A model based active geolocation service," in *Proceedings of the 2011 IEEE INFOCOM*, 2011, pp. 3173–3181.

[22] M. Mansoori and I. Welch, "How do they find us? a study of geolocation tracking techniques of malicious web sites," *Computers & Security*, vol. 97, p. 101948, 2020.

[23] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, "Towards ip geolocation using delay and topology measurements," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC '06)*, 2006, pp. 71–84.

[24] Microsoft, "Conditional access: Location condition," https://learn.microsoft.com/en-us/entra/identity/conditional-access/location-condition, (accessed Feb. 10, 2024).

[25] "General data protection regulation (gdpr) - chapter 5," https://gdpr-info.eu/chapter-5/, (accessed Mar. 12, 2024).

[26] R. Editor, "A framework for ip based virtual private networks," https://www.rfc-editor.org/rfc/rfc2764, rFC 2764, Mar. 2000 (accessed Mar. 12, 2024).

[27] OpenVPN, "Openvpn protocol," https://openvpn.net/community-resources/openvpn-protocol/, (accessed Mar. 12, 2024).

[28] P. I. Access, "Private internet access," https://www.privateinternetaccess.com/, (accessed Nov. 10, 2024).

[29] NordVPN, "Nordvpn," https://nordvpn.com/, (accessed Feb. 10, 2024).

[30] ExpressVPN, "Expressvpn," https://www.expressvpn.com/, (accessed Feb. 10, 2024).

[31] E. Chiapponi, M. Dacier, O. Thonnard, M. Fangar, and V. Rigal, "Badpass: Bots taking advantage of proxy as a service," in *Information Security Practice and Experience*, ser. Lecture Notes in Computer Science, C. Su, D. Gritzalis, and V. Piuri, Eds., vol. 13620. Springer, Cham, Switzerland, 2022.

[32] D. Xue, R. Ramesh, A. Jain, M. Kallitsis, J. A. Halderman, J. R. Crandall, and R. Ensafi, "Openvpn is open to vpn fingerprinting," *Commun. ACM*, vol. 68, no. 1, p. 79–87, Dec. 2024. [Online]. Available: https://doi.org/10.1145/3618117

[33] C. Wang, J. Yin, Z. Li, H. Xu, Z. Zhang, and Q. Liu, "Identifying vpn servers through graph-represented behaviors," in *Proceedings of the ACM Web Conference 2024*, ser. WWW '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1790–1799. [Online]. Available: https://doi.org/10.1145/3589334.3645552

[34] A. Goel, A. Kashyap, B. D. Reddy, R. Kaushik, S. Nagasundari, and P. B. Honnavali, "Detection of vpn network traffic," in *2022 IEEE Delhi Section Conference (DELCON)*, 2022, pp. 1–9.

[35] S. Miller, K. Curran, and T. Lunney, "Detection of virtual private network traffic using machine learning," *International Journal of Wireless Networks and Broadband Technologies*, vol. 9, pp. 60–80, 07 2020.

[36] R. Editor, "Transmission control protocol (tcp)," https://datatracker.ietf.org/doc/html/rfc9293, rFC 9293, Sept. 2021 (accessed Mar. 12, 2024).

[37] ——, "The transport layer security (tls) protocol version 1.3," https://datatracker.ietf.org/doc/html/rfc8446, rFC 8446, Aug. 2018 (accessed Mar. 12, 2024).

[38] N. Tschacher, "Detecting proxies and vpns with latency measurements," https://incolumitas.com/2021/06/07/detecting-proxies-and-vpn-with-latencies/, (accessed Feb. 25, 2024).

[39] R. Ramesh, P. Winter, S. Korman, and R. Ensafi, “{CalcuLatency}: Leveraging {Cross-Layer} network latency measurements to detect {Proxy-Enabled} abuse,” in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 2263–2280.

[40] A. T. Webb and A. L. N. Reddy, “Finding proxy users at the service using anomaly detection,” in *2016 IEEE Conference on Communications and Network Security (CNS)*, 2016, pp. 82–90.

[41] P. Callejo, M. Gramaglia, R. Cuevas, and Cuevas, “A deep dive into the accuracy of ip geolocation databases and its impact on online advertising,” *IEEE Transactions on Mobile Computing*, vol. 22, no. 8, pp. 4359–4373, Aug 2023.

[42] Y. Xie, Z. Zhang, Y. Liu, E. Chen, and N. Li, “Evaluation method of ip geolocation database based on city delay characteristics,” *Electronics*, vol. 13, p. 15, 2024.

[43] scaleway, “Network latency: how latency, bandwidth & packet drops impact your speed,” https://www.scaleway.com/en/blog/understanding-network-latency, (accessed Feb. 4, 2025).

[44] Obkio, “What is jitter?” https://obkio.com/blog/what-is-jitter/, (accessed Dec. 16, 2024).

[45] B. Data, “Residential proxies,” https://brightdata.com/proxy-types/residential-proxies, (accessed Dec. 9, 2024).

[46] Selenium, “Webdriver documentation,” https://www.selenium.dev/documentation/webdriver/, (accessed Mar. 15, 2024).

[47] PureVPN, “Purevpn,” https://www.purevpn.com/, (accessed Nov. 10, 2024).

[48] R. Editor, “Ikev2 protocol,” https://datatracker.ietf.org/doc/html/rfc5996, rFC 5996, Oct. 2010 (accessed Mar. 18, 2024).

[49] WireGuard, “Wireguard protocol,” https://www.wireguard.com/papers/wireguard.pdf, (accessed Mar. 12, 2024).

[50] X. Mi, X. Feng, X. Liao, B. Liu, X. Wang, F. Qian, Z. Li, S. Alrwais, L. Sun, and Y. Liu, “Resident evil: Understanding residential ip proxy as a dark service,” in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1185–1201.

[51] B. Data, “We are bright data. we were luminati,” https://brightdata.com/luminati), (accessed Feb. 10, 2025).

[52] IPinfo, “Ip geolocation api,” https://ipinfo.io/products/ip-geolocation-api, (accessed Mar. 12, 2024).

[53] R. NCC, “Ripe atlas probes,” https://atlas.ripe.net/, (accessed Feb. 25, 2024).