Linking Suspicious and Insecure Apps in the Virtual Private Network Ecosystem

By Benjamin Mixon-Baca, Dr. Jeffrey Knockel, and Dr. Jedidiah R. Crandall

Abstract

Ownership transparency in the VPN ecosystem allows users to make informed decisions about who they trust with their data. Researchers have recently begun investigating the relationships between seemingly distinct VPN providers, but such analysis is currently limited to a small sample of providers in the VPN ecosystem. One known family of providers — Innovative Connecting, Autumn Breeze, and Lemon Clove — has been previously scrutinized by two research efforts that have linked them to the People's Liberation Army in China.

Dr. Jeffrey Knockel, Dr. Jedidiah R. Crandall, and I identified and analyzed three families of VPN providers. Combined, their download counts on the Google Play Store exceed 700 million.[1] Similar to previous research, we used information from business filings and Android Package Kits (APKs) to link distinct providers together. However, we also built upon past work by introducing new methods for revealing how VPN providers are connected. In this report, we show that providers even share VPN servers' cryptographic credentials, including Shadowsocks passwords that are hard-coded into their APKs (hard-coded Shadowsocks passwords allow an attacker to decrypt the traffic of these providers' clients, compromising the security claimed by these providers).

Our analysis reveals that these applications (apps) share not only common ownership, but a common set of security issues. This led us to conclude that their providers are not merely misleading users about their ownership, but about the extent of their security properties as well.

[1] We focused on Google Play Store because the majority of downloads come from this app distributor, and because more than half of the VPN apps we identified initially were not on Apple's App Store.

Table of Contents

1. Introduction

To ensure the security and privacy of their network communications, users must know who develops, owns, and operates their VPN services. This is because VPN operators are able to observe all communications transmitted by or to each client. VPN providers obfuscating their ownership interferes with users' ability to make informed decisions about who to trust with their data. Furthermore, any deception on the part of the providers undermines user trust and may suggest other problems, such as with privacy and security practices of the provider in question.

In this paper, we analyzed the privacy and security of VPN apps whose providers intentionally disguise their ownership. We used hidden relationships between supposedly distinct VPN providers as an indirect indicator of deceptive behavior. We then searched for VPN-specific security issues in the deceptive providers' apps to uncover shared flaws. These shared flaws themselves serve as a signature by which to map relationships between providers.

VPN Pro was the first to widely report on hidden provider relationships.[2] They identified seemingly obfuscated relationships between Innovative Connecting, Autumn Breeze, and Lemon Clove. According to VPN Pro's research, these providers claimed to be based in Singapore, but were owned by a Chinese national and subject to Chinese law. VPN Pro linked these providers together using information collected from WHOIS records and resources in the binary the providers distributed through app stores. Recently, the Tech Transparency Project (TTP) uncovered links between these VPN providers (and multiple other providers) and Chinese computer security firm Qihoo 360, a company the United States government sanctioned in June 2020 for its connection to the People's Liberation Army (PLA).[3] Their findings were based on legal documents, such as mergers and acquisitions filings.

These reports together reveal concerning details about ownership transparency in the VPN ecosystem and motivated us to ask the following questions: To what extent do VPN providers provide inaccurate or incomplete corporate-ownership information on the Google Play Store? Using an analysis of legal filings, APK artifacts, and network communications, can seemingly distinct VPN apps be clustered into common-operator families? Which forensic signals are most reliable for that clustering? Within each detected family, to what extent are code bases, server infrastructure, and cryptographic materials reused? And how does such reuse contribute to exposing users to widespread security issues?

To answer these questions, we created a list of seemingly distinct VPN providers and their respective applications. We collected information about the providers from their Google Play pages, APKs, websites, domain registration information, business filings, their social media presence, and social media posts on Google, GitHub, Reddit, Twitter/X, and LinkedIn. We used

[2] Dovydas Vėsa and Justė Kairytė Barkauskienė. 2019. Who owns your VPN? 105 VPNs run by just 24 companies. https://vpnpro.com/blog/hidden-vpn-owners-unveiled-97-vpns-23-companies/ Accessed on: April 20, 2025.
[3] Tech Transparency Project. 2025. Apple Offers Apps With Ties to Chinese Military. https://www.techtransparencyproject.org/articles/apple-offers-apps-with-ties-to-chinese-military/ Accessed on: April 20, 2025.

this information to look for VPN providers that appear to be obfuscating their ownership information and using deceptive business practices.

We identified three distinct families of VPN providers, and analyzed each for security issues. We found that, in addition to sharing other code similarity features, each family of apps also shared problematic security properties. For two of the families, we identified hard-coded Shadowsocks passwords shared by their VPN clients. We also confirmed that we could decrypt their traffic when these apps were using the Shadowsocks protocol. An attacker can use these credentials to decrypt all of the traffic for all of the clients of these providers. These families alone have over 700 million Google Play Store downloads. Therefore, these issues put the traffic of large numbers of users at risk.

The remainder of this paper is structured as follows: In Section 2, we provide background on VPN technologies and their previously known weaknesses. Section 3 covers the methodology we used for VPN selection and our analysis. Section 4 details our findings. Lastly, we discuss in Section 5 the implications of these findings, as well as potential remedies.

2. Background

VPN apps implement a variety of tunneling protocols, which can operate on either the network layer or the application layer of the Open Systems Interconnection (OSI) model. In this section, we provide a brief summary of the security issues we looked for in each app depending on its supported protocol(s). We then conclude the section by summarizing previous research identifying hidden connections between distinct VPN providers.

2.1 VPN Protocols

VPNs are not defined by a single protocol. Rather, a VPN is an abstract model of networking that many unique protocols satisfy. The protocols of interest in this work operate at either the network layer or the application layer, each of which have specific security implications.

2.1.1 Network Layer VPNs

Network Layer VPNs are truly VPNs in that they create a network layer connection between the VPN client and VPN server. Examples of network layer VPNs include OpenVPN,[4] WireGuard,[5] and Internet Protocol Security (IPsec).[6] When a client transmits a packet, the packet is routed

[4] Inc. OpenVPN Technologies. 2002. OpenVPN. https://openvpn.net/community-resources/. Open-source VPN software. Accessed on: July 31, 2025.
[5] Jason A Donenfeld. 2017. WireGuard: Next Generation Kernel Network Tunnel. In NDSS. 12 pages.
[6] Sheila Frankel and Suresh Krishnan. 2011. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. https://www.rfc-editor.org/info/rfc6071. Accessed on: July 31, 2025.

through a TUN/TAP (in the case of OpenVPN and IPsec) or WireGuard (wg) interface. The packet, including all network and transport layer headers, is encrypted, encapsulated into a payload of a new packet destined for a VPN server, and then routed to it. When the VPN server receives the packet, the VPN server program decapsulates and decrypts the packet, then routes it to the original destination. The VPN server is acting, essentially, as a network address translation (NAT) layer for clients. Encryption is typically intended to protect the payload from eavesdroppers between the VPN client and VPN server, and the NAT obfuscates the client's public Internet Protocol (IP) address.

These are the purported security features of VPNs. The detail that makes these protocols implement truly virtual private networks is the inclusion of network and transport layer headers, and the reliance on the underlying operating system's network stack, routing tables, and firewall hooks to facilitate NAT and routing.

2.1.2 Network Layer VPN Weaknesses

Several security issues have been discovered that are specific to modern network layer VPNs.[7],[8],[9]

Blind-in/on-path Attacks

Blind-in/on-path attacks are a class of attack where an in/on-path attacker, while unable to observe the traffic inside of the tunnel (i.e., "blind"), can still tamper with the inner connection.[10] This attack comes in two flavors: client-side and server-side. The client-side attack permits an attacker to infer active connections between a VPN client and the external server to which the client is communicating through the VPN tunnel. The server-side attack permits an attacker between the VPN client and server to infer active connections or inject packets into the connection and take it over.

The client-side attack assumes the attacker is on a local network, such as a shared Wi-Fi network at a conference venue. The attacker spoofs packets to the client, setting the destination IP to the IP of the client's TUN//TAP or wg interface's and source IP to the web server's IP. If the source and destination ports match and the Transmission Control Protocol (TCP) sequence number is in-window, the target responds with a challenge acknowledgement (ACK) through the

---

[7] Benjamin Mixon-Baca, Jeffrey Knockel, Diwen Xue, Tarun Ayyagari, Deepak Kapur, Roya Ensafi, and Jedidiah R Crandall. 2024. Attacking connection tracking frameworks as used by virtual private networks. Proceedings on Privacy Enhancing Technologies 2024 (2024), 109–126. Issue 3.

[8] William J. Tolley, Beau Kujath, Mohammad Taha Khan, Narseo Vallina-Rodriguez, and Jedidiah R. Crandall. 2021. Blind In/On-Path Attacks and Applications to VPNs. In the 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, 3129–3146. https://www.usenix.org/conference/usenixsecurity21/presentation/tolley. Accessed on: July 31, 2025.

[9] Nian Xue, Yashaswi Malla, Zihang Xia, Christina Pöpper, and Mathy Vanhoef. 2023. Bypassing Tunnels: Leaking VPN Client Traffic by Abusing Routing Tables. In the 32nd USENIX Security Symposium (USENIX Security 23). USENIX Association, Anaheim, CA, 5719–5736. https://www.usenix.org/conference/usenixsecurity23/presentation/xue. Accessed on: July 31, 2025.

[10] See footnote 7.

VPN tunnel. The attacker will infer the ACK based on its size, implying an active connection despite the tunnel encryption. Source address validation (i.e., strict reverse path filtering, (rp_filter) in Linux) mitigates this threat, although mobile phones default to loose source address validation.

The server-side attack, on the other hand, works even when the attacker is not on the local network, such as a router or Internet Service Provider (ISP) between the VPN server and client. Instead of spoofing a packet to the client directly, the attacker spoofs packets to the VPN server. If the spoofed packet matches the connection, then it will reach the VPN client. The client will respond with a challenge ACK or process the attacker's packet as if it is a legitimate packet. There is no known mitigation for a server-side attack, because the root cause is fundamental to VPN architecture (IP sharing).

Port Shadow Vulnerabilities

One limitation of blind-in/on-path attacks is the on-path requirement, because it is a privileged location. The port shadow vulnerability permits attackers without that access to achieve it by allowing the attacker to escalate from adjacent (i.e., the attacker and client connect to the same VPN server) to in-path.[11] In this work, however, we did not find any VPNs vulnerable to this attack, primarily due to the VPNs' use of secure network layer protocols, such as IPsec, or alternatively, their use of application layer protocols (which are never vulnerable), such as Shadowsocks.

2.2 Application Layer VPNs

The only application layer VPN we considered in this work is Shadowsocks. Shadowsocks operates similarly to network layer VPNs, in that a client sends a packet to an intermediary (the VPN/proxy server) that then forwards the packet to the client's intended destination. The major difference is that application layer VPNs do not operate on the Layer 3 and Layer 4 network layers the way network layer VPNs do.

2.2.1 Shadowsocks

Shadowsocks is an application layer proxy designed explicitly to circumvent the Great Firewall of China.[12] Its use is offered by many VPN providers, including those featured in this study. Shadowsocks is a Fully Encrypted Protocol (FEP),[13] meaning it attempts to look completely random. Shadowsocks operates similarly to Socket Secure 5 (SOCKS5) proxies and adds

[11] See footnote 7.
[12] Clowwindy. 2012. 发一个自用了一年多的翻墙工具 shadowsocks. https://web.archive.org/web/20120422191812/http://www.v2ex.com/t/32777. Accessed on: July 31, 2025.
[13] Ellis Fenske and Aaron Johnson. 2023. Security notions for fully encrypted protocols. In Free and Open Communications on the Internet. 7 pages.

symmetric encryption to implement its FEP. When a user visits a website in their browser, the application layer data is sent through the proxy using the SOCKS5 protocol.

Application layer VPNs must take additional steps to route all TCP, User Datagram Protocol (UDP), and Internet Control Message Protocol (ICMP) traffic through the proxy. To facilitate this, libraries such as REDSOCKS[14] and tun2socks[15] intercept outgoing packets and route them to a Socket Sure (SOCKS) proxy that can then operate on the packets as required. This operation is similar to network layer VPNs in that the operating system's connection tracking framework intercepts and routes outgoing packets through the proxy. A service listens for and completes connections at Layer 4 with the service sending the packets to the Shadowsocks proxy. The application layer data is then encapsulated using the SOCKS5 protocol and sent to the proxy server.

2.2.2 Shadowsocks' Weaknesses

Providers who claim that a Shadowsocks-based VPN offers confidentiality or integrity are misleading their users. Although Shadowsocks uses (symmetric) encryption, it was not designed specifically to satisfy these or any other security properties.[16] Therefore, we are concerned with two weaknesses: decryption oracle attacks when using deprecated ciphers, and hard-coded passwords. In addition to these problems, a Shadowsocks client that uses REDSOCKS or tun2socks is susceptible to the client-side blind-in/on-path attack due to interaction with the connection tracking framework.

Decryption Oracles

Shadowsocks originally offered a "stream ciphers" suite for encryption. In 2020, Zhiniang Peng discovered that the packets lacked an integrity check, leading to a decryption oracle. An attacker could use this attack to decrypt client traffic when those ciphers are used,[17] resulting in their deprecation. Among the ciphers in the suite, only the Authenticated Encryption with Associated Data (AEAD)[18] one is still recommended. For example, if the VPN application contains a static configuration file, then this weakness can be identified by locating this file and identifying the cipher that the file specifies.

Hard-coded Passwords

Symmetric encryption uses the same key to encrypt and to decrypt. Therefore, after having extracted a hard-coded Shadowsocks password in a VPN app—from which a symmetric key is

---

[14] Leonid Evdokimov. 2011. REDSOCKS. https://darkk.net.ru/redsocks/. Accessed on: April 20, 2025.
[15] Jason Lyu. 2019. tun2socks. https://github.com/xjasonlyu/tun2socks. Accessed on: June 12, 2025.
[16] David Fifield. 2023. Comments on certain past cryptographic flaws affecting fully encrypted censorship circumvention protocols. Cryptology ePrint Archive, Paper 2023/1362. https://eprint.iacr.org/2023/1362. Accessed on: July 31, 2025.
[17] Ibid.
[18] Clowwindy. 2012. ShadowSocks: Stream Ciphers. https://shadowsocks.org/doc/stream.html. Accessed on: July 31, 2025.

deterministically derived—a network eavesdropper can decrypt all traffic for all clients of the affected application. Identifying this weakness requires the same actions as the decryption oracle: find the static information and record the key. An implementation facilitating private connections would use non-hard-coded keys.

2.3 Identifying Hidden VPN Provider Relationships, Ownership, and Deception

Before summarizing the previous research uncovering obfuscated ownership and hidden relationships between providers, this section first defines what we mean by "distinct provider," "link," "provider linkage," and "deception," which are terms we use throughout the remainder of this report.

2.3.1 Definitions

Distinct Provider, Link, and Provider Linkage

By "distinct provider," we mean the name a user sees when reading the developer details on an application's Google Play page, or in the terms of service or privacy policy documents available on the provider's website. "Link" means, given two distinct providers (such as Innovative Connecting and Lemon Clove, for example), a single entity operates both services. The process of "linking" or "provider linkage" is using information, such as business records, information within APKs, or shared VPN servers, to deduce that a single entity operates multiple providers.

Deception

It is not uncommon for VPN providers to appear to be distinct, but actually be operated by a single entity. For example, Kape Technologies operates multiple VPN services, including ExpressVPN and Private Internet Access. Kape discloses the fact that they operate both VPN services on their website so as to not be deceptive. When we speak of "deception," we mean that the relationship between providers requires uncovering by reviewing sources beyond those that an average user would use to find it, such as the provider's or app's website or Google Play page. This may include the need to inspect legal documents, domain registration information, decompiled APK files, and other sources.

2.3.2 Previous Research

The most convincing reports linking providers together and uncovering deceptive practices have leveraged business records. Research by Tech Transparency Project (TTP) linked Innovative Connecting, Autumn Breeze, Lemon Clove, and several other providers both to each other, and to Chinese firm Qihoo 360.[19] Their investigation involved comparing multiple sources of data, including mergers and acquisitions records. Qihoo 360 is based in mainland China, meaning that the operator is subject to Chinese laws and regulations that require censorship of certain

[19] See footnote 3.

content and enable surveillance of the internet. Another concerning detail, revealed by VPN Pro and corroborated by TTP, was the apparent lengths the providers went to hide the fact that they were owned by a Chinese national. This detail is not apparent when viewing the Google Play Store pages of any of these VPNs, nor is it on any website. Instead, these apps state they are Singapore-based.

Much of the work linking providers depends on business filings. While researchers have identified and localized various types of hosts and services (for example, proxies),[20],[21],[22],[23],[24],[25] similar methods have not been applied to linked VPN providers. Researchers have linked providers together by identifying potentially shared infrastructure,[26],[27],[28] identifying similarities between provider VPN clients, or performing text comparisons with privacy policies,[29] but definitive proof of such relationships typically remains elusive.

In our study, we realized the inherent limitations of using privacy policy comparisons or code comparisons to infer links. A developer taking shortcuts might copy and paste the privacy policy of a competitor. Two providers might employ the same software developer without knowing it. Therefore, while such comparison techniques are a good start, shared infrastructure is more

[20] Genevieve Bartlett, John Heidemann, and Christos Papadopoulos. 2007. Understanding passive and active service discovery. In Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (San Diego, California, USA) (IMC '07). Association for Computing Machinery, New York, NY, USA, 57–70. https://doi.org/10.1145/1298306.1298314. Accessed on: July 31, 2025.
[21] Shinyoung Cho, Zachary Weinberg, Arani Bhattacharya, Sophia Dai, and Ramsha Rauf. 2024. Selection of Landmarks for Efficient Active Geolocation. In 2024 8th Network Traffic Measurement and Analysis Conference (TMA). 1–9. https://doi.org/10.23919/TMA62044.2024.10559002. Accessed on: July 31, 2025.
[22] Manaf Gharaibeh, Anant Shah, Bradley Huffaker, Han Zhang, Roya Ensafi, and Christos Papadopoulos. 2017. A look at router geolocation in public and commercial databases. In Proceedings of the 2017 Internet Measurement Conference (London, United Kingdom) (IMC '17). Association for Computing Machinery, New York, NY, USA, 463–469. https://doi.org/10.1145/3131365.3131380. Accessed on: July 31, 2025.
[23] Zachary Weinberg. 2019. Toward Automated Worldwide Monitoring of Network-level Censorship. (1 2019). https://doi.org/10.1184/R1/7571876.v1. Accessed on: July 31, 2025.
[24] Zachary Weinberg, Shinyoung Cho, Nicolas Christin, Vyas Sekar, and Phillipa Gill. 2018. How to Catch when Proxies Lie: Verifying the Physical Locations of Network Proxies with Active Geolocation. In Proceedings of the Internet Measurement Conference 2018 (Boston, MA, USA) (IMC '18). Association for Computing Machinery, New York, NY, USA, 203–217. https://doi.org/10.1145/3278532.3278551. Accessed on: July 31, 2025.
[25] He Yan, Ashley Flavel, Zihui Ge, Alexandre Gerber, Dan Massey, Christos Papadopoulos, Hiren Shah, and Jennifer Yates. 2012. Argus: End-to-end service anomaly detection and localization from an ISP's point of view. In 2012 Proceedings IEEE INFOCOM. 2756–2760. https://doi.org/10.1109/INFCOM.2012.6195694. Accessed on: July 31, 2025.
[26] Anna Ablove. 2022. VPNalyzer: Researching VPN Vulnerabilities on a Large Scale. Technical Report. University of Michigan.
[27] Muhammad Ikram, Narseo Vallina-Rodriguez, Suranga Seneviratne, Mohamed Ali Kaafar, and Vern Paxson. 2016. An Analysis of the Privacy and Security Risks of Android VPN Permission-enabled Apps. In Proceedings of the 2016 Internet Measurement Conference (Santa Monica, California, USA) (IMC '16). Association for Computing Machinery, New York, NY, USA, 349–364. https://doi.org/10.1145/2987443.2987471. Accessed on: July 31, 2025.
[28] Reethika Ramesh, Leonid Evdokimov, Diwen Xue, and Roya Ensafi. 2022. VPNalyzer: systematic investigation of the VPN ecosystem. In Network and Distributed System Security, Vol. 10. 16 pages.
[29] See footnote 2.

compelling as evidence. Our investigation improves upon the current understanding of relationships between VPN providers, articulated in other research, by using the cryptographic credentials from one provider to establish a tunnel with the servers of a different provider. These shared, hard-coded credentials will also prove to be a shared security defect.

3 METHODS

Our methodology consisted of three steps. First, we collected developer and app names for 100 VPN providers. We then filtered this list down to 50 providers by excluding providers who were based in the United States. Lastly, by way of a security analysis, we identified the subset of VPNs that appear deceptive.

3.1 VPN Selection

We selected the 100 most downloaded VPN apps globally, according to SensorTower[30] and AppMagic,[31] who both provide aggregated metadata about mobile applications for market research. We combined the datasets by comparing the download counts for each dataset and selecting the larger value reported when they differed. The initial list was filtered down to 50 by including only those apps whose providers were based outside of the U.S. We prioritized Singapore-based providers because of the previous research that observed deceptive providers incorporating in this country.[32] We then collected information about the providers from their Google Play pages, websites, domain registration information, GitHub, GitLab, and Gitee accounts, as well as their social media pages. We identified 13 potentially related and deceptive providers. We then performed security analyses for each to search for more definitive evidence of deception via infrastructure connections.

In the remainder of this section we summarize our collection and analysis methodology for these sources below. Overall, we found that analyses of providers' websites, business filings, and apps' source code were the most useful for linking providers and identifying deception.

3.1.1 Websites

For each selected application, we collected the developer name, address or location, and website and privacy policy link from the Google Play Store, and downloaded the APK onto our analysis device. We then reviewed the website, privacy policy, terms of service, and APK code, and built a search-terms list to search Google, Twitter/X, Reddit, LinkedIn, Telegram, Facebook/Instagram, YouTube, GitHub, GitLab, and Gitee. When we decompressed the APKs, we noted the asset and shared library names, and searched the decompiled code for email

[30] SensorTower. 2024. SensorTower. https://sensortower.com/. Accessed on: June 7, 2025.
[31] AppMagic Inc. 2024. AppMagic. https://appmagic.rocks/. Accessed on: June 7, 2025.
[32] See footnote 1.

addresses, URLs, and similar terms that we added to our search-term list. A sample of the list of search terms that we used is provided in Table 2 in Appendix A.

3.1.2 Business Filings

We used the provider name cited on the Google Play Store and mentioned in each provider's privacy policy and terms of service documents as search terms in OpenCorporates to find business records for each provider.[33] An example search result is provided in Figure 1 below.

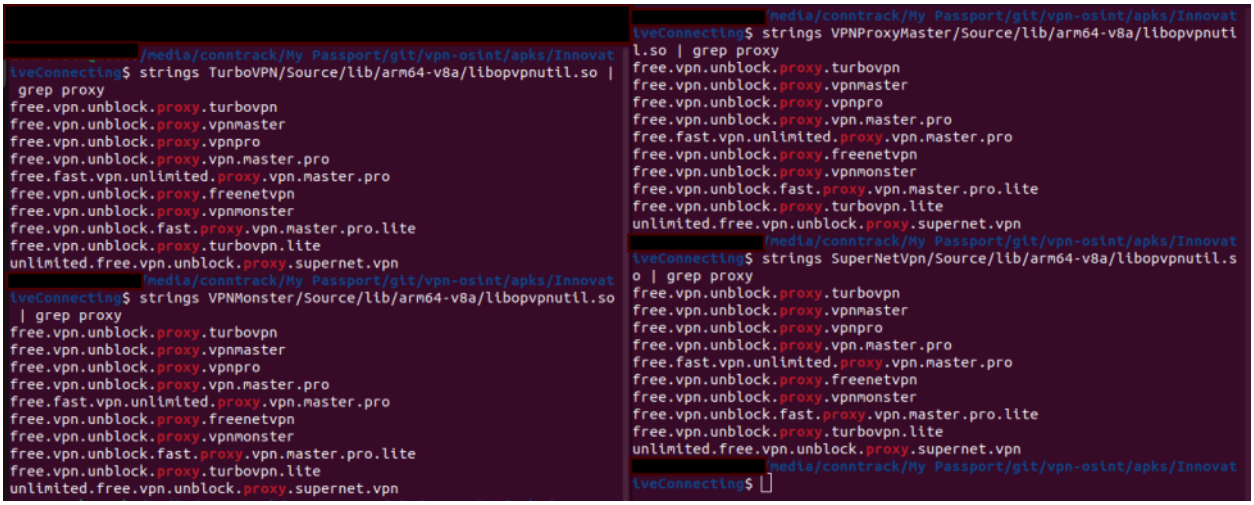

Figure 1: strings output showing references to the multiple VPN applications from supposedly distinct VPN providers.

We found 10 providers with records in OpenCorporates and three without such records. We then reviewed available business records, such as copyright filings. We cross-referenced addresses in these documents with addresses listed on websites and Google Play pages and noted inconsistency. We found 10 providers had addresses listed in China while claiming to be operating out of Singapore. The case of Innovative Connecting is shown in Figure 2 below.

3.1.3 Social Media

We searched for and noted VPN providers and apps that had social media profiles on Facebook, Instagram, Twitter/X, Reddit, Telegram, Discord, and YouTube. We did not find any information useful for provider linkage or identifying deception for our study. We also searched GitHub, GitLab, and Gitee for any repositories the VPN providers might maintain, but we failed to locate any such repositories.

[33] OpenCorporates is a London-based NGO that aggregates business records, like tax and copyright records, of companies around the world to promote transparency. See Open Corporates team. 2025. Open Corporates. https://opencorporates.com/. Accessed on: April 21, 2025.

Figure 2: Innovative Connecting linked to Beijing through copyright filings with USPTO.

3.1.4 DNS Records

We collected DNS records using dig and WHOIS. In all cases, the VPN providers either redacted all information from their WHOIS records or used a domain registrar, such as Domain Protection Services or Domain by Proxy, to anonymize this information. Thus, domain-related information was largely not useful for making an association between distinct VPN providers.

3.1.5 APKs

We downloaded the APK for each application onto our analysis device (detailed below in Section 3.2). We used the strings command to search for specific file paths and email addresses. We found shared libraries were the most productive source linking VPN providers together. For example, VPN applications from Innovative Connecting, Autumn Breeze, and

Lemon Clove each had strings in libopvpnutils.so explicitly linking the VPN applications for each provider together (see *Figure 1* above).

3.2 Security Analysis

Our analysis goals were twofold: first, identify VPN-specific threats; second, uncover deception via provider linkage. All applications were loaded onto a rooted Google Pixel 7a device for testing. We used a laptop as a Wi-Fi hotspot and connected the mobile device to the hotspot to collect packet captures using the laptop.

3.2.1 Finding VPN-specific Security Issues

We searched for the VPN-specific security issues outlined in Section 2 using static and dynamic analyses.

Static Analysis

For our static analysis, we compared the filenames and SHA256 hashes of files in the assets and lib/arm64 directories. We used the GNU strings command to extract strings from shared libraries. We then performed consistency checks of file types by comparing file extensions (for example, .png) with the file type reported by the file command. We also recorded instances when the file reported data file types and the file extension was either missing or of types for images (such as .png or .jpg) and common text file formats (such as .txt or .json). We used Jadx[34] to decompile .dex files and Ghidra[35] to decompile native code libraries.

Dynamic Analysis

For dynamic analysis, we focused on identifying credentials like WireGuard configuration files, OpenVPN and IPsec certificates, or Shadowsocks passwords. We used Frida[36] to trace the execution of specific libraries and functions we identified during static analysis and fridump[37] to extract objects like credentials from process memory. We noted instances when the app appeared to use anti-reverse-engineering countermeasures. We recorded packet captures to identify VPN server IP addresses and connections to APIs to which the VPN app made calls. We also configured the device to proxy traffic through mitmproxy[38] to search for API connections if the app used TLS outside the VPN tunnel for such connections.

---

[34] Skylot. 2025. JADX - Dex to Java decompiler. https://github.com/skylot/jadx. Accessed on: April 20, 2025.
[35] National Security Agency (NSA). 2019. Ghidra - Software Reverse Engineering Framework. https://ghidra-sre.org/. Accessed on: April 20, 2025.
[36] Ole André V. Ravnås and Håvard Sørbø. 2014. Frida - Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers. https://frida.re/. Accessed: April 20, 2025.
[37] Nightbringer21. 2016. fridump. https://github.com/Nightbringer21/fridump. Accessed: 21 June 2025.
[38] Aldo Cortesi, Maximilian Hils, Thomas Kriechbaumer, and contributors. 2010–. mitmproxy: A free and open source interactive HTTPS proxy. https://mitmproxy.org/. [Version 11.1].

4. Results

We identified three families, which we called Families A, B, and C, containing three, five, and two providers, respectively. Three of the VPN providers that we analyzed did not appear to have any similarities with the other VPNs, which we assign to an “other” catch-all group.

In the remainder of this section, for each family identified, we summarize the signatures that led us to cluster the families’ providers together. For each family, we also summarize the results of our security analysis. For Families A and B, we performed a full security analysis, and for Family C, we performed a partial analysis that did not consist of analyzing the apps’ cryptography. For the latter, we describe each of the security issues we identified, but leave analysis of their cryptography to future work.

4.1 Family A: Innovative Connecting, Autumn Breeze, and Lemon Clove

Family A consists of the providers Innovative Connecting, Autumn Breeze, and Lemon Clove, who collectively operate eight VPN applications.[39] Other researchers[40] have linked more apps to Lemon Seed, the holding company, but we could only confirm shared infrastructure used by these three providers specifically. Each application contained nearly identical decompiled Java code, shared libraries, and assets. The overt copying-and-pasting suggests a single developer implemented all of the apps and reused significant portions of code. While it is plausible that someone could have copied their APK and added their own code (given the signatures, security weaknesses, and shared infrastructure), this seems unlikely.

[39] These include Turbo VPN, Turbo VPN Lite, and VPN Monster developed by Innovative Connecting; VPN Proxy Master and VPN Proxy Master - Lite developed by Lemon Clove PTE LTD; and Snap VPN, Robot VPN, and SuperNet VPN developed by Autumn Breeze.
[40] See footnotes 2 and 3.

Family A VPN providers, their respective apps, and respective download count on the Google Play Store

Provider Name	*VPN Name*	*Google Play Downloads (millions)*
Innovative Connecting	Turbo VPN	100
	Turbo VPN Lite	50
	VPN Monster	10
Lemon Clove	VPN Proxy Master	100
	VPN Proxy Master - Lite	10
Autumn Breeze	Snap VPN	50
	Robot VPN	10
	SuperNet VPN	1

4.1.1 Signatures

We found consistent patterns across the apps' protocols, software implementations, and code obfuscation methods.

Protocols

Each app supported at least the IPsec and Shadowsocks protocols. IPsec was implemented in part by libcharon.so, libstrongswan.so, and libipsec.so, whereas Shadowsocks was implemented in part by libsslocal.so, libredsocks.so, and libtun2socks.so. In the case of Robot VPN, Snap VPN, SuperNet VPN, TurboVPN, and VPN Monster, the settings menu contained a radio button for OpenVPN, but it was not selectable at the time of analysis.

Code Signatures

We found significant code overlap both at the Java and native level, and within the assets. Each app contains four characteristic files: aaa_new.png, cert.pem, proxy.builtin, and server_offline.ser.

Defense Mechanisms

The code appeared to be designed to deceive analysts or automated security checks. For example, in some apps (for example, VPN Monster) the file aaa_new.png contains bytes that are read into a helper function that decrypts it and uses it for IPsec configuration. The key used

to decrypt this file is also built dynamically in native code. These characteristics were present in all of the apps in this family. We successfully extracted the IPsec and Shadowsocks configuration parameters using Frida during execution and from memory using fridump.

4.1.2 Security

We confirmed multiple security deficits shared by this family of VPN providers.

Blind-in/on-path Attacks

All eight applications were susceptible to client-side blind-in/on-path attacks. We found the underlying issue to be in libredsocks.so, which uses the operating system's firewall (Netfilter for Android) to divert all packets through the Shadowsocks tunnel. This design enables a client-side attack that lets an adversary interfere with active connections.

Location Information

We configured the devices to proxy their communication through mitmproxy. We found that the VPNs call APIs to, for example, download VPN configuration files, download public keys, and upload user telemetry. Even when the VPN did not request the location permission, it requested the zip code of the user's public IP from ip-api.com, which it subsequently uploaded to a Firebase endpoint. The apps' privacy policies all claim that they do not collect user addresses (personal or business), yet we observed them doing so.

Weak Encryption

The Shadowsocks configuration files use rc4-md5 for the encryption method and specify "plain" for the obfuscation technique. rc4-md5 is one of several deprecated ciphers supported by Shadowsocks. The Shadowsocks implementation does not discard the first 512 bytes of keystream prior to encryption, which is recommended,[41],[42] making a confirmation attack possible. Fortunately, the initial MD5 hashing, use of a longer initialization vector (IV), and

[41] Ilya Mironov. 2002. (Not So) Random Shuffles of RC4. In Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '02). Springer-Verlag, Berlin, Heidelberg, 304–319.
[42] Mathy Vanhoef and Frank Piessens. 2015. All Your Biases Belong to Us: Breaking RC4 in WPA-TKIP and TLS. In 24th USENIX Security Symposium (USENIX Security 15). USENIX Association, Washington, D.C., 97–112. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/vanhoef.

construction of key material protect against key-recovery attacks such as Fluhrer, Mantin, Shamir (FMS)[43] or Wi-Fi Protected Access (WPA) style attacks.[44],[45],[46]

Hard-coded Keys

The apps contain a hard-coded password for Shadowsocks configuration in the fileassets/server_offline.ser. This file is encrypted using AES-192-ECB. When the app first connects to a Shadowsocks server, it attempts to download a configuration file from a remote server, although we never observed the app successfully download a remote configuration. If or when that fails, the app then calls the native function, NativeUtils.getLocalCipherKey, implemented in the library, libopvpnutil.so. This function deterministically builds the secret key used to decrypt the configuration file. The key built depends on which VPN app is calling the function (for example, VPN Monster and TurboVPN build the same key, but VPN Monster and Snap VPN do not). We confirmed that server_offline.ser is the same file across different geographic locations and devices. Figure 3 depicts the Shadowsocks password shared by every user who has downloaded VPN Monster, TurboVPN, or TurboVPN Lite, and Figure 4 demonstrates how knowledge of such a password can decrypt their traffic.

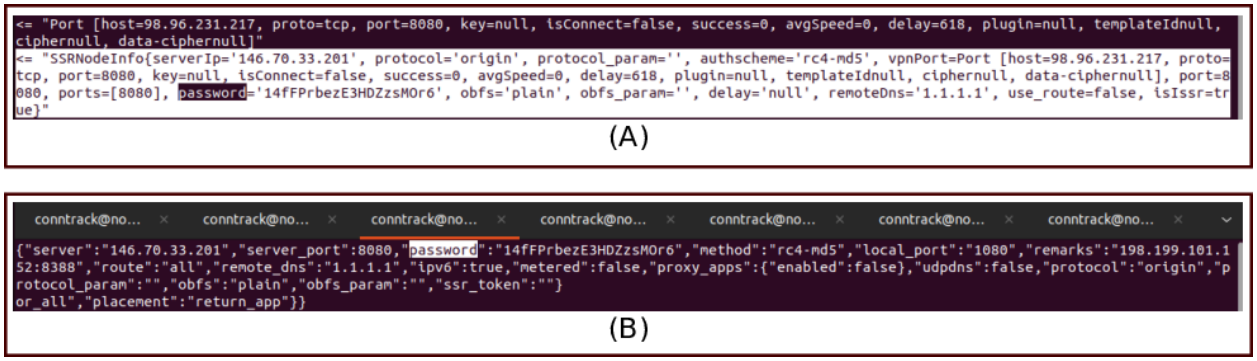

Figure 3: VPN Monster: Shadowsocks passwords are present in both the frida-trace (top) taken during execution and the memory dump for the VPN Monster process (bottom).

[43] Scott Fluhrer, Itsik Mantin, and Adi Shamir. 2001. Weaknesses in the Key Scheduling Algorithm of RC4. In Selected Areas in Cryptography, Serge Vaudenay and Amr M. Youssef (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 23 pages.
[44] See footnote 41.
[45] Nikita Borisov, Ian Goldberg, and David Wagner. 2001. Intercepting mobile communications: the insecurity of 802.11. In Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (Rome, Italy) (MobiCom '01). Association for Computing Machinery, New York, NY, USA, 180–189. https://doi.org/10.1145/381677.381695.
[46] Erik Tews and Martin Beck. 2009. Practical attacks against WEP and WPA. In Proceedings of the Second ACM Conference on Wireless Network Security (Zurich, Switzerland) (WiSec '09). Association for Computing Machinery, New York, NY, USA, 79–86. https://doi.org/10.1145/1514274.1514286.

Figure 4: Hard-coded keys in VPN Proxy Master enable a network eavesdropper to decrypt traffic (upper: the original, encrypted traffic; lower: encircled in red is the decrypted traffic).

Another consequence of the hard-coded password is that anyone who downloads the app can “freeload” off of the VPN service by establishing a Shadowsocks tunnel from a laptop using the extracted parameters. We used libopvpnutil.so as a search term in Google and GitHub, and found some instances of files with this name. However, the newly discovered instances lacked a feature present in the apps that are part of our study. The instances of libopvpnutil.so found in the apps in our study reference the various VPN apps distributed by these providers. This was one of the ways we connected them together. The versions on GitHub have no such signature. It appears that whoever developed these apps used the version on GitHub as a template for integrating custom native code.

Server Enumeration

Another consequence of the hard-coded password is that an attacker can enumerate additional VPN servers operated by the same entity. We tested this by selecting IP addresses in the same /24 as a VPN server confirming that we could connect to other servers on that /24. We used this capability to confirm that supposedly different VPN providers share server infrastructure.

4.1.3. Uncovering Hidden VPN Provider Relationships

Shared, Hard-coded Credentials

The credentials present in the hard-coded configuration files serve not only for offensive security purposes, that is, to decrypt user communications, but also to link distinct providers together. Because the hard-coded configuration files are the same, they contain the same credentials across different providers. This serves as a signature linking the distinct providers together.

Cross-provider Infrastructure Sharing

We used the above “freeload” capability to test whether the providers share the same servers. To do this, we established tunnels to VPN servers using each app and recorded the VPN server IP to which we connected. We then connected to each of them using the same hard-coded

credentials to confirm that Innovative Connecting, Autumn Breeze, and Lemon Clove share infrastructure.

4.2 Family B: MATRIX MOBILE PTE LTD, ForeRaya Technology Limited, WILDLOOK TECH PTE LTD, Hong Kong Silence Technology Limited, and Yolo Mobile Technology Limited

Family B consists of the five providers MATRIX MOBILE PTE LTD, ForeRaya Technology Limited, WILDLOOK TECH PTE LTD, Hong Kong Silence Technology Limited, and Yolo Mobile Technology Limited. Notably, all of these providers' privacy policies explicitly reference Innovative Connecting.

Family B VPN providers, their respective apps, and respective download count on the Google Play Store

Provider Name	***VPN Name***	***Google Play Downloads (millions)***
MATRIX MOBILE	Global VPN	10
	XY VPN	100
ForeRaya Technology Limited	Super Z-VPN	10
Hong Kong Silence Technologies Limited	Touch VPN - Stable & Secure	50
Yolo Mobile Technology Limited	VPN ProMaster-Secure your net	50
	3X VPN - Smooth Browsing	100
WILDLOOK TECH	VPN Inf	10
	Melon VPN - Secure Proxy VPN	50

4.2.1 Signatures

The patterns we saw across these apps consisted of the particular protocols they support and their implementations, identifying strings we found in the code, a particular form of code obfuscation that we observed, and the sharing of VPN server IP addresses.

Supported Protocols

All eight apps appeared to support only Shadowsocks, facilitated by libsslocal.so. They use libredsocks.so and libtun2socks.so to build the proxy.

Code Signatures

There is a mixture of similarities and differences in code signatures with these apps. While XY VPN and Super Z-VPN have similar code structures, they are different from Global VPN. Global VPN and Super Z-VPN reach out to the same Shadowsocks service at 149.28.197.166:443, but XY VPN does not. None of these apps are similar to Family A. For example, they use libsslocal.so, whereas Family A uses libss-local.so. All of the apps contain the file libcore.so, which contains explicit references to the APK files for these eight VPN clients.

Defense Mechanisms

XY VPN and Super Z-VPN obfuscate themselves at the Java level by concatenating real words together. They do this to make package and function names appear legitimate and even informative (for example, “RingAdaptorDecrypted”), but do not reflect functionality.

4.2.2 Security

We confirmed multiple security deficits shared by this family of VPN providers.

Blind-in/on-path Attacks

These applications are susceptible to connection inference attacks using client-side blind-in/on-path attacks, because of the libredsocks.so and libtun2socks.so dependencies.

Weak Encryption

The VPN clients contain obfuscated passwords in the shared library libcore.so. Similar to Family A, the apps decrypt a Shadowsocks configuration based on which APK is executing, and use a hard-coded password to connect to Shadowsocks servers. Each app offers 33 built-in servers. Each server has 19 open ports, each running a Shadowsocks process. Depending on the app’s package name, one of these ports and one of 14 hard-coded passwords are selected.

Server Enumeration

This family of providers appeared to use a smaller set of servers compared to Family A, and we were unable to enumerate servers beyond those specified in the apps' configuration files. All of Family B's VPN servers are hosted by a single company, GlobalTeleHost Corp. (gthost).

4.2.3. Uncovering Hidden VPN Provider Relationships

Shared, Hard-coded Credentials

Similar to Family A, Family B's use of shared, hard-coded credentials permit us to link distinct providers together. Unlike Family A, which has been linked by multiple other researchers, Family B was not previously known and is a novel finding contributing to research in this space.

Cross-provider Infrastructure Sharing

We collected IP addresses for each VPN and were able to confirm that the VPN apps connected to the same set of IP addresses—though as we noted earlier, they do so on different ports.

4.3 Family C: Fast Potato Pte. Ltd and Free Connected Limited

Family C consists of the providers Fast Potato Pte. Ltd and Free Connected Limited, who distribute Fast Potato VPN and X-VPN, respectively. Fast Potato has no business filings according to OpenCorporates. Free Connected Limited is based in Hong Kong and has business records.

Family C VPN providers, their respective apps, and respective download count on the Google Play Store

Provider Name	***VPN Name***	***Google Play Downloads (millions)***
Fast Potato Pte. Ltd	VPN PotatoVPN - WifiProxy	10
Free Connected Limited	X-VPN	50

4.3.1 Signatures

We found multiple patterns across these providers, including a shared, proprietary protocol implementation, shared filenames and strings, and similarities in their code organization.

Protocols

Based on their packet captures, both apps appeared to use a custom tunneling protocol and make connections to servers on port 53 (DNS). None of the content in the application layer payloads was valid DNS and was instead obfuscated. This is possibly to bypass firewall rules permitting DNS on port 53.

Code Signatures

The decompiled code of these two VPNs is structurally and functionally similar. Both contain the same hard-coded, preshared key value in the variable psk. Both import the same shared library, libxjp6xdkbew.so. This library is the source of a second psk value. Both load the library in the same class named wcyybbcujkCs. Both utilize identically named resource files, assets/jsd5xcyjr5w587dk5usn, although their exact contents vary. Neither the name of the shared library nor the asset file yield results when searching VirusTotal, Google, Brave, GitHub, Reddit, Twitter, or other sources, indicating that these names are specific to these VPN clients.

Defense Mechanisms

Both apps implement the same obfuscation and anti-reverse engineering countermeasures. At the Java level, there is minimal code outside of imports for advertising and analytics. Most of the functionality is implemented in native code. At the native code level, each contains the single, shared library mentioned above (libxjp6xdkbew.so). The tunnel functionality is contained within this library along with anti-reversing countermeasures.

4.3.2 Security

Both applications are susceptible to connection inference attacks using the client-side blind-in/on-path attacks. We localized the issue to the applications' dependency on Netfilter to reroute packets through the tunnel (TUN) interface, as well as their dependency on libredsocks.so. We tested each application to confirm by using the client-side blind-in/on-path attack method outlined in Section 2.1.2.

4.4. Other VPNS

The "Other" group of providers consists of VPN Super Inc., Miczon LLC, and Secure Signal Inc., which distribute VPN Super Inc, TetraVPN, and Secure VPN - Safer Internet, respectively. They appear to have no links to other VPNs. We briefly summarize their most notable characteristics below.

VPN Super Inc.

The application does not appear to employ much obfuscation beyond ProGuard, and there are no anti-reverse engineering countermeasures present. The application supports OpenVPN and IPsec (IKEv2). We tested both and were able to extract memory dumps. This confirms that there appear to be public keys and configuration parameters consistent with these protocols.

TetraVPN

TetraVPN has no anti-reversing countermeasures, except for ProGuard obfuscation, and contains no assets. Only the WireGuard protocol is supported, which contains configuration files that are shared by anyone who downloads the app, at res/raw/uk_client.conf and res/raw/japan_client.conf.

Secure VPN - Safer Internet

This application uses ProGuard obfuscation. The core functionality is implemented in native code in libchannel.so, which is obfuscated.

5 DISCUSSION

5.1 Security and privacy concerns and their impacts

We identified three classes of problematic security and privacy issues with varying impacts on users. Firstly, the undisclosed location collection is a major violation of user trust and privacy given the providers explicitly stating that they do not collect such information. Secondly, the client-side blind-in/on-path attacks allow an attacker to infer with whom a VPN client is communicating. Thirdly, and most critically, on many of the VPNs we analyzed, a network eavesdropper between the VPN client and VPN server can use the hard-coded Shadowsocks password to decrypt all communications for all clients using the apps. These weaknesses nullify the privacy and security guarantees the providers claim to offer. These issues are even more concerning when—in addition to the fact that the providers in Family A appear to be owned and operated by Chinese company Qihoo 360— all providers in Families A and B have gone to great lengths to hide this fact from their combined number of 700 million-plus users. Furthermore, three of the providers in Family B explicitly state they are operated from Hong Kong. While we could not link them directly to Qihoo 360, the fact that their privacy policies explicitly cite Innovative Connecting suggest they have links to a Chinese national, at minimum.

The issues we identified affect users, providers, and app stores. At the very least, VPN users who value privacy should avoid using Shadowsocks, including the apps from these developers, as Shadowsocks was not designed to facilitate privacy, but merely censorship circumvention.[47] App store operators, like Google, face major challenges identifying and verifying the ownership

[47] See footnote 16.

of apps on the Play Store, as well as ensuring Play Store apps are secure. Ownership identity verification and app security auditing is currently labor intensive and would require sophisticated, automated tools to achieve at scale. Google currently offers a security audit badge for VPN apps. Whether a similar badge for verified identity makes sense is debatable, because there are valid reasons why a VPN provider might not want to reveal that information. For one, it could expose them to legal or digital attack from a country or entity that opposes VPNs. Finally, VPN providers should either avoid offering Shadowsocks to users, or carefully explain the risks. The Shadowsocks protocol has no built-in asymmetric cryptography and requires the insecure use of hard-coded passwords, from which symmetric keys are deterministically derived. Alternatively, VPN providers should devise and implement a system for the secure distribution of these passwords. Prior work has found that home-rolled cryptographic systems commonly contain major flaws.[48],[49],[50] Thus, if not devised and maintained by experts, such a password distribution system would be liable for the introduction of additional security issues, and it may increase a user's vulnerability to network censorship if not carefully implemented.

We have several unresolved questions based on our findings. First, it is unclear why a single entity would split their user base across multiple distinct providers and multiple VPN apps the way these providers appear to have done. One assumption is that doing so isolates reputational damage to a single provider. Second, it is unclear why they did not also segment their code, credentials, or server infrastructure considering they went through the trouble of segmenting the legal side of their business. Simple possible explanations include cost, or ease of execution, or management. It costs less to pay one developer to make one app and repackage it than to pay for multiple different apps. It is also easier and cheaper to have multiple apps use the same credentials and infrastructure, since the provider only needs to pay a single team to manage that infrastructure. Third, it was counterintuitive to find deprecated ciphers and hard-coded passwords in these apps, given that they are security-sensitive apps. In particular, why would there be such serious security and privacy issues among the VPNs whose providers are owned by Qihoo 360, a leading Chinese cybersecurity firm? We do not aim to imply that these VPNs were intentionally developed insecurely or for malicious purposes, but it is important to highlight the challenges of developing secure software—even for cybersecurity experts.

5.2 Recommendations for VPN App Users and Distributors

[48] Jeffrey Knockel, Adam Senft, and Ronald Deibert. 2016. Privacy and Security Issues in BAT Web Browsers. In 6th USENIX Workshop on Free and Open Communications on the Internet (FOCI 16). USENIX Association, Austin, TX, 7 pages. https://www.usenix.org/conference/foci16/workshop-program/presentation/knockel.

[49] Jeffrey Knockel, Mona Wang, and Zoë Reichert. 2023. "Please do not make it public": Vulnerabilities in Sogou Keyboard encryption expose keypresses to network eavesdropping. Technical Report. The Citizen Lab.

[50] Jeffrey Knockel, Mona Wang, and Zoë Reichert. 2024. The Not-So-Silent Type: Vulnerabilities in Chinese IME Keyboards' Network Security Protocols. In Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (Salt Lake City, UT, USA) (CCS '24). Association for Computing Machinery, New York, NY, USA, 1701–1715. https://doi.org/10.1145/3658644.3690302.

If possible, users should avoid using Shadowsocks from these particular providers (unless and until they implement mitigations)—if not from Shadowsocks operators at large. Hard-coded Shadowsocks passwords counterproductively increase the exposure of users' communications to eavesdroppers compared with using no VPN at all. Shadowsocks was designed to be censorship resistant, not private. Therefore, VPN providers need to be upfront with their users about the risks if they use Shadowsocks with only symmetric cryptography.

App stores like the Play Store are in a challenging position given the scalability limitations around vetting developers and identifying software with misleading security properties in their store. Google currently offers a security audit badge for VPN apps. Making such a badge mandatory for VPN apps and offering an identity verification badge for developers who go through an identity verification process might provide users additional information and protection. Google is potentially exposing its brand to reputational damage by hosting and profiting from deceptive and insecure apps such as the ones we investigated.

While increased requirements for identity verification may help to keep users safe, such requirements must be balanced with the rights of developers to anonymously distribute software. In the censorship circumvention space, in particular, developers can be especially at risk of legal jeopardy and transnational repression.[51] However, while we recognize the importance of developer anonymity, we also note that anonymity is distinct from deception, and software distributors could respect authors' anonymity while still taking action against those who have misrepresented their corporate associations.

About the Authors and Project Funding

About the Authors

Benjamin Mixon-Baca is a PhD student at Arizona State University (ASU). He has been working in the security space for ~10 years in various capacities. Early in his career, Mixon-Baca served as an Information Controls Fellowship Program (ICFP) fellow with Open Technology Fund (OTF). During this time, he worked with The Citizen Lab helping NGOs defend against targeted threats by deploying and maintaining intrusion detection systems and analytics software. In 2016, he held a seasonal fellowship at ICSI where he worked on developing Zeek scripts to fingerprint Tor based on byte patterns of the TLS four-way handshake. Mixon-Baca finished his master's degree in computer science in 2017.

From 2017 to 2020, he worked in the private sector leading different research efforts. This included automated attack frameworks similar to the Metasploit framework; machine learning projects to rank vulnerabilities from most to least exploitable using open source intelligence from the national vulnerability database (NVD), Twitter, and similar. At the end of 2019, Mixon-Baca

[51] gfw report. 2023. Many Popular Censorship Circumvention Tools Deleted or Archived since November 2, 2023. https://github.com/net4people/bbs/issues/303. Accessed on: June 20, 2025.

and his colleagues from the University of New Mexico founded Breakpointing Bad, a nonprofit focused on internet freedom research.

In spring 2020, he left the private sector to pursue his PhD and transferred to ASU under the mentorship of Dr. Jedidiah Crandall. His current research has focused broadly on developing attacks and defenses against computer systems. During this time, he and Dr. Crandall developed multiple attacks against VPNs, called Network Alchemy. These attacks allow the attacker to place himself between a VPN client and server from an initially off-path position, break the VPN anonymity, or reroute VPN client packets to an attacker. From 2021-2022 he worked as an OTF ICFP fellow with the University of Michigan where they developed CryptoSluice, a tool for automatically identifying weak and unencrypted traffic at scale in an ethical way. CryptoSluice allows an analyst to process real network traffic at high speed and generate a list of candidate applications that an analyst can reverse engineer while protecting the privacy of the network being analyzed. Mixon-Baca conducted the research presented in this report during his third fellowship with OTF.

Contact Information:
Email: ben@breakpointingbad.com
Signal: bmixonbaca.22

Dr. Jeffrey Knockel is an assistant professor at Bowdoin College Knockel is also a member of Breakpointing Bad.

Dr. Jedidiah R. Crandall is an associate professor at Arizona State University, in the Biodesign Center for Biocomputation, Security and Society and the School of Computing and Augmented Intelligence. He has been fighting for Internet freedom through teaching, outreach, and research for nearly two decades. This includes Internet measurements, reverse engineering, and attacks on low-level network tracking and routing primitives (e.g., as used by VPNs).

About the Funder

Open Technology Fund (OTF) is an independent nonprofit organization committed to advancing global Internet freedom. OTF supports projects focused on counteracting repressive censorship and surveillance, enabling citizens worldwide to exercise their fundamental human rights online.

This project was funded by OTF's Information Controls Fellowship Program, which supports examination into how governments in countries, regions, or areas of OTF's core focus are restricting the free flow of information, impeding access to the open internet, and implementing censorship mechanisms—thereby threatening the ability of global citizens to exercise basic human rights and democracy. Work focused on mitigation of such threats is also supported. The program supports fellows to work within host organizations that are established centers of expertise by offering competitively paid fellowships for three, six, nine, or twelve months in duration.

Appendix: Supplementary Material on Methodology

A.1 Business Analysis

A.1.1 Search Terms

List of search terms built through static and dynamic analysis and used while researching providers.

Family	Search Term	Term Location
A	libxjp6xdkbew.so	lib/arm64-v8a
A	jsd5xcyjr5w587dk5usn	assets
A	XgS6aaUmMB0WtOkrzrTr/5-S1jSxiFu-pCAZan12C/pvsm0pA_cUa7oAzrYvGl/uJ9H4qB7ojPP2slohjpM	libxjp6xdkbew.so
A	z93emquwpbdk9qvqfkfc8z552vaf52szsvzbmvd6qjdynmxm7yh6nq23c9yw4drs	libxjp6xdkbew.so
A	kshpqn53ps	libxjp6xdkbew.so
B	aaa_new.png	asset directory
B	b1bbceaffd6c52a2	asset directory
B	cert.pem	asset directory
B	proxy.builtin	asset directory
B	server_offline.ser	asset directory
B	libopvpnutil.so	shared library
B	free.vpn.unblock.proxy.turbovpn	lib/arm64-v8a
B	free.vpn.unblock.proxy.vpnmaster	lib/arm64-v8a
B	free.vpn.unblock.proxy.vpnpro	lib/arm64-v8a
B	free.vpn.unblock.proxy.vpn.master.pro	lib/arm64-v8a
B	free.fast.vpn.unlimited.proxy.vpn.master.pro	lib/arm64-v8a
B	free.vpn.unblock.proxy.freenetvpn	lib/arm64-v8a
B	free.vpn.unblock.proxy.vpnmonster	lib/arm64-v8a
B	free.vpn.unblock.fast.proxy.vpn.master.pro.lite	lib/arm64-v8a
B	free.vpn.unblock.proxy.turbovpn.lite	lib/arm64-v8a
B	unlimited.free.vpn.unblock.proxy.supernet.vpn	lib/arm64-v8a
B	43a41a300d06092a864886f70d01010505003	lib/arm64-v8a
C	libcore.so	lib/arm64-v8a
C	co.infinitevpn.free.proxy	libcore.so
C	com.free.neo.vpn	libcore.so
C	com.free.turbo.unlimited.touch.vpn	libcore.so
C	com.free.unblock.melon.vpn	libcore.so
C	com.free.unlimited.lemon.vpn	libcore.so
C	com.smart.nord.global.vpn	libcore.so
C	com.vpnbottle.melon.free.unblock.fast.vpn	libcore.so

C	com.vpncapa.vpnmaster.free.unblock.vpn	libcore.so

A.1.2 Business Filings

We searched for VPN provider names in the OpenCorporates database.

A.1.3 Social Media

We searched for and noted VPN providers and apps that had social media profiles on Facebook, Instagram, Twitter/X, Reddit, Telegram, Discord, and YouTube. We also used the search term list, which produced minimally useful results in terms of associating distinct providers together. One interesting interaction we found, depicted in Figure 5, shows a user comment on Turbo VPN's Discord channel. In it the user specifically claims they (Turbo VPN) "work for the CCP"[52] — they provide no supporting evidence for this claim. In terms of transparency, a malicious VPN provider, especially a sophisticated, well-resourced one, would likely create a social media footprint to appear legitimate and throw off investigators.

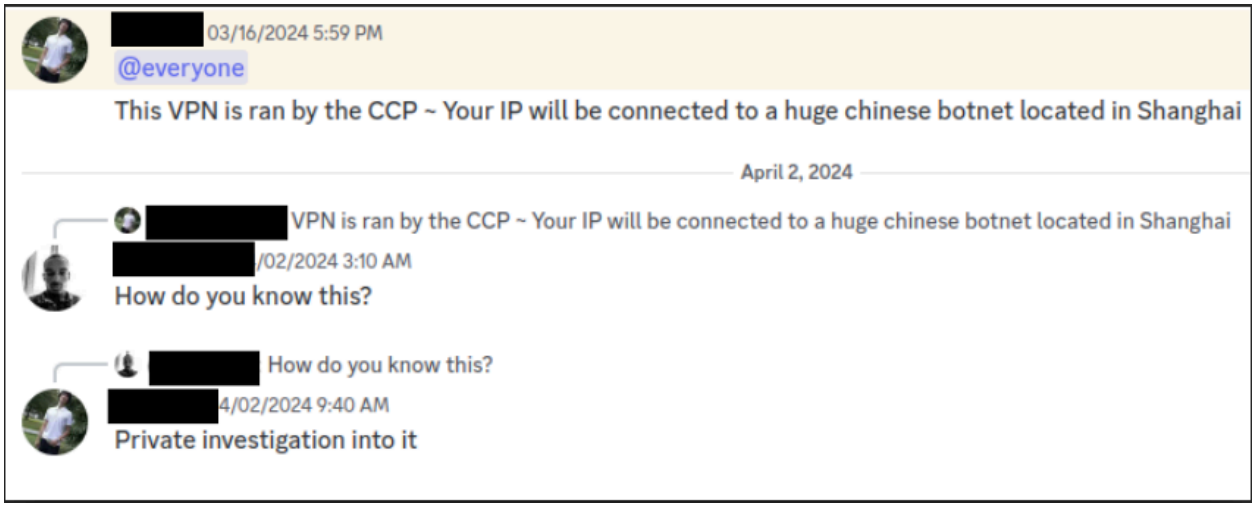

Figure 5: Discord user alleging Turbo VPN is part of a Chinese Communist Party (CCP)-controlled botnet originating in Shanghai.

A.1.4 DNS Records

We collected DNS records using dig and WHOIS. In all cases, the VPN providers either redacted all information from their WHOIS records or used a domain registrar, such as Domain Protection Services or Domain by Proxy, to anonymize this information. Domain-related

[52] The assumption is that the user is referring to the Chinese Communist Party.

information was largely not useful for making associations between distinct VPN providers. In only one case, the email address “liaoymo2015@gmail.com” was present in Start of Authority (SOA) records for Innovative Connecting’s Turbo VPN and Lemon Clove’s VPN Proxy Master. This email was also identified by VPN Pro’s report.[53]

A.1.5 Code Repositories

We searched Github, Gitlab, and Gitee using search terms. We were unable to locate repositories for VPN providers or application code in all cases. We did periodically find other researchers had uploaded results from tools like automated analysis tools, or references to libraries in the APK, but all of the VPNs appeared to be closed source.

[53] See footnote 2.