Fingerprinting VPNs with Custom Router Firmware: A New Censorship Threat Model

Sultan Almutairi*‡, Yogev Neumann*, Khaled Harfoush**
*Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC
**Department of Computer Science, North Carolina State University, Raleigh, NC
‡Department of Computer Engineering, Qassim University, Saudi Arabia
{ssalmuta, yneuman, kaharfou}@ncsu.edu

***Abstract*—**
Virtual Private Networks are effective in bypassing Internet censorship. Extensive research has been done to obfuscate VPN traffic in order to circumvent control filtering. In this paper, we introduce a new threat model in which a censorship body can use home routers running a custom firmware or an embedded OS to identify VPN connections. Monitoring network traffic at home routers enables efficient and accurate fingerprinting of VPN traffic before the traffic is NATed to the Internet. The proposed model leverages a vulnerability in VPN implementations. Experimental results highlight its ability to fingerprint with negligible false positives/negatives. The purpose of the study is to increase awareness of this issue and inspire others to take this threat model as a reasonable risk that needs to be addressed.

***Keywords*—Virtual Private Networks, Traffic Fingerprinting, Censorship**

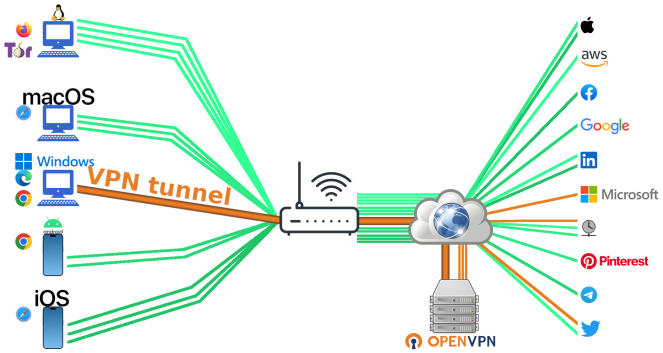

Fig. 1. An end-device on a home network using a VPN will have all its traffic destined to the VPN server.

I. Introduction

Virtual private network (VPN) services have become increasingly popular as means of hiding Internet activity, particularly in contexts where Internet censorship and surveillance are prevalent. A VPN tunnels network activity via a secured channel to a VPN server and from the VPN server to the requested websites or services. VPN services are used not only by journalists or activists who may be vulnerable to monitoring or targeting, but also by general public who value their online privacy while browsing the web over public WiFi networks, connecting remotely to their workplace services, or to access content/services that may be censored in certain parts of the Internet. As VPNs have become more widely used, censorship bodies such as states seeking to control and restrict Internet access that are often motivated by moral or religious values have invested heavily in developing and deploying sophisticated censorship technologies to prevent the dissemination of information, suppress challenges to their official narrative, maintain authority, and prevent disruptions in society [1], [2].

A typical way people connect to the Internet is through *home routers*. Home routers act as gateways and provide local devices/peripherals a convenient and cost-effective way to access the Internet. However, home routers can also be a point of vulnerability due to security and privacy breaches if they are not properly secured or maintained. Attackers can gain control of private home routers by using a backdoor built into the router [3] or by exploiting a vulnerability in the router's code [4], [5]. Home routers are often provided for free with Internet contracts and come with a customized firmware or an embedded operating system. While this can be convenient for users, it also means that home routers may not be trusted especially if they have not been tested by a third party.

In this paper, we introduce a new threat model in which a censorship body aiming to fingerprint VPN traffic does so by leveraging custom home router firmware using a simple statistical model. The proposed approach is simpler, more efficient and more accurate compared to existing VPN fingerprinting techniques such as Machine Learning (ML) algorithms [6]–[8]. VPN fingerprinting at home routers further offers many advantages. Besides distributing the processing load on home routers rather than relying on a centralized infrastructure, it overcomes the widely used obfuscation techniques that are used to avoid detection using Deep Packet Inspection (DPI) [9], [10] and overcomes the obfuscation caused by multiple devices communicating behind a single IP address (NAT box). It also enables real-time detection of VPN connections. The simplicity of the calculations needed at home routers adds negligible overhead.

The key idea behind our threat model is that end devices using a VPN connection will, by default, send all their traffic to the same destination (the VPN server) identified by its public IP address – Refer to Figure 1. On the other hand, non-VPN traffic is typically sent to a mix of different destinations; e.g. websites, weather widget, OS update server, etc. We tested the identified threat model in realistic setups. Experimental results

highlight its ability to detect VPN connections with negligible false positives/negatives. To the best of our knowledge, this threat model was not considered in the literature. The purpose of our study is to increase awareness of this issue and inspire others to take this threat model as a reasonable risk that needs to be addressed.

The rest of this paper is organized as follows: In section II, we describe the threat model. In section III, we explain the design of the fingerprinting algorithm. In section IV, we evaluate the threat model in realistic setups and report on our findings. In section V, we compare our work with other research on the subject. We finally conclude in section VI.

II. Threat Model

Our threat model relies on fingerprinting VPN traffic at home routers due to their ability to access user traffic before it is Network Address Translated (NATed), allowing for tracking the traffic of each device individually.

The feasibility of this attack can arise from exploiting vulnerabilities in the routers, by modifying the firmware of the routers, or even during the common practice of provisioning home routers through specialized servers managed by Internet Service Providers (ISPs). These servers possess the capability to distribute updates to these routers [11]. Even the supply chain of home routers can be tampered with by censorship bodies without the user's knowledge [12].

Our threat model assumes that alterations implemented in the user's home router are solely for *passive* traffic analysis purposes and do *not* involve any active modifications of the traffic in order to prevent raising suspicion. Also, given the typical limited capabilities of home routers, the attack should not require substantial computational power in order to avoid deteriorating performance.

The proposed threat compromises a common design *vulnerability* in major VPN implementations in which all user traffic is tunneled to a *single* public IP address. The home router can track the packets exchanged between *any* device on the local network and a public IP address over time to fingerprint VPN connections. We next provide the details of our proposal.

III. Design

Our fingerprinting technique involves tracking contacted public IP addresses over time from *each* internal (home) device. Our technique relies on the notion of a **session**. A session for a home device is a duration of time in which the home device is exchanging packets with the *same* public IP address. A new session is initiated (and the old one terminated) when a new public IP address for the device is detected at the home router. The home router counts the number of packets, $PACKETS\_COUNT$, in each device session, and if the count exceeds a threshold, T, a potential VPN connection is reported.

Figure 2 illustrates the fingerprinting idea. It displays the $PACKETS\_COUNT$ for consecutive sessions of a device over time. In this example, A VPN connection was established during the last session leading to a large

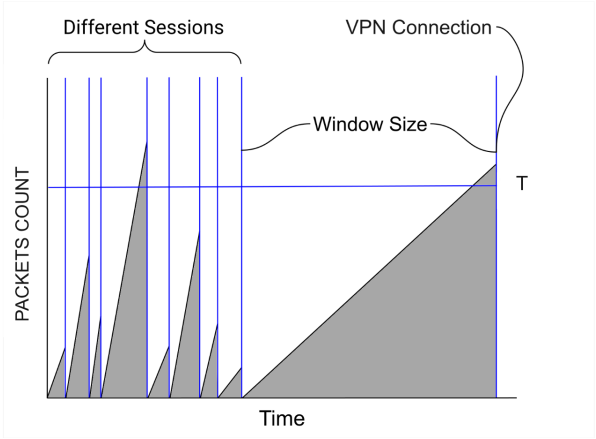

Fig. 2. Visual Representation of Fingerprinting VPN Connections.

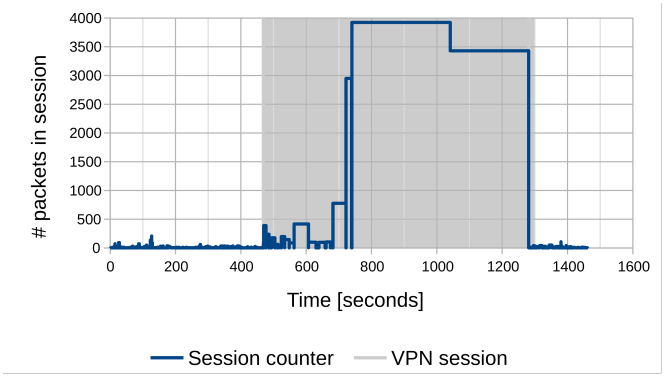

Fig. 3. VPN Fingerprinting from an actual packet trace. Dark area corresponds to active VPN connection.

$PACKETS\_COUNT$, as all device traffic is headed to the same VPN server (with the same public IP address), exceeding T and leading to a VPN connection being reported. However, note there is another session (the fourth session in the Figure), which does not correspond to a VPN connection, but still has $PACKETS\_COUNT$ exceeding T. This motivates the need to observe $PACKETS\_COUNT$ exceeding T *only* for sessions after exceeding a certain amount of time, which we refer to as the $WINDOW$. The choice of the threshold, T, and the value of the $WINDOW$ are important for an accurate fingerprinting, and should strike a balance between the ability to detect short VPN connections while minimizing false negatives (miss-classifications) and false positive reports. We evaluate the impact of the different choices in section IV.

Figure 3 displays the $PACKETS\_COUNT$ for consecutive sessions of a device over time from an actual trace of packets collected at a home router. The dark area corresponds to an active VPN connection. An important point to note is that the $PACKETS\_COUNT$ value does *not* start increasing once the VPN becomes active as the home device may still have open connections that are not using the VPN tunnel. This leads to packets, which are not destined to the VPN server, being noticed at the home router and to the creation of

new sessions. This *leakage* behavior delays the detection of VPN connections to some extent and highlight the need for a relatively large $WINDOW$. Although this is not a problem for long sessions, short VPN sessions can slip through the proposed detection method. After the leakage is resolved, the router creates a session for the VPN server and counts the packets until they reach the time window, which is set at 300 seconds in this example. The router checks the packet count and reports the IP address as belonging to a VPN server. It then starts a new session. In this trace, the VPN connection lasts for more than the time window but does not complete another window before being terminated. Notably, Our model is not bound by specific VPN protocol specifications, granting it the ability to identify a broad array of VPN protocols for comprehensive detection.

Note that the home router doing the fingerprinting does not have to maintain much information nor conduct much processing. Basically the router maintains for *each* home end-device (identified by its private IP address) the following information about the device *latest* session: (1) timestamp reflecting the time the session started, (2) session public IP address, (3) $PACKETS\_COUNT$ for the session. When a new packet from the home device is received at the router, the router checks the public IP address and if it is new IP, a new session is created (updating timestamp and public IP address, set packet count to zero). Otherwise it increment $PACKETS\_COUNT$ and then check algorithm metric: the window and $PACKETS\_COUNT$.The algorithm has an additional feature called "overlapping windows" that helps reduce sensitivity to the choice of metrics. This feature considers packet counts for two complete consecutive windows that have the same public IP address. If they satisfies an overlapping threshold, it flags the connection as a VPN.

The pseudo-code in Algorithm 1 and 2 provides the logic of the proposed fingerprinting algorithm applied at the home router. It relies on the $H[src]$ data structure storing the details of the session associated with home device with IP address src, where $H[src].dst$, $H[src].ts$ and $H[src].count$ represent the destination IP address, timestamp and $PACKETS\_COUNT$ values of the session, respectively. Note that the pseudo-code considers both ingress and egress traffic of the home network. A VPN connection is reported only if both (1) the session time exceeds $WINDOW$ amount of time, and (2) the session $PACKETS\_COUNT$ exceeds a threshold T.

While the fingerprinting can report a suspected VPN connection, verifying whether it is actually a VPN connection can be done by the back-end infrastructure of the censorship body, for example through IP probing as performed by Diwen Xue et al. in [13]. Our code with compilation instructions is publicly available in a git repository [14].

Algorithm 1 Analyze new packet

```
1:  procedure AnalyzeIP(src, dst, ts)
2:      if not is_internal(src) then                    ▷ Ingress
3:          src, dst ← dst, src                         ▷ Flip
4:      if is_internal(dst) then                        ▷ Internal communication
5:          return
6:      if src ∉ H then                                 ▷ Initial record
7:          H[src] ← [0, 0, 0, 0]
8:      if H[src].dst ≠ dst then                        ▷ New connection
9:          H[src].dst ← dst
10:         H[src].ts ← ts
11:         H[src].count ← 0
12:     H[src].count ← H[src].count + 1
13:     if ts − H[src].ts > WINDOW then
14:         if IsSuspectedVPN(src) then
15:             Report(src, dst, ts)
16:         H[src].ts ← ts
17:         H[src].count_prev ← H[src].count
18:         H[src].count ← 0
```

Algorithm 2 Decide if a source IP is suspected as a VPN

```
1:  procedure IsSuspectedVPN(src)
2:      if H[src].count > T then
3:          return True
4:      if H[src].count_prev + H[src].count
          > WINDOW_OVERLAP then
5:          return True
6:      return False
```

IV. Experimental Results

We evaluated the threat model (1) on a home router in a realistic setup that mimics a conventional home network consisting of a variety of end devices and (2) on a playback of an IoT dataset to gain deeper insights into the behavior of IoT devices. We next provide the experimental details and discuss the results.

We implemented our statistical model as a module in the Linux kernel of the NETGEAR R6120 **router**, which has 580 MHz processor, 16 MB of ROM, and 64 MB of RAM. This router was chosen due to its modest hardware capabilities in order to demonstrate that our statistical model is capable of operating on weak routers. We changed the original firmware of NETGEAR router to OpenWrt [15] which is an open-source operating system for networking devices that allows users to customize and extend their functionality.

In our experiments, we tested a variety of **devices**, which were divided into three main groups: (1) computers, (2) cell phones, and (3) IoT. We also tested different Operating systems on computers and cell phones. Using this conventional setup, we conducted two types of experiments: **Controlled** and **uncontrolled** as we detail next.

A. Controlled Tests

Prior to evaluating our statistical model, we conducted a **controlled test** where a single device within the home network performed a predefined activity for a limited duration. The objective of this test was to gather statistical data regarding the characteristics of the observed sessions. This data was then used to determine appropriate values for the parameters

TABLE I
CONTROLLED SETUP RESULTS SUMMARY

Category	Device	Activity	Measurement Duration	Number Sessions	Average Session [sec]	Longest Session [sec]	# Packets In Longest Session
Computers	Linux OS (Ubuntu 22.04)	Idle	20 minutes	582	1.4622	710	2
		Streaming	20 minutes	651	1.9416	27	3,709
		Surfing	20 minutes	28175	0.0439	8	2
		Downloading	20 minutes	1149	0.9965	238	487,350
	Mac OS (MacBook MacOS Big Sur)	Idle	20 minutes	388	2.5515	486	2
		Streaming	20 minutes	1,612	0.7593	41	40
		Surfing	20 minutes	17,306	0.0682	105	247
		Downloading	20 minutes	1,242	0.9686	185	14,168
	Windows OS (Windows 10)	Idle	20 minutes	252	4.2937	79	10
		Streaming	20 minutes	9,577	0.1326	10	6
		Surfing	20 minutes	33,387	0.035	8	10
		Downloading	20 minutes	4,070	0.3091	20	29,380
Cell phones	Android (Galaxy Note 10)	Idle	40 minutes	17,328	0.1405	147	6
		Steaming	30 minutes	17,012	0.1099	19	2
	iOS (iPhone Pro max 13)	Idle	40 minutes	278	8.7842	449	2
		Streaming	30 minutes	2,067	0.8684	18	136
IoT devices	Printer (HP LaserJet M402dw)	Idle	40 minutes	10	172.4	332	14
		sending commands	20 minutes	5	241.2	302	11
	Smart Bulb (Merkury)	Idle	40 minutes	11	218.9091	301	16
		Sending commands	30 minutes	6	230.8333	301	73
	Smart TV (Samsung TU7100D)	Idle	40 minutes	40	21	111	9
		Streaming	30 minutes	1,278	1.4108	34	90
	Vacuum cleaner (iRobot 980)	Idle	20 minutes	4	243	328	18
		Sending commands	15 minutes	3	201.3333	303	94
	WiFi Camera (Merkury)	Idle	40 minutes	14	164.2857	316	48
		Streaming	40 minutes	17,892	0.1373	1	328

$WINDOW$ and T, capable of avoiding mis-classification of non-VPN traffic as VPN traffic. The **predefined activities** for this controlled test were (1) streaming, (2) idle state, (3) web browsing, (4) file downloading, and (5) sending commands. Table I reports on the number of sessions observed over a period of time, the average session time, the longest session time and the number of packets in the longest session in each tested scenario. For example, for an idle Windows 10 operating system, we observed 252 sessions over the course of 20 minutes. An important observation based on the results is that many IoT devices lead to a small number of sessions compared to the non-IoT devices. This highlights the fact that non-IoT devices typically connect to multiple IP addresses in the absence of VPN connections, while IoT devices connect with a small number of IP addresses even in the absence of VPN connections. Based on these characteristics, IoT devices may be more challenging to our algorithm. In order to further study the behavior of IoT devices, we conduct an analysis on an IoT dataset in the subsequent section.

B. IoT Dataset

We applied our statistical model to the CIC IoT dataset [16], which was collected in 2022 at the Canadian Institute of Cybersecurity. The dataset consists of packet traces of IoT devices that were recorded while researchers were freely using these devices. Specifically, we checked the active traces for a variety of devices such as cameras, smart lights, speakers, doorbells, door/window sensors, base stations, motion sensors, actuators, multi-sensors, weather stations/sensors, smart hubs, smart plugs, smart coffee makers, smart TVs, and smart boards. The dataset did not have any VPN connections. Figure 4 displays the misclassification percentage for different combinations of $WINDOW$ and Threshold T values, obtained through an analysis of the dataset.

C. Choosing WINDOW and T

The selection of appropriate values for the $WINDOW$ and T parameters requires a careful balance. A window that is overly large might overlook legitimate VPN sessions, while one that is too small could lead to an increased number of false positives. Based on the sessions characteristics provided in Table I and the insights from Figure 4, we recommend considering $WINDOW$ ranging from 5 to 50 minutes. While several combinations are plausible, our final decision was to set the $WINDOW$ to 300 seconds and the T parameter to 500 packets. Upon applying this configuration to the IoT dataset, we observed that out of all identified sessions, only two devices were mis-classified as having VPN sessions. The

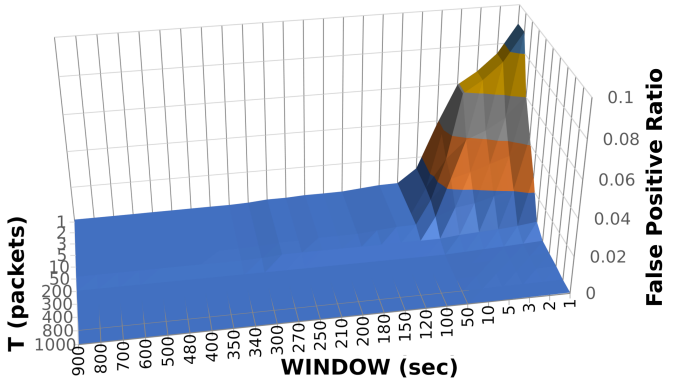

Fig. 4. False Positive Ratio for the CIC IoT dataset.

first device, a Google Nest Cam, connected to the Google cloud server at $nexusapi - us1.dropcam.com$. The second device communicated with the Alibaba cloud server. The primary challenge is classifying IoT devices that generate significant traffic during a single prolonged session, as is the case with WiFi cameras. These mis-classifications can be effectively addressed at the censorship body.

D. Uncontrolled Tests

We tested our algorithm with these configuration in **uncontrolled test** spanning four days, in which we freely interacted with several devices, including a Smart TV, WiFi camera, smart bulbs, Windows PC, and Android and iOS smartphones. During this test, we intentionally introduced VPN connections using different VPN providers, and used these connections to download content. We tested VPN connections by different **VPN providers** such as ProtonVPN [17], Hide.me [18], Turbo VPN [19], KasperskyVPN [20], Hotspot Shield [21], Secure VPN [22], Fast VPN Pro [23], VPN Super [24], and VPN Gate [25]. The providers were selected based on their popularity in the Google Play Store. Table II provides the results for each VPN provider. The router was able to successfully flag all VPN connections of 7 out of the 9 that were tested. We analyzed the applications that were not detected, and we found that Hotspot Shield has traffic leakage where not all traffic tunneling via the secure channel as was reported by Khan *et al*. [26]. KasperskyVPN was tunnelling all traffic to a single server; however, the client was exchanging packets with a second server every 300 seconds, and since this is our time window the VPN connection was not detected, in order to detect KasperskyVPN traffic the window size should be less than 300 seconds. Throughout our uncontrolled experiment, we did not encounter any false positives among the tested devices.

V. Related Work

Extensive efforts were aimed at fingerprinting VPN traffic for censorship. These can be classified based on whether they rely on Deep Packet Inspection (DPI) [13], [27], [27]–[29] or Machine Learning techniques [6]–[8].

TABLE II
Uncontrolled Setup Results for VPN Providers.

App name	Downloads	Result
ProtonVPN	10M	Detected
Hide.me	1M	Detected
Turbo VPN	100M	Detected
KasperskyVPN	100M	Undetected
Hotspot Shield	100M	Undetected
Secure VPN	100M	Detected
Fast VPN Pro	50M	Detected
VPN Super	50M	Detected
VPN Gate	N/A	Detected

Deep packet inspection (DPI) is a method of examining packets passing through a network to identify its type [29]. The authors in [27] [28] study Russia's national firewall, Sovereign RuNet, and its censorship capabilities. They find that Russia deployed DPI devices in a decentralized way close to end users, allowing fine-grained control over privately-owned ISPs but the censorship was centrally coordinated. Researchers in [27] also suggested that deploying censorship devices close to end users is more effective. Our threat model involves censorship fingerprints located inside home routers, distributed in a decentralized manner. The authors in [13] show that fingerprinting OpenVPN is feasible and practical at scale for censorship states. They use DPI to fingerprint VPN connections during the handshake and then probe servers to confirm by eliciting protocol-specific behavior that reveals the OpenVPN server. Their implementation in the core network requires high computational power, while our threat model operates in the home network without the need for DPI and with a much simpler model that requires significantly less computational power. Additionally, our model is capable of fingerprinting VPN connections across widely used VPN protocols.

Along another dimension, studies have been performed to overcome the limitations of DPI using machine learning (ML) algorithms. ML algorithms extract the statistical features or characteristics extracted from the packet flow. In [6], Sikha Bagui *et al*. compare the performance of 6 ML algorithms to detect VPN traffic. In [7], Shane Miller *et al*. use a multi-layer ANN to detect VPN connections. In [8], Lulu *et al*. compare CAE and CNN in their ability to fingerprint VPN traffic. In contrast to our model, which requires lower computational resources, these methods are costly and typically require higher computing power. In addition, obfuscation techniques could potentially undermine the efficacy of these methods, making them less reliable compared to our model.

In [30], the authors exploit vulnerability in UNIX-like operating systems based on the weak host model to determine if a connected user is using VPN and make inferences about the websites they are visiting, and inject data into the TCP stream. In contrast, our research is passive in nature, aimed at fingerprinting and blocking VPN connections through censorship measures, rather than actively injecting data into the network.

Many efforts aimed at evading censorship. In [31], the

obfsproxy was introduced to encrypt traffic to disguise the true nature of the traffic, making it difficult for censors to detect and block. The OpenVPN protocol has adopted this technique to aid users in circumventing censorship efforts [32]. In [33] a circumvent tool is proposed using volunteers to provide a VPN relay service using various VPN protocols. Our statistical model can identify VPN connections offered by these tools effectively at scale as their VPNs tunnel all user traffic through a single server.

VI. Conclusions and Future Work

In this study, we introduce a new unexplored threat model to user privacy and propose a simple but effective technique where a censorship body seeking to censor cyberspace by blocking uses' access to VPN servers can do so by leveraging the mission into home routers. We tested the technique using network traffic datasets and by conducting controlled and uncontrolled experiments mimicking a small house network. We were able to achieve a 100% detection rate for all VPNs that connect to a single VPN server.

In order to counter the threat, we propose widespread adoption of traffic splitting, so that not all traffic is tunneled through one secure channel. This helps to neutralize network traffic. Additionally, we recommend not keeping a VPN session for an extended period of time, as VPN tunnels can switch between VPN servers at random time intervals. In conclusion, our study aims to raise awareness of the issue and encourage others to take the threat model seriously.

References

[1] A. Thompson, "Buying Silence: The Price of Internet Censorship in China," Jan. 2021.

[2] B. Toulas, "Russia's 'Oculus' to use AI to scan sites for banned information," *BleepingComputer*, Aug. 2022. [Online]. Available: https://www.bleepingcomputer.com/news/security/russias-oculus-to-use-ai-to-scan-sites-for-banned-information/

[3] B. Meyer, "Walmart-exclusive router and others sold on Amazon & eBay contain hidden backdoors to control devices," *CyberNews*, Nov. 2022. [Online]. Available: https://cybernews.com/security/walmart-exclusive-routers-others-made-in-china-contain-backdoors-to-control-devices/

[4] L. Vaas, "Millions of Routers Exposed to RCE by USB Kernel Bug," *Threatpost*, Jan. 2022. [Online]. Available: https://threatpost.com/millions-routers-exposed-bug-usb-module-kcodes-netusb/177506/

[5] S. Gatlan, "Over 19,000 end-of-life Cisco routers exposed to RCE attacks," *BleepingComputer*, Jan. 2023. [Online]. Available: https://www.bleepingcomputer.com/news/security/over-19-000-end-of-life-cisco-routers-exposed-to-rce-attacks/

[6] S. Bagui, X. Fang, E. Kalaimannan, S. C. Bagui, and J. Sheehan, "Comparison of machine-learning algorithms for classification of VPN network traffic flow using time-related features," *Journal of Cyber Security Technology*, vol. 1, no. 2, pp. 108–126, Apr. 2017, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/23742917.2017.1321891.

[7] S. Miller, K. Curran, and T. Lunney, "Multilayer Perceptron Neural Network for Detection of Encrypted VPN Network Traffic," in *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, Jun. 2018, pp. 1–8.

[8] L. Guo, Q. Wu, S. Liu, M. Duan, H. Li, and J. Sun, "Deep learning-based real-time VPN encrypted traffic identification methods," *Journal of Real-Time Image Processing*, vol. 17, no. 1, pp. 103–114, Feb. 2020.

[9] admin, "Security Hardened OpenVPN Config with Traffic Obfuscation," Jan. 2020. [Online]. Available: https://hide.me/en/blog/security-hardened-openvpn-config-with-traffic-obfuscation/

[10] "Pluggable Transports." [Online]. Available: https://pluggabletransports.info/

[11] Many home routers supplied by ISPs can be compromised en masse, researchers say | PCWorld. [Online]. Available: https://www.pcworld.com/article/440767/many-home-routers-supplied-by-isps-can-be-compromised-en-masse-researchers-say.html

[12] J. Robertson and M. Riley, "The Long Hack: How China Exploited a U.S. Tech Supplier," *Bloomberg.com*, Feb. 2021. [Online]. Available: https://www.bloomberg.com/features/2021-supermicro/

[13] D. Xue, R. Ramesh, A. Jain, M. Kallitsis, J. A. Halderman, J. R. Crandall, and R. Ensafi, "OpenVPN is Open to VPN Fingerprinting," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 483–500. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/xue-diwen

[14] S. Almutairi and Y. Neumann, "VPN Fingerprinting at Home Router," Apr. 2023. [Online]. Available: https://github.com/xqgex/VPNFingerprintingAtHomeRouter

[15] R. Brown, "Welcome to the OpenWrt Project," Sep. 2016, section: 2022-09-05T19:03:56-04:00. [Online]. Available: https://openwrt.org/start

[16] Sajjad Dadkhah, "IoT Dataset 2022 | Datasets | Research | Canadian Institute for Cybersecurity | UNB," Aug. 2022. [Online]. Available: https://www.unb.ca/cic/datasets/iotdataset-2022.html

[17] "Proton VPN: Secure and Free VPN service for protecting your privacy," Apr. 2020. [Online]. Available: https://protonvpn.com/

[18] "World's Fastest VPN and Privacy Protection." [Online]. Available: https://hide.me/en/

[19] "Turbo VPN: Best Unlimited, Fast & Secure VPN Service." [Online]. Available: https://turbovpn.com/

[20] "Kaspersky VPN Secure Connection – Protect Your Online Privacy | Kaspersky." [Online]. Available: https://usa.kaspersky.com/vpn-secure-connection

[21] Hotspotshield, "Hotspot Shield: Fastest VPN for Streaming, Gaming & More." [Online]. Available: https://www.hotspotshield.com/

[22] Signal Lab, "Secure VPN - Safer Internet - Apps on Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.fast.free.unblock.secure.vpn&hl=en_US

[23] NGP Developer Studio, "FastVPN Pro - Secure Proxy - Apps on Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.fast.vpn.pro.android.network.app&hl=en_US

[24] "SuperVPN Fast VPN Client - Apps on Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.jrzheng.supervpnfree&hl=en_US

[25] D. Nobori and Y. Shinjo, "VPN Gate - Public Free VPN Cloud by Univ of Tsukuba, Japan." [Online]. Available: https://www.vpngate.net/en/

[26] M. T. Khan, J. DeBlasio, G. M. Voelker, A. C. Snoeren, C. Kanich, and N. Vallina-Rodriguez, "An Empirical Analysis of the Commercial VPN Ecosystem," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 443–456. [Online]. Available: https://dl.acm.org/doi/10.1145/3278532.3278570

[27] D. Xue, B. Mixon-Baca, ValdikSS, A. Ablove, B. Kujath, J. R. Crandall, and R. Ensafi, "TSPU: Russia's decentralized censorship system," in *Proceedings of the 22nd ACM Internet Measurement Conference*. Nice France: ACM, Oct. 2022, pp. 179–194. [Online]. Available: https://dl.acm.org/doi/10.1145/3517745.3561461

[28] D. Xue, R. Ramesh, V. S. S, L. Evdokimov, A. Viktorov, A. Jain, E. Wustrow, S. Basso, and R. Ensafi, "Throttling Twitter: an emerging censorship technique in Russia," in *Proceedings of the 21st ACM Internet Measurement Conference*. Virtual Event: ACM, Nov. 2021, pp. 435–443. [Online]. Available: https://dl.acm.org/doi/10.1145/3487552.3487858

[29] J. Nakutavičiūtė, "What is Deep Packet Inspection? | NordVPN," Jan. 2020. [Online]. Available: https://nordvpn.com/blog/deep-packet-inspection/

[30] W. J. Tolley, B. Kujath, M. T. Khan, N. Vallina-Rodriguez, and J. R. Crandall, "Blind In/On-Path Attacks and Applications to VPNs," p. 19.

[31] A. , "Obfsproxy: the next step in the censorship arms race | Tor Project," Feb. 2012. [Online]. Available: https://blog.torproject.org/obfsproxy-next-step-censorship-arms-race/

[32] "TrafficObfuscation – OpenVPN Community." [Online]. Available: https://community.openvpn.net/openvpn/wiki/TrafficObfuscation

[33] D. Nobori and Y. Shinjo, "VPN gate: a volunteer-organized public VPN relay system with blocking resistance for bypassing government censorship firewalls," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'14. USA: USENIX Association, Apr. 2014, pp. 229–241.