

Stealthy Peers: Understanding Security and Privacy Risks of Peer-Assisted Video Streaming

1st Siyuan Tang
Indiana University Bloomington
tangsi@iu.edu

2nd Eihal Alowaisheq
King Saud University
ealowaisheq@ksu.edu.sa

3rd Xianghang Mi
University of Science and Technology of China
xmi@ustc.edu.cn

4th Yi Chen
Indiana University Bloomington
chen481@iu.edu

5th XiaoFeng Wang
Indiana University Bloomington
xw7@indiana.edu

6th Yanzhi Dou
Independent Researcher
aaron.yzdou@gmail.com

Abstract—Peer-assisted delivery network (PDN) can significantly reduce the bandwidth cost incurred by traditional CDN services. However, it is unclear whether they have been deployed extensively and their security implications have never been investigated thoroughly. In this paper, we report the first effort to address this issue through an automatic pipeline to discover real-world PDN services and their customers, and a PDN analysis framework to test the potential security and privacy risks of these services. Our results have revealed the extensive adoption of PDN across the Internet, especially by Chinese video platforms. Most importantly, our analysis on these PDN services has brought to light a series of novel security vulnerabilities, i.e., free riding of PDN services, video segment pollution, and unreported privacy risks, i.e., resource squatting and extensive leakage of video viewers' IPs. We have responsibly disclosed these security risks to relevant PDN providers which in turn have well acknowledged our findings.

Index Terms—Peer-assisted delivery network; P2P network; content pollution; IP leak; security analysis; WebRTC

I. INTRODUCTION

With the ever-expanding footprint of video streaming in Internet traffic (projected to reach 74% mobile traffic in 2024 [15]), the techniques and infrastructures for effective and efficient delivery of video content become increasingly important. Past decades have witnessed the incessant growth of content delivery networks (CDNs) for distributing and caching web content across different geolocations, which however is considered to be expensive for video streaming. Further, CDNs today have been constrained by their deployment that may not be adequate for serving video-on-demand (VOD) or live-streaming users around the world, given that even the largest CDN provider has only 325K servers located in 1.4K networks by April 2021 [17]. An answer to these challenges is the emergence of *Peer-assisted Delivery Network (PDN)* that utilizes a Peer-to-Peer (P2P) protocol (i.e., *WebRTC* [12]) to facilitate video transmission among web browsers. This alternative is considered to be more scalable and cost-effective: for example, Peer5, one of the most popular PDN services, claims to be able to offload 95% bandwidth cost for its customers [27].

PDNs can be easily integrated into today's video streaming infrastructures: a video streaming website simply needs to subscribe to a PDN service and embed the respective PDN JavaScript SDK into its video streaming web pages or apps. Then, an ad-hoc P2P network among their viewers will be built up, with all coordination and management tasks handled by the PDN provider behind the scene. On the other hand, given known weaknesses of other P2P networks [39], [48], [58], [78], this new content delivery model may have serious security and privacy implications, which however have never been fully understood. Specifically, one may ask whether PDN services can prevent unauthorized use, whether video contents relayed by untrustworthy peers have been properly protected, whether user consents are clearly and freely communicated, whether and to what extent users' privacy is protected when involving video viewers in a PDN network.

Challenges and solutions. Answering these questions requires an in-depth analysis of existing PDN services, which turns out to have multiple challenges. First, PDN providers tend to hide their technical mechanisms with few or no publicly available technical documents as well as heavily obfuscated client-side PDN libraries, making it challenging to understand and evaluate existing PDN systems. Second, PDN services take a hybrid model of combining normal CDN traffic with P2P traffic, and most PDN activities are mixed with heterogeneous in-browser web activities, rendering them stealthy and hard to detect. Making it more complicated is that PDN services are dynamically loaded when visiting a video website or app, and a PDN customer may set various preconditions before loading the PDN services, e.g., the PDN traffic of Douyu TV (a live streaming platform) is only observable through IP addresses located in China.

Despite these challenges, we performed the first systematic study on PDN's security implications. Our study started by collecting publicly available PDN providers and their customers (e.g., video streaming websites). Specifically, our study identified 3 most popular public PDN providers as well as 10 private PDN services. For the public PDN providers, we extracted signatures for fingerprinting PDN SDKs (JavaScript and Android) and moved to build up a signature-based PDN

* This is a regular paper.

customer detector. Our detector led to the discovery of 134 websites and 38 Android apps as the *potential PDN customers*. Then, through dynamic analysis, we successfully triggered PDN traffic for 17 websites and 18 apps, which are considered as the *confirmed PDN customers*. Among these confirmed cases, 9 websites have over 1 million monthly visits, e.g., RT News (*rt.com*) and Clarin (*clarin.com*), while 11 apps have over 1 million downloads on Google Play, e.g., France TV (*fr.francetv.pluzz*) and iFlix (*iflix.play*). Furthermore, another 10 popular video streaming websites have been confirmed with proprietary PDN solutions integrated (*private PDN services*), among which, 8 are mainstream Chinese video streaming platforms, e.g., Bilibili (*bilibili.com*), Tencent Video (*v.qq.com*). Upon those PDN services and customers, we have conducted a comprehensive analysis in an attempt to identify fundamental security risks and privacy concerns. This has been made possible by a PDN analyzer we built up to automatically run predefined security tests.

Security discoveries. Our study on PDN services has brought to light significant security implications of these services, which have never been reported before. Particularly, we found that public PDN services are seriously vulnerable, due to not only misconfigurations on the side of the PDN customer, but also insufficient protection enforced by the PDN provider. More specifically, all public PDN services we discovered are meant to authenticate the peers of a customer using a static API key that is directly embedded in the customer's website. As a result, the attacker could easily retrieve the key to free-ride the PDN service at the cost of legitimate PDN customers. Our experiments show that 11 out of 40 API keys extracted from detected PDN customers did not enforce any protection against the free riding attack, and all public PDN services are vulnerable to an advanced free riding attack through domain spoofing. Furthermore, such a free-riding vulnerability is also confirmed for a private PDN service which serves a popular video streaming platform (i.e., Mango TV).

Another security-critical weakness we identified is video segment pollution: a malicious peer could alter any video segment it receives and forward it to other benign peers without being noticed. Although content pollution is a known threat in traditional P2P networks, previous attacks [39], [51], [52] rely on understanding of P2P protocols and access to local storage, which are not applicable to PDN scenarios since PDN services utilize customized data protocols and store the downloaded data in the cache under the protection of browsers. In our research, we proposed a novel attack wherein the attacker can replace arbitrary video segments without knowledge of P2P protocols or access to local storage. Our evaluation results demonstrated the feasibility of the attack over all public PDN providers and a demo [1] is published online: <https://sites.google.com/view/pdnsec/home/demo>.

Also discovered are concerning privacy violations in PDN services: a PDN service automatically exposes the real IPs of viewers and consumes their resources without consent. In our research, we discovered an extensive IP leak caused by PDN services, exposing viewers' IPs to untrusted peers. Our

experiments indicated that all existing PDN services, both public and private ones, do not have sufficient protection in place to restrain viewers' IP exposure. For instance, through watching a single live streaming channel, a peer under our control collected over 7K distinct IPs of viewers during a one-week experiment. Furthermore, we found that PDN services consumed peers' computing and bandwidth resources without consent. Our experiments showed that serving as a PDN peer generally incurs 15% more CPU and 10% more memory usage. Also, as the number of neighboring peers grows, the cost of uploading bandwidth increases significantly. Our further analysis of Peer5 customers revealed that 3 highly popular apps (i.e., *com.bongo.bioscope*, *com.portonics.mygp*, *com.arenacloudtv.android*) even allowed the PDN service to use viewers' cellular data for both uploading and downloading, which may incur extra financial cost to viewers.

Mitigation. To mitigate the security risks discovered, we discussed the limitations of known defense mechanisms and presented several protection suggestions along with a feasibility evaluation under a simulated environment. More specifically, for the service free riding risk, we proposed an authentication mechanism that utilizes a video-binding and disposable token, which can effectively demotivate unauthorized use of PDN services. To address the video segment pollution threat, we proposed a peer-assisted defense mechanism wherein the PDN server randomly selects a subset of peers to report and verify the integrity metadata (IM) for each video segment. This protection raises the bar for a content pollution attack, which will only succeed when all randomly selected peers are malicious. We also discussed the countermeasures for peer privacy risks (i.e., resource squatting and IP leak) by limiting the resource consumption for each peer and deploying TURN servers to relay peer-to-peer traffic.

Contributions. The contributions of the paper are outlined as follows:

- *A large-scale characterization on real-world PDN participants.* Our research reveals the technical mechanisms of PDN services as well as their prevalence through an automatic detection framework, leading to the discovery of 3 public PDN providers with their customers and 10 private PDN services.
- *The first study on PDN security.* Our study, for the first time, revealed serious security vulnerabilities (*service free riding*, *video segment pollution*) on all identified public PDN services and some private ones, and also reported concerning privacy risks (*IP leak*, *resource squatting*) on all PDN services.

II. BACKGROUND

Peer-assisted video streaming. Due to the significant benefit of traffic savings, peer-assisted video streaming has been adopted by numerous commercial CDNs including Xunlei Kankan [79], LiveSky [77], Spotify [44], and Akamai [81]. These P2P-CDNs generally require end users to install client-side software and design ad-hoc protocols for peer-to-peer communication. It is also a key challenge to integrate P2P with existing CDN services [77]. As previous P2P-CDNs discontinued for various reasons [14], PDN services based on

WebRTC emerge as the next-generation peer-assisted video streaming network, which provides a more convenient SDK and better security mechanisms. These PDN services are also embedded into web players [4] and enterprise content delivery network (eCDN) [6] for wide deployment. Regarding network structures, peer-assisted networks can be classified into two types: tree-based and mesh-based [43]. Literally, tree-based networks organize peers in a structure of multiple trees, selecting some peers as root nodes and others as leaf nodes. In a mesh-based network, peers dynamically connect to a subset of random peers based on attributes such as content/network availability. In our research, PDN takes the mesh-based network.

Video streaming protocols. A video streaming protocol specifies how media data is delivered over the Internet. Early examples of such a protocol include Real Time Streaming Protocol (RTSP) [70] and Real Time Messaging Protocol (RTMP) [64]. However, these protocols are either proprietary or do not support video streaming through HTTP. Thus, in the past decade, a set of HTTP-based adaptive bit-rate protocols have emerged and gained popularity, particularly HTTP Live Streaming (HLS) [3] and Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [74]. These protocols break a video into smaller segments that can be downloaded through HTTP. Such segments are made available at different bitrates, so as to allow the video client to adapt the video streaming to various network conditions. A manifest file is also created to trace these video segments. Among these protocols, WebRTC and Secure Reliable Transport [8] (SRT) are characterized by new features such as lower latency, security, P2P support, etc.

III. UNDERSTANDING THE PDN ECOSYSTEM

In this section, we present our understandings of the PDN ecosystem. We first introduce a typical PDN scenario and its key players, i.e., PDN providers, PDN customers, and peers (§III-A). Then we propose our methodology by identifying a set of representative PDN providers (§III-B) and detecting PDN customers (both websites and mobile apps) at a large scale (§III-C). We further analyze the impact of identified PDN customers (§III-D).

A. Overview

Generally, the PDN ecosystem consists of three key players, i.e., PDN providers, PDN customers, and peers. Among these players, PDN providers offer the PDN service and the integration SDK (e.g., JavaScript API), PDN customers refer to video streaming services and their apps or websites that subscribe to the PDN service, and peers denote the video viewers of these video websites or apps. Figure 1 illustrates a typical traffic workflow in the PDN ecosystem. In the traditional CDN mode, ① when a viewer (e.g., Peer A) opens a video website, ② it sends an HTTP request to the CDN which stores the specified video files and ③ downloads the video files before playing them with a video player. If the PDN service is enabled, ④ the PDN SDK integrated into the video website will automatically initiate a WebRTC interface and connect to the PDN server. In

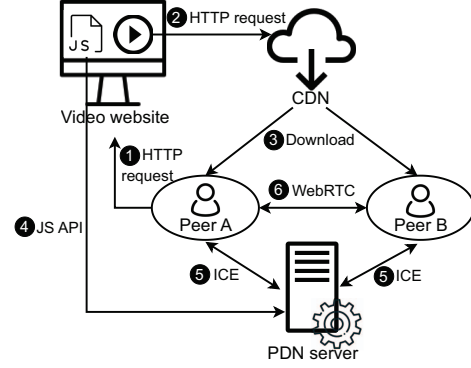


Fig. 1: Traffic flows of a typical PDN scenario.

the process of Internet Connectivity Establishment (ICE), ⑤ Peer A shares the meta data of video files and its network information (e.g., IP and port) with the PDN server. With the help of the PDN server, ⑥ other peers (e.g., Peer B) can request video segments from Peer A instead of the CDN network.

PDN distinguishes itself from previous P2P-CDN services in two aspects: First, PDN services introduce a trusted 3rd-party, i.e., the PDN server, to manage and control the peer-to-peer connections, which may introduce new security risks. Second, the PDN service operates as a plugin to existing CDN services and therefore the video platform can directly enable/disable the PDN service with little configuration to existing CDN services. Such convenience, however, renders PDN services almost unnoticeable to viewers and significantly increases the difficulty in detecting the presence of PDN services.

B. Identifying PDN Providers

To discover PDN providers, we first queried Google Search with PDN-related keywords like “P2P live streaming” and “P2P CDN”, which returned both highly-ranked PDN websites as well as market reports [28] covering popular PDN services. The providers identified were further verified by contacting them to get their service details. Through manual communication, we confirmed three popular PDN providers: Peer5 [22], Streamroot [19], and Viblast [26]. We further signed up as a customer of the verified PDN services so as to access their documentation, client-side SDKs as well as customer portals, which enabled us to gain insights into how these services work. Based on the new insights, we constructed a set of robust PDN signatures to help identify potential PDN customers, and manually verified them to understand their effectiveness (§III-C). Subscription to the services also allows us to enable the PDNs on our experimental video streaming website, through which we captured PDN traffic to study the PDN protocols and workflows (Figure 1).

C. Detecting PDN Customers

In our research, we proposed a novel framework to detect PDN customers at a large scale, including the websites and Android apps. Specifically, we first identified websites and Android apps integrated with PDN services, which we call *potential PDN customers*, using a signature-based approach. We further conducted a dynamic analysis to detect PDN traffic on those *potential PDN customers*.

Signature-based detector. Through an analysis on PDN services (§III-B), we collected a set of robust signatures to fingerprint their customers, which were extracted from their documentation and source code (js files or mobile SDK). These signatures consist of URL patterns (e.g., `api.peer5.com/peer5.js?id=*`), unique namespaces (e.g., `com.viblast.android`), and meta-data in the Android manifest file (e.g. `io.streamroot.dna.StreamrootKey`). All identified signatures are presented in our website [1]. Leveraging these identified signatures, we built up a scanner to crawl high-profile websites, in an attempt to find potential PDN customers.

Since PDN mainly serves video streaming platforms, our scanner focused on those popular video-related websites. Specifically, we first queried the top 300K domains according to Tranco Top Sites Ranking [65]. The Tranco ranking improves upon the shortcomings of existing ranking lists and turns out to be more stable and resilient to malicious actors. Then we collected the category information of these top domains as provided by the 5 category engines in VirusTotal, i.e., Forcepoint ThreatSeeker, Sophos, BitDefender, Comodo Valkyrie Verdict, and alphaMountain.ai. For each domain, if any of the 5 engines returns a category label containing keywords such as "tv" or "media", we consider it a video-related domain. In this way, we found 68,713 top video-related domains through this category filtering. Also, we queried source code search engines NerdyData [21] and PublicWWW [23] using PDN signatures as keywords, which reported 44 potential PDN-related websites. Altogether we gathered 68,757 domains for our PDN detector.

We then performed a signature-based scan between January 2022 and February 2022 using Selenium [10], a framework for automatic web application testing. Our scanner dynamically crawled the website of a given domain by downloading its HTML files and all JavaScript files if the site contains a "video" tag on its web page, and then traversed all the subpages under the same domain until a PDN signature was found. To limit the depth of searching, our scanner only examined the subpages within a depth of 3. To avoid non-negligible overhead to a website, we limited the crawl rate to 1 webpage per 3 seconds with a timeout of 10 minutes for a given domain. If any PDN signature was found in these subpages, the scanner considered the domain as a *potential PDN customer*.

We also collected popular apps and their APKs from Androzoo [31], a large repository of Android APKs from multiple app stores, including Google Play, Anzhi, and AppChina. By

TABLE I: Detected PDN customers.

PDN provider	Confirmed/Potential PDN customers		
	# website	# app	# APK
Peer5	16/60	15/31	199/548
Streamroot	1/53	3/6	53/68
Viblast	-/21	-/1	-/11
Total	17/134	18/38	252/627

June 2022, it contains 19,661,675 different Android APKs from 7,954,395 apps. Since Androzoo does not provide information about the app category or downloads, we randomly sampled 1.5M apps among the 8M apps. Our scanner automatically downloaded the latest APK version of sampled apps and then unpacked it to search PDN signatures on Android. An APK is considered as a *potential PDN customer* if it contains at least one PDN signature. We further checked all historical APK versions of detected apps to estimate the scale of different APK versions.

Our signature-based detector has led to the discovery of 134 websites and 38 apps (with 627 different APK versions) as *potential PDN customers*. As shown in Table I, among the services behind these customers, Peer5 is the most popular one, with a potential customer base of 60 websites and 31 Android apps. Then, it is Streamroot whose potential PDN customers consist of 53 websites and 6 Android apps.

Detecting PDN traffic. To further validate whether these potential PDN customers have actively enabled the respective PDN services, we designed a dynamic method to detect PDN traffic. Our approach is based upon the observation that PDN utilizes the plain-text STUN protocol to exchange IP information between peers. Then, when dynamically running a potential PDN customer, we captured its network traffic, from which STUN binding requests can be easily identified if available along with IP addresses of candidate peers. As WebRTC enforces a DTLS handshake between peers [9], we then checked all the DTLS connections that typically follow the STUN binding requests. If a DTLS connection is observed between known candidate peer pairs, we consider the respective website or app as a *confirmed PDN customer*.

For each *potential PDN customer*, we randomly selected 3 video links and watched them for 15 minutes so as to capture the traffic. As a result, we have successfully detected the PDN traffic for 17 websites and 18 apps. We listed all these confirmed PDN customers in Table II and Table III. For other *potential PDN customers*, we failed to capture any PDN-related traffic due to the challenges in triggering the service. Some potential PDN customers (e.g., eon.tv) did not allow our dynamic analysis server to access their video sources due to geolocation restrictions or service subscriptions, while some other customers may set specific constraints (e.g., IP constraint) to load PDN service or only enable the service for some subpages which could be missed in our testing.

Impact of PDN customers. We found that most PDN customers have a large number of viewers. To measure the popularity of PDN customers, we checked the monthly vis-

its of PDN websites from SimilarWeb [25] and installs of PDN apps from Google Play. Among the 17 confirmed PDN websites, 9 were found to have over 1 million monthly visits, including popular news websites RT News (*rt.com*), Clarin (*clarin.com*) and RTVE (*rtve.es*). Also, among the 18 confirmed PDN apps, 11 have over 1 million downloads from Google Play, including iFlix (*iflix.play*) with over 50 million downloads and 4 apps with over 10 million downloads, e.g., France TV (*fr.francetv.pluzz*) and Red Bull TV (*com.nousguide.android.rbtv*).

TABLE II: Confirmed PDN websites

PDN websites	PDN provider	# Monthly visits
rt.com	Streamroot	117M
clarin.com	Peer5	69M
rtve.es	Peer5	35M
jn.pt	Peer5	12M
ojogo.pt	Peer5	8M
dn.pt	Peer5	6M
servustv.com	Peer5	4M
www.popcornflix.com	Peer5	1M
tsf.pt	Peer5	1M
dinheirovivo.pt	Peer5	1M
www.sliver.tv	Peer5	—
hdo.tv	Peer5	—
www.souvenirsfromearth.tv	Peer5	—
www.severestudios.com	Peer5	—
www.performancevetsupply.com	Peer5	—
www.schoolfordesign.net	Peer5	—
9uu.com	Peer5	—

TABLE III: Confirmed PDN apps

PDN apps	PDN provider	# Google Play downloads
iflix.play	Streamroot	50M
fr.francetv.pluzz	Streamroot	10M
com.nousguide.android.rbtv	Peer5	10M
com.portonics.mygp	Peer5	10M
mivo.tv	Peer5	10M
com.bongo.bioscope	Peer5	5M
tv.fubo.mobile	Peer5	5M
com.rt.mobile.english	Streamroot	1M
vn.com.vega.clipvn	Peer5	1M
com.flipps.fitetv	Peer5	1M
vn.com.vega.clipvn	Peer5	1M
com.arenacloudtv.android	Peer5	500K
com.television.burma	Peer5	50K
com.totalaccessstv.live	Peer5	—
dev.hw.app.tgnd	Peer5	—
tv.almighty.apk	Peer5	—
com.rvcomx.brpro	Peer5	—
com.lts.cricingif	Peer5	—

D. Private PDN services

During our detection, we also observed 10 websites are embedded with HTML or JavaScript code that shares similar patterns with ones from PDN services, but they are not customers of either of the three PDN providers nonetheless. Unlike known PDN providers, these cases do not involve third-party API keys or external JavaScript APIs, but associate their own domains (usually their subdomains or relevant domains)

TABLE IV: Confirmed private PDN services

PDN websites	PDN server	# Monthly visits
bilibili.com	hw-v2-web-player-tracker.biliapi.net	911M
ok.ru	vm.mycdn.me	662M
douyu.com	wsproxy.douyu.com	95M
v.qq.com	webrtcpunch.video.qq.com	92M
iqiyi.com	broker-qx-ws2.iqiyi.com	82M
huya.com	wsapi.huya.com	61M
youku.com	ws.mmstat.com	60M
tudou.com	ws.mmstat.com	44M
mgvtv.com	signal.api.mgtv.com	42M
younow.com	signaling.younow-prod.video.propsproject.com	1M

with the involved PDN servers. Thus we consider them as *private PDN services* since they are ad hoc services with each dedicated to a specific video/live streaming platform.

Specifically, our signature-based detector identified 385 websites matching the general WebRTC-related signatures. We then conducted our dynamic detection on the top 57 websites that rank in top 10K websites and detected that 10 of them has integrated proprietary PDN functionalities (i.e., private PDN services), including 6 popular video streaming platforms, e.g., Youku (*youku.com*), Tencent Video (*v.qq.com*), and 4 live streaming platforms, e.g., OK Social Network (*ok.ru*), Huya TV (*huya.com*). We list these popular private PDN services in Table IV. It is interesting that such private PDN services are extremely popular in China, covering most top video hosting and live streaming platforms. One possible explanation is related to legal concerns since PDN services are similar to Torrent, which has been forbidden in US and EU. This also explains why we could only observe PDN traffic in China for some live streaming platforms (e.g., Douyu TV). Another possibility could be bandwidth cost. As the bandwidth cost in China is higher than that in EU/US, video streaming services in China have more incentives to use PDN services. We also noticed that 2 adult video platforms, i.e. *xhamsterlive.com* and *stripchat.com*, utilize WebRTC protocols to relay traffic. For the other 45 cases, we confirmed 3 cases invoking WebRTC APIs for web tracking. Yet we failed to trigger any PDN traffic for the remaining 42 cases.

IV. SECURITY AND PRIVACY RISKS IN PDN SERVICES

With an in-depth understanding of the PDN ecosystem (§III), we move forward to reason about and analyze potential security and privacy risks introduced by PDN services to the parties involved, especially the video viewers and PDN customers. In this section, we first present our threat model and ethical considerations and then propose a PDN analyzer framework (§IV-A). Following, we elaborate detailed security and privacy risks discovered in our research.

The threat model. In our research, we assume that the attacker is able to participate in the PDN system as a PDN peer, as well as intercept the traffic between a PDN peer (under his control) and the PDN server. Specifically, the attacker can configure the peer with a self-signed root certificate to decrypt the traffic. In the free riding attack, we assume the attacker is

capable of retrieving the access token (which is plain text in the HTML code or traffic) and integrating PDN services with its own websites/apps. In the video segment pollution attack, we assume the attacker has access to the original video files. Unlike previous works [39], [62], [75], we do not require the attacker to have any knowledge of PDN protocols, nor does the attacker need to access the local storage.

A. The PDN Analyzer

Monitoring PDN activities. When running PDN peers as separate containers, we want to monitor PDN activities in terms of network traffic and resource consumption (e.g., CPU and memory). To monitor the network traffic, tcpdump [11] is started to dump incoming and outgoing network traffic on the default virtual network interface *docker0*, following

Fig. 2: The architecture of the PDN analyzer.

Tests on PDN services. We made an attempt to uncover the security and privacy risks of PDN services from their documentation, previous studies [32], [43], [81], and preliminary experiments. Specifically, previous studies on peer-assisted networks focus on three fundamental issues, i.e., *peer authentication* (§IV-B), *content integrity* (§IV-C), and *peer privacy* (§IV-D). Following these directions, we designed a series of tests with the help of our PDN analyzer and identified multiple security and privacy risks, as detailed below.

As discussed in §III-B, public PDN services operate in a pay-as-you-go model and a PDN customer is charged for every use under its name. Leveraging our PDN analyzer and subscriptions to PDN services, we explored whether the use of a PDN service is well authenticated and whether a PDN customer can be overcharged for the use incurred by other parties. It turns out that a persistent access token (API key) issued by the PDN provider is used to authenticate PDN customers and PDN peers. Such an access token was found to be publicly visible to attackers since it is statically embedded by the PDN customers in either the PDN mobile app or the video webpages. This allows an attacker (e.g., a misbehaving video streaming site) to easily steal a legitimate PDN customer token through either a colluding peer or static analysis of the respective mobile app if available. It can later utilize the token to *free-ride* the PDN service at the cost of the legitimate customer, or even maliciously consume P2P traffic between controlled peers to incur extra cost to the targeted PDN customers. Our evaluation of real-world customers further confirmed the pervasiveness of the service free riding risk. Following we elaborate on the service free riding risk and our findings, under the threat model where an attacker is capable of retrieving the PDN access token and the domains (origins) from a legitimate PDN customer.

token to authenticate peers, which, however, makes it possible for the attacker to free-ride the PDN services.

We then design two peer authentication tests to evaluate the authentication mechanisms of PDN services. In the first test (*cross-domain attack*), we first integrate known PDN SDKs into our own test website (www.test.com) and play a customized video stream source. To simulate peers' behaviors, we run two peer containers and configure the web driver to open a customized video stream for each peer. We then analyze the traffic between these containers (peers) and the PDN server: a successful binding of the peers indicates the peer passes the authentication process of the PDN server. In the second test (*domain-spoofing attack*), we configure the proxy server to spoof the HTTP headers, i.e., "Origin" and "Referrer", between our test website and the PDN server. In our settings, our test website (www.test.com) integrates the PDN SDK with an API key retrieved from the victim domain (www.example.com). When a viewer opens the test website (www.test.com), the proxy server modifies the HTTP headers to the victim domain (www.example.com), which deceives the PDN server that the HTTP requests are initiated from a legitimate domain. During all the experiments, we disabled the auto-play function and ensured that no data was actually transferred between peers, thus no cost was generated for customers owning these tokens.

Service free riding. We applied for a free trial from all the 3 public PDN providers and evaluated our peer authentication tests on them. As a result, we found serious free riding risks on these PDN services. In the default settings, both Peer5 and Streamroot allow arbitrary domains to connect to the PDN server with the API key. Thus our cross-domain attack can easily bypass the authentication process. Viblast requires setting up the domain allowlist before enabling the PDN service, which is effective against the cross-domain attack. We then enable the domain allowlist protection for all the 3 PDN services and verify whether the domain spoofing attack works. As a result, all three PDN services were found to be vulnerable to the domain-spoofing attack.

To verify the free riding risk in the real world, we also evaluated the peer authentication tests on detected PDN customers. From the detected potential PDN customers, we successfully extracted 44 API keys through regular expression matching, while others are heavily obfuscated (e.g., `_0x101f38[_0x2c4aeb(0x234)]`) or dynamically loaded in the runtime. We tested all of them and found 40 were valid during our test and the other 4 were expired. Among the 40 valid API keys, 11 of them were vulnerable to the cross-domain attack, which means they did not enforce the domain allowlist protection, and all of them were vulnerable to the domain-spoofing attack. Our findings prove the prevalence of free riding risk in the real world.

The free riding risk allows an attacker to abuse the PDN subscription of a legitimate PDN customer and maliciously generate peer traffic to increase the cost of the victim PDN customer. Among the 3 PDN providers, Peer5 and Streamroot charge their customers based on monthly P2P traffic (e.g.,

TABLE V: Security and privacy risks of PDN services.

Security risks	Peer5	Streamroot	Viblast	Private
Peer Authentication				
cross-domain attack	11/36 ¹	0/1	0/3	1
domain-spoofing attack	□	□	□	1
Content Integrity				
direct content pollution	×	×	×	–
video segment pollution	□	□	□	–
Peer Privacy				
IP leak	□	□	□	□
resource squatting	□	□	□	□

¹ a/b represents # vulnerable API keys / # all valid API keys.

Peer5 charges 500\$ for 50TB of P2P traffic), and Viblast is priced at 0.01\$ per concurrent viewer hour. Thus an attacker could generate a significant volume of P2P traffic or a large number of concurrent viewers on his/her own website integrated with the victim's PDN SDKs, which would increase the PDN cost of the victim customer. In our experiments with free trial accounts, we successfully increased the PDN service cost by initiating multiple containers as peers to generate P2P traffic, which demonstrates the feasibility of such an attack.

Private PDN services. Private PDN services operate in a way similar to public ones, which assigns a temporary token to each peer and utilizes the token to authenticate later communications. However, unlike public PDN services, private PDN services are hidden in the complex source code of popular video platforms and are usually deeply coupled with other video streaming modules, making it extremely hard to integrate them into our own test website and play our customized video source. This prevents us from performing our peer authentication tests ethically.

Despite the aforementioned challenges, we managed to hook the player SDK of one popular private PDN service (i.e., Mango TV) after months of effort to analyze its code, and successfully integrated it into our test website. Particularly, we observed effective PDN traffic for data transmission between peers on the test website. This confirms that the attacker can free-ride such a PDN service with no constraints. Our findings support that the free riding risk also exists in private PDN services. Moreover, our further analysis suggests that another popular private service, i.e., Tencent Video, does not bind the authentication token with the video source URL, which may expose it to similar free-riding risks.

C. Content Integrity

Previous pollution attacks on P2P video streaming networks directly send polluted chunks to other peers [39], [62], [75] and the feasibility of such attacks is largely dependent on the speed at which the attacker can modify content [53]. Different from existing P2P networks (e.g., BitTorrent), PDN enforces protection mechanisms over both the communication channels and the storage. First, PDN utilizes TLS encryption to protect ICE communication between peers and the PDN server (5 in Figure 1). Also, peers in PDN are connected via WebRTC, which supports video streaming protocols over

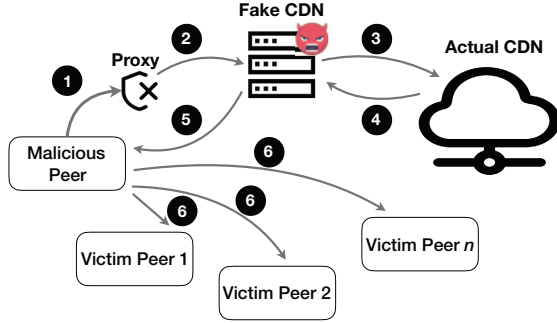


Fig. 3: Illustration of our content integrity attack.

DTLS encryption (6 in Figure 1). Second, PDN caches the downloaded content in the memory of browsers, which is protected by the same-origin policy and purged after a short time. Such protections render existing content pollution attacks ineffective. Thus PDN providers claim the PDN service is as secure as traditional CDN services [24]. In our research, however, we proposed a novel attack to compromise the content integrity in PDN. Our attack is based on the observation that the PDN server does not store video segment files and thus is unable to verify the integrity of a video segment by itself. Although the other channels are well protected under the assumptions, the attacker can still download “fake” video segment files and then spread them to other peers with the help of a malicious peer. Note we do not require the attacker to have any knowledge of PDN implementations or protocols, nor does the attacker need to access the local storage, i.e., cache of browsers. Instead, we assume that the attacker has access to the original video files and the corresponding manifest files. This is practical with the help of existing browser plugins such as *Live Stream Downloader* [5].

Our attack. Our idea is to run the proxy server in the middle of a controlled peer and the real CDN. The proxy acts as a fake CDN to download video files from the real CDN, and automatically replaces the video files before forwarding to the controlled peer. As illustrated in Figure 3, when the malicious peer under the attacker’s control visits a video website, the video source URL (pointing to the CDN that stores the target video) is redirected by a proxy server to a fake CDN. Then the fake CDN utilizes the video source URL to download the original video files, and alters video files for pollution. When the malicious peer downloads the altered video, it deceives the PDN server and other peers that the original video content is played. This allows the attacker to propagate the polluted video files to other peers.

Content integrity tests. As discussed in §II, video streaming protocols usually split a large video file into small segments (TS files) and a manifest file (e.g., an M3U8 file) is utilized to track these segments. To evaluate the content integrity of PDN services, we first run the *direct content pollution attack*, in which the attacker pollutes all the video segments based on the manifest file. In our PDN analysis framework, we set up two peer containers to play a customized video stream. The

proxy server redirects one peer’s video stream URL to a fake CDN and replaces the video files. Then we observe whether the other peer’s video files are polluted. In the second test (*video segment pollution test*), the attacker replaces one or more video segments (except for the first several ones) and keeps the other video segments and manifest file unchanged.

Video segment pollution. As mentioned in §IV-B, we leveraged free-trial access to all three public PDN services and performed the content integrity tests. As a result, the direct content pollution attack failed for all three PDN services. This is because PDN services utilize a “slow start” strategy and the first several video segments are directly downloaded from the CDN. Thus the polluted video segments would cause inconsistency and be detected by PDN services. However, all three PDN services were vulnerable to the video segment pollution attack, as shown in our demo [1]. Our findings indicate PDN services fail to enforce robust integrity verification on video files and have serious security concerns. During our experiments on popular PDN customers, we typically observed over 10 concurrent connections trying to download content from our controlled peer. As revealed in a recent study [75], a content pollution attack in a P2P live streaming system will quickly propagate to 47% of viewers in the initial stage even when the initial number of polluters is small. Considering the popularity of PDN customers, the video segment pollution attack in PDN can easily impact millions of viewers.

Private PDN services. As mentioned in §IV-B, we were unable to conduct our content integrity tests directly on private PDN services due to ethical concerns. Thus we evaluated our content integrity tests on the private PDN service of Mango TV, which we successfully extracted and integrated into our test website with a customized video source. From our tests, the private PDN service of Mango TV can prevent the direct content pollution attack. Regarding the video segment pollution attack, although we observed consistent DTLS data transmission between the malicious peer and the victim peer, we failed to observe the polluted video segments being played. This is probably because private PDN services maintain access control on all the existing video sources (e.g., digital rights management (DRM)) and our customized video source is not registered. Note an attacker could conduct our attacks on the original video sources to bypass such constraints. We acknowledge it as our limitation.

D. Peer Privacy

In this section, we study whether viewers are well aware of and able to disable the PDN service, and how significant the potential IP leak and extra resource consumption caused by PDN services.

User consent. A previous study involving large-scale users [81] reveals that only around 30% of all video viewers opt-in to participate in P2P video streaming networks. Thus it is significant to ask for consent when recruiting a video viewer into PDN, otherwise, it is a compromise of privacy. As a PDN participant, a viewer should be informed that the potential risks and costs of joining the PDN network and be able to disable

the PDN service. We manually checked all the potential PDN customers (including the 134 websites, 38 Android apps, and 10 private cases) detected in our study and manually inspected their services and public documentation. The results showed that none of them provide any pop-up windows to ask for viewers' consent or communicate with their viewers the P2P network they are about to join through "Terms of Use" or other web content. Therefore, we believe that their viewers are completely left in the dark about the price, both in terms of potential privacy leaks and the extra resource consumption. Also, none of the PDN providers we studied allow viewers to turn off the PDN function.

In the absence of user consent, we further design two privacy tests to evaluate the extent of potential IP leak and extra resource consumption caused by both public and private PDN services.

IP leak test. To verify the IP leaks in public PDN services, we initiated two remote peers (one located in the US and the other in China) that watched the same video stream on our test website. PDN services utilize STUN protocols in the WebRTC API [68], which is designed for NAT traversal to exchange IP addresses. To collect the peer IP exchange, we write a Wireshark script to automatically extract the IP exchange requests and responses in STUN protocols. In our experiments, we successfully collected the other peer's IP for all three PDN services, which implies the extensive IP leak.

IP leak in the wild. To measure the extent of peer IP leaks in the real world, we selected two popular PDN customers: RT News app integrated with the Streamroot service, and Huya TV website integrated with a private PDN service. During the experiments, our tests collected only IP addresses of viewers communicating with our controlled peer, which were removed immediately after generating the aggregated statistics. Specifically, we collected two-hour traffic from a controlled peer in a live channel for each of the two customers lasting for one week. Altogether, our PDN analyzer gathered 7,740 unique peer IP addresses, including 7,055 from Huya TV and 685 from RT News. We then further queried IPInfo [20] for these addresses' IP WHOIS information (e.g., geolocations) and found that 7,159 of these IPs are public IPs, along with 581 as bogons [18]. Among these bogon IPs, 543 are in private networks, 33 are for NAT [76], and the other 5 are reserved IPs. These IPs (private, NAT, reserved) were returned probably due to the errors in the NAT traversal process, which replied with unreachable IPs to our controlled peer. Also, among the public IPs, 98% of Huya TV are in China, while IPs from RT News distribute across 259 cities in 56 countries, with United States (35%), Britain (17%), and Canada (13%) being the top 3 countries. The results are consistent with the distribution of viewers for these two PDN customers. We also performed the same experiment on all the 10 private PDN services. The results are similar to Huya TV except for ok.ru, in which we only collected 8 Russian IPs possibly due to geolocation constraints. Our experiments demonstrated that all PDN services expose viewers' real IPs extensively with few protections. This enables an attacker to harvest viewers' IPs

and link them to the content of the videos being watched.

Resource squatting tests. We further estimate the extra resource consumed for supporting PDN services. In our PDN analyzer, we run a set of peer containers and configure their web drivers to open our test website simultaneously. On top of these containers, the monitor records through Docker Engine APIs the status of each container per second, including the CPU usage, memory statics and network I/O.

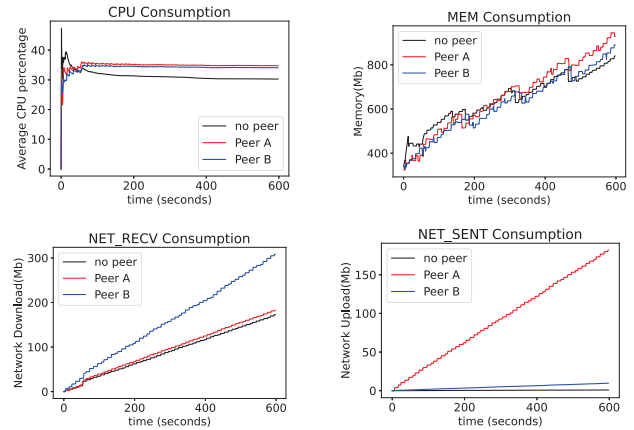


Fig. 4: Resource consumption of serving as a PDN peer.

Figure 4 shows our test results on the Peer5 PDN service, including the CPU and memory usage and download/upload bytes measured under two peers, Peer A and Peer B, together with *no peer*, which means viewers directly request the video from CDN. As we can see, the utilization of the Peer5 PDN service incurs non-negligible overhead for both peers, at a cost of an additional 15% CPU and 10% memory compared with *no peer*. This is mainly caused during the process of data encryption and decryption to transmit the video segments.

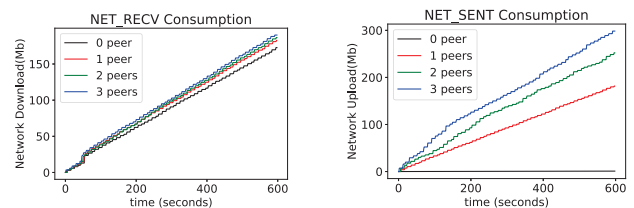


Fig. 5: Bandwidth consumption of serving multiple peers.

We also measured the resource consumption of Peer A with the existence of multiple peers. When adding the number of peers (up to 3 peers other than Peer A), we found that the CPU, memory, and download traffic do not have significant differences, mainly due to the scalability of WebRTC protocols. However, the upload traffic increases significantly (up to 200% of the download traffic with 3 peers) as the number of peers grows, as shown in Figure 5. Due to the limit of our network bandwidth, adding more peers (over 5 peers) will significantly lower the download traffic of peers and thus affect our experiment results.

Our results show that PDN services incur extra resource consumption (i.e., CPU, memory, bandwidth) for video viewers without their consent, which may have legal concerns. Similar to cryptojacking [40], PDN services unauthorizedly exploit viewers' resources for monetization. PDN services also expose viewers' IP addresses to untrusted peers and thus compromise their privacy. Moreover, such resource squatting behavior has also motivated viewers to disable or filter PDN services. For example, viewers have utilized AdblockPlus to block the domain of PDN servers to disable these services [16].

Resource squatting in the wild. We further analyzed how real-world video platforms configure the PDN service running in their viewers' browsers. This was made possible by our observation that Peer5 included in its JavaScript code an unprotected variable specifying the configurations. Through analysis of these configurations, we identified 3 Android apps, i.e., *com.bingo.bioscope*, *com.portonics.mygp*, *com.arenacloudtv.android*, allowed the use of cellular data for both uploading and downloading. As shown from our resource squatting tests, such settings may increase traffic consumption significantly for their viewers and generate extra costs. The other customers were in *leech* mode, i.e., consuming cellular data for downloading only. Our measurement shows that some PDN customers consume peers' cellular data for uploading data, and 3 popular Android apps (with over 15 million Google Play downloads in total) are under such configuration, which may generate extra cellular data cost to their viewers.

V. RISK MITIGATION SUGGESTIONS

In this section, we provide several suggestions for mitigating the risks of service free riding (§V-A), video segment pollution (§V-B), and peer privacy (§V-C).

A. Mitigating the Risk of Service Free Riding

Existing authentication mechanisms. As discussed in §IV-B, the existing domain allowlist defense can be easily bypassed through a spoofing attack. OAuth [45] is a widely used authentication framework proposed to authorize third-party access without providing credentials. Compared with a persistent API key, OAuth utilizes a temporary authentication token and thus can reduce the risk of exposing the credentials (API keys). However, for the service free riding attack, an attacker can perform a man-in-the-middle (MITM) attack to redirect viewers' requests to a legitimate PDN customer and get valid tokens to access the PDN service. To defend against the MITM attack, existing mechanisms (e.g. Token binding [49]) are based on the trust of clients, which is unfortunately not the case for peers in PDN services.

Disposable and video-binding token. To address the challenges above, we suggest to implement a disposable and video-binding authentication token that binds to valid video streams on legitimate PDN customers. In this case, the attacker cannot utilize these tokens to offload the traffic of its own video streaming, which significantly reduces the economic motivations for a service free riding attack. Also, we suggest to add usage time limits and TTL in a token to prevent a replay

attack. Listing 1 illustrates an example of the token. Firstly, *customer_id* is a string designed for uniquely identifying each PDN customer, which should be assigned from the PDN provider. Then, the PDN customer server can assign each PDN peer a unique identifier, namely, *pdn_peer_id*. Also included is a *video_ids* field to identify the set of videos to be streamed in the current page. There are multiple options to compose a video identifier, e.g., utilizing the full-qualified video URL. Following are the token issuance time (a Unix timestamp) and a *tvl* field to denote the time to live in seconds since issuance. The *timestamp* and *tvl* jointly decide whether the token is expired or not. Another field *usage_limit* is defined to constrain the number of usage limit for this token.

Listing 1: The token structure.

```
"customer_id": "xx.yy",
"pdn_peer_id": "1",
"video_ids": [
  "https://xx.yy/zz.m3u8",
  "https://xx.yy/hh.m3u8"
],
"timestamp": 1619814238,
"ttl": 60,
"usage_limit": 1
```

To implement this token, we use JSON Web Token (JWT) [50], an open industry standard for authentication, to transmit the token and its digital signature. In our experiment, the example token along with its HMAC-SHA256 signature will result in an encoded JWT of 283 bytes. Our evaluation shows that this defense incurs acceptable extra transmission bytes during generation, transmission and verification, which is aligned with previous works on JWT applications [66], [73].

B. Mitigating the Risk of Video Segment Pollution

Existing content pollution defenses. Previous studies [39], [42], [62], [82] require the video source to distribute every video chunk with an extra integrity attribute (e.g., hash-based chunk signature) for verification. However, such defenses may lead to higher CDN bandwidth costs for PDN customers since all the viewers (even those not in PDN) download these integrity attributes from CDN. Although it is possible to reduce the bandwidth cost by signing multiple chunks, this means the viewer has to wait for the arrival of multiple chunks for verification, which incurs longer delays. Also, PDN, as a third-party service, is required to minimize the changes to existing video streaming services. Motivated by previous defenses and the unique natures of PDN, we proposed a peer-assisted integrity-checking that incurs no extra CDN bandwidth cost and is compatible with existing streaming infrastructures.

Peer-assisted integrity checking. Similar to [82], our mechanism randomly selects PDN peers to calculate the integrity metadata (IM, e.g., cryptographic hash) for a video segment. However, our mechanism differs by utilizing the trusted PDN server, which is used to resolve IM conflicts and blacklist malicious peers. In our mechanisms, these calculated IMs are

TABLE VI: Evaluation for IM checking.

Browser	PDN	IM checking	CPU	Memory	Latency
Chrome	No	No	1	1	-
Chrome	Yes	No	1.11	1.21	67ms
Chrome	Yes	Yes	1.14	1.24	140ms

reported to the PDN server, and the PDN server consider an IM is *authentic* if all selected peers report the same IM. Since malicious peers can report fake IMs, the PDN server downloads the specified video segment from CDN and calculate the authentic IM if an IM conflict was detected. Peers reporting falsified IMs will be blacklisted. *As long as there are benign peers reporting IMs, the peer-assisted integrity checking can help identify the authentic one.* The authentic IM will be further signed by the PDN server, resulting in signed integrity metadata (SIM). The SIM for each video segment will be distributed to peers for integrity checking. Note a peer will report the IM only when the respective video segment is downloaded directly from the CDN, and a video segment downloaded from other peers must be verified using its SIM.

To ensure the integrity of video content, the integrity metadata should be robust to the replay attack. Specifically, an attacker may record a legitimate video segment along with its SIM, masquerade it as another video segment, send it to the victim peers, and thus disrupt the video delivery. Also, such a replay attack can occur across videos. Therefore, the IM should also be able to verify which video a segment belongs to as well as its position in the manifest file. In our design, the IM is calculated as the cryptographic hash of the tuple of video segment content, the video identifier, and the position of the video segment in the manifest file.

The peer blacklist. Since the PDN server will download a video segment from CDN when a conflict occurs, the attacker may keep sending fake IMs to increase the server overhead and traffic cost for the CDN. It is necessary for the PDN server to track peers and maintain a peer blacklist. Specifically, the PDN server assigns a unique ID to each peer at the start of the session. The ID should bind to the peer's public IP address and port or other information for tracking. If a peer is detected to have involved in suspicious behaviors (e.g., sending a fake IM), it will be blacklisted and removed from the peer candidates. Note this ID should only be visible to the PDN server in case other peers may abuse the ID for tracking.

Evaluation. We set up a simulation environment to evaluate the feasibility and performance of our peer-assisted integrity checking mechanism. Given the absence of any open-source PDN system, we built our simulation based upon PeerJS [7], an open-source WebRTC library. We first developed a custom signaling server and a PDN JavaScript SDK, along with a website integrating this SDK. Then we implemented our defense and evaluated the performance with our PDN analyzer. It shows our defense successfully detected polluted segments in the existence of malicious peers. We further evaluated the performance overhead, specifically profiling the resource consumption in IM calculation and verification. This is achieved

through three groups of control experiments. In each group, we specified 6 peers, with 3 as the senders and the other 3 as the receivers. Each receiver peer requests from the senders a typical video segment with a length of 10 seconds, lasting for a total of 600 seconds. Different groups are set based on settings including whether to do P2P video segment delivery, and whether to do IM calculation for the sending peers and IM verification for the receiving peers. During the experiments, we use Docker API to measure peers' resource consumption, and the latency of IM checking is calculated by the time difference of $T_{recv} - T_{send}$, where T_{recv} is the receiving time after IM verification and T_{send} means the sending time before IM calculation. As shown in Table VI, our defense will incur negligible CPU and memory consumption on peers. And the latency incurred by IM checking is less than 80ms for a video segment of 3MB size.

C. Mitigating the Risk on Peer Privacy

Existing peer privacy defenses. Previous works [38], [71] have proposed various mechanisms to protect users' stream privacy in peer-assisted video streaming networks. These defenses distribute fake streams to prevent attackers from inferring the real stream that users are watching. However, such solutions significantly change the existing video stream infrastructure and cannot prevent the peer IP leak risk.

Peer privacy risk mitigation. Although existing video streaming infrastructure cannot prevent peer privacy risks, we suggest potential countermeasures to restrain these risks. To mitigate the resource squatting risk, PDN providers and customers should inform viewers of the existence of PDN services and allow viewers to disable PDN services and control the resources consumed for P2P purposes, such as limiting the maximum uploading bandwidth.

Regarding the IP leak risk, a straightforward solution is to limit the number of candidate peers that exchange IPs with each other. Specifically, the PDN server can retrieve the real IP of peers and query the information of these IPs such as geolocation and ISP. Based on the information, the PDN server can configure candidate peers to those sharing the same country or ISP. Such countermeasures can prevent unnecessary IP exposure effectively. From the test results in §IV-D, the number of leaked peer IPs will decrease significantly, i.e., only 35% leaked IPs from RT News are in the same country as our controlled peer, and none of the leaked IPs from Huya TV will be visible to our controlled peer.

Although the heuristic method above mitigates the IP exposure to some extent, an attacker can still bypass this defense through a proxy peer. Also, constraining the number of candidate peers may affect the QoS of PDN services. A fundamental solution provided by WebRTC is to relay traffic between peers through TURN servers [13]. TURN servers act as proxy servers between peers and can be utilized to circumvent network censorship [33]. With the existence of TURN servers, peers do not communicate directly and thus prevent the peer IP leak risk. As mentioned in §III-C, we observed two adult video platforms (*xhamsterlive.com* and

stripchat.com) utilized TURN servers to relay traffic. This is probably designed to protect the viewers' privacy since watching adult videos is privacy-sensitive. However, peer communications in PDN can incur a large volume of network traffic and thus cause huge overhead to TURN servers, which is not feasible in a large-scale PDN system.

VI. DISCUSSION

Microsoft eCDN. As Microsoft eCDN acquired Peer5, we further measured whether the identified risks still exist in the Microsoft eCDN service, which can be integrated into Microsoft Stream and Teams to offload traffic for live events. From the documentation [6], Microsoft eCDN utilizes Microsoft tenant ID as the API key, which is shared across the enterprise and no longer publicly visible. Thus it prevents the free riding attack. Regarding content integrity, we conducted our tests on the silent simulator provided by Microsoft eCDN, which runs peers in headerless browsers to transmit data. In the direct content pollution test, no peer connection is observed; in the video segment pollution test, we observed the polluted video segments being transmitted from the malicious peer to the victim peer. Our results indicated that Microsoft eCDN also suffers from our video segment pollution attack.

Limitations. Our detection of PDN customers (§III-C) may miss cases when the signature-based detector failed to capture the web pages that dynamically load the signatures, or when the dynamic analysis failed to trigger the PDN traffic due to various real-world constraints. Also, while the PDN ecosystem is found to be vulnerable to serious security risks, we failed to evaluate some of our security tests on most private PDN services due to ethical concerns. In our future work, we plan to work together with private PDN providers and explore how to mitigate such risks in real-world PDN activities.

Responsible disclosure. We have responsibly reported the aforementioned security risks to relevant parties, including all public PDN providers, Microsoft eCDN, and Mango TV. Both Peer5 and Viblast have responded to our disclosure and acknowledged the disclosed risks, and we are waiting for others' responses. Specifically, for the service free riding risk, Peer5 acknowledged that non-browser clients could spoof the origin and incur extra costs to the customers. And regarding the video segment pollution attack, both Peer5 and Viblast acknowledged the security vulnerability. Peer5 also claimed that they provide premium features to check the integrity of video segments, which requires a custom HTTP delivery. Viblast mentioned that they provide a player plugin to implement an MD5 segment hash checking, which requires a server to compute the MD5 value of each video segment. Both solutions require changes of existing infrastructure and thus are not feasible for PDN customers. In terms of user consent, both Peer5 and Viblast argued that they suggest their customers inform users of the potential resource consumption and not use cellular traffic for uploading.

Data and code release. Relevant datasets and source code have been released online [1]. We have open-sourced most of

our study infrastructure, including our PDN customer detector and PDN analysis framework.

VII. RELATED WORKS

P2P video streaming. Multiple large-scale measurements have been explored to leverage residential peers for video streaming services, called P2P-CDNs, including Xunlei Kankan [79], LiveSky [77], Akamai [81] and Spotify [44]. All these P2P-CDNs require users to install client-side software and user consent to enable P2P services. Another set of work explored utilizing residential gateway devices such as Wi-Fi hotspots and cellular base stations [54], [55]. More recent research aims to get rid of client-side software or devices through WebRTC. Typical examples include Hive.js [69] and Maygh [80], which are similar to the paradigm under our study but not compatible with existing CDN infrastructures. Moving forward, [36] explores how to utilize edge nodes to fulfill privacy-preserving video streaming. Our research, for the first time, systematically measures the security risks of the emerging PDN ecosystem built upon WebRTC, which reveals serious vulnerabilities.

Security risks in P2P streaming. P2P streaming networks have been proven vulnerable to various attacks [35], [46], [51], [72]. Prithula Dhungel et al. [39] performed the first content pollution attack in a commercial P2P live stream by mixing bogus chunks to degrade the quality of a video stream. William Conner et al. [35] investigated selfish and malicious behaviors such as DoS attacks in P2P media systems. Maya Haridasan et al. [46] first proposed the concept of collusion attacks in which the attacker compromises a subset of nodes in P2P live streaming networks. To address these risks, a lot of works [35], [46], [53] proposed defense mechanisms in P2P streaming networks. Roverli P. Ziwich et al. [82] proposed a distributed diagnosis of content pollution in P2P live streaming networks based on a comparison among all neighboring peers, which inspires our mitigation for the video segment pollution attack. Our work revealed a new attack in the PDN system, i.e., the free riding attack, and a novel method to perform the pollution attack through the collusion of a fake CDN and a malicious peer.

IP leak in P2P networks. It has been known that P2P networks leak peers' information and can be abused to compromise anonymity [57], [61]. To constrain the IP leaks, previous P2P networks deploy P2P overlays on top of physical IP layers to protect peers' IP addresses [56]. Recently, a series of studies [30], [41], [47], [67] investigated the IP leaking concerns caused by WebRTC, which reveals users' real IP addresses through WebRTC API. And De Groef et al. [37] studied the identity authenticity of communicate peers and proposed several attack scenarios to compromise peers' identity authenticity.

Resource squatting. A line of works [34], [40], [63] have revealed cryptojacking wherein device computing resources are abused by miscreants for cryptocurrency mining. In addition, another abuse scenario is the unauthorized monetization of

residential and mobile devices into web proxies to relay third-party network traffic [59], [60]. Natalie Silvanovich of Project Zero Team [29] also reported a series of important CVEs in WebRTC-based voice calls, such as initiating silent calls without user consent. Moving forward from these studies, we reveal for the first time how PDN services expose viewers' IPs and consume viewers' extra resources without user consent.

VIII. CONCLUSIONS

In this paper, we carried out the first empirical study on the security risks of PDN ecosystem. Our study revealed that PDN services have been deployed in many real-world services, especially in Chinese video platforms. Through a PDN analysis framework, we uncovered and evaluated significant security risks, i.e., service free riding, video segment pollution, and unreported privacy violations, i.e., IP leak, and resource squatting, which may affect millions of video viewers. Upon a solid understanding of these security and privacy risks, we proposed several defense options to mitigate the risks.

IX. ACKNOWLEDGEMENT

We would like to thank the anonymous reviewers for their insightful comments. This work is supported in part by NSF CNS-1801432. Xianghang Mi is supported by NSFC through award 62302473 and by USTC through the Innovation Fund for Young Investigators.

REFERENCES

- [1] Attack demo on content pollution. <https://sites.google.com/view/pdnsec/home>.
- [2] Develop with Docker Engine API. <https://docs.docker.com/engine/api/>.
- [3] HTTP Live Streaming. <https://tools.ietf.org/html/rfc8216>.
- [4] JW Player - Hybrid Peer-to-Peer Delivery. <https://demos.jwplayer.com/peer-accelerated-delivery/>.
- [5] Live Stream Downloader. <https://chrome.google.com/webstore/detail/live-stream-downloader/looebdlpjgdmkpcdfhdbmpbcfekj?hl=en>.
- [6] Microsoft eCDN Documentation. <https://learn.microsoft.com/en-us/ecdn/>.
- [7] PeerJS - Simple peer-to-peer with WebRTC. <https://peerjs.com/>.
- [8] Secure Reliable Transport. <https://tools.ietf.org/html/draft-sharabayko-srt-00>.
- [9] Security Considerations for WebRTC. <https://www.rfc-editor.org/rfc/rfc8826.txt>.
- [10] SeleniumHQ Browser Automation. <https://www.selenium.dev/>.
- [11] TCPDUMP/LIBPCAP public repository. <https://www.tcpdump.org/>.
- [12] WebRTC. <https://webrtc.org/>.
- [13] WebRTC TURN server. <https://webrtc.org/getting-started/turn-server>.
- [14] Spoify removes peer-to-peer technology from its desktop client. <https://techcrunch.com/2014/04/17/spotify-removes-peer-to-peer-technology-from-its-desktop-client/>, 2014.
- [15] Streaming video - from megabits to gigabytes. <https://www.ericsson.com/en/reports-and-papers/mobility-report/articles/streaming-video>, November 2018.
- [16] douyu-p2p-block. <https://github.com/jmz331/douyu-p2p-block>, 2019.
- [17] Akamai network deployment. <https://www.akamai.com/us/en/about/facts-figures.jsp>, 2021.
- [18] Bogon Filtering. https://en.wikipedia.org/wiki/Bogon_filtering, Apr. 2021.
- [19] Content Delivery Network:CDN Mesh Delivery—Lumen. https://www.lumen.com/en-us/edge-computing/mesh-delivery.html?utm_source=Streamroot&utm_medium=link&utm_campaign=Streamroot-transition, 2021.
- [20] IPinfo.io: Comprehensive IP address data, IP geolocation API and database. <https://ipinfo.io/>, 2021.
- [21] NerdyData.com: Search The Web's Source Code. <https://www.nerdydata.com/>, 2021.
- [22] Peer5. <https://www.peer5.com/product/>, 2021.
- [23] PublicWWW.com: Search Engine for SourceCode. <https://publicwww.com/>, 2021.
- [24] Security - Peer5 P2P Docs. <https://docs.peer5.com/security/>, 2021.
- [25] SimilarWeb - SimilarWeb Traffic Analysis. <https://www.similarweb.com/>, 2021.
- [26] Viblast PDN. <https://viblast.com/pdn/enterprise/>, 2021.
- [27] Web archive for Peer5's homepage on March 5th, 2021. <http://web.archive.org/web/20210305133908/https://www.peer5.com/>, 2021.
- [28] Commercial P2P CDN. <https://www.marketresearch.com/Global-Industry-Analysts-v1039/Commercial-P2P-CDN-33963948/>, 2023.
- [29] Project Zero. <https://googleprojectzero.blogspot.com/2018/12/adventure-s-in-video-conferencing-part-1.html>, 2023.
- [30] Nasser Mohammed Al-Fannah. One leak will sink a ship: WebRTC IP address leaks. In *2017 International Carnahan Conference on Security Technology (ICST)*, pages 1–5. IEEE, 2017.
- [31] Kevin Allix, Tegawendé F Bissyandé, Jacques Klein, and Yves Le Traon. Androzoo: Collecting millions of android apps for the research community. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 468–471. IEEE, 2016.
- [32] Nasreen Anjum, Dmytro Karamshuk, Mohammad Shikh-Bahaei, and Nishanth Sastry. Survey on peer-assisted content delivery networks. *Computer Networks*, 116:79–95, 2017.
- [33] Diogo Barradas, Nuno Santos, Luís Rodrigues, and Vítor Nunes. Poking a Hole in the Wall: Efficient Censorship-Resistant Internet Communications by Parasitizing on WebRTC. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 35–48, 2020.
- [34] Hugo LJ Bijmans, Tim M Booi, and Christian Doerr. Inadvertently making cyber criminals rich: A comprehensive study of cryptojacking campaigns at internet scale. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1627–1644, 2019.
- [35] William Conner and Klara Nahrstedt. Securing peer-to-peer media streaming systems from selfish and malicious behavior. In *Proceedings of the 4th on Middleware Doctoral Symposium*, pages 1–6, 2007.
- [36] Simon Da Silva, Sonia Ben Mokhtar, Stefan Contiu, Daniel Négru, Laurent Réveillère, and Etienne Rivière. Privatube: Privacy-preserving edge-assisted video streaming. In *Proceedings of the 20th International Middleware Conference*, pages 189–201, 2019.
- [37] Willem De Groef, Deepak Subramanian, Martin Johns, Frank Piessens, and Lieven Desmet. Ensuring endpoint authenticity in WebRTC peer-to-peer communication. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 2103–2110, 2016.
- [38] Jérémie Decouchant, Antoine Boutet, Jiangshan Yu, and Paulo Esteves-Verissimo. P3LS: Plausible deniability for practical privacy-preserving live streaming. In *2019 38th Symposium on Reliable Distributed Systems (SRDS)*, pages 1–109. IEEE, 2019.
- [39] Prithula Dhungel, Xiaojun Hei, Keith W Ross, and Nitesh Saxena. The pollution attack in p2p live video streaming: measurement results and defenses. In *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, pages 323–328, 2007.
- [40] Shayan Eskandari, Andreas Leoutsarakos, Troy Mursch, and Jeremy Clark. A first look at browser-based cryptojacking. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 58–66. IEEE, 2018.
- [41] Alexandros Fakis, Georgios Karopoulos, and Georgios Kambourakis. Neither Denied nor Exposed: Fixing WebRTC Privacy Leaks. *Future Internet*, 12(5):92, 2020.
- [42] Attilio Fiandrotti, Rossano Gaeta, and Marco Grangetto. Simple countermeasures to mitigate the effect of pollution attack in network coding-based peer-to-peer live streaming. *IEEE Transactions on Multimedia*, 17(4):562–573, 2015.
- [43] Gabriela Gheorghe, Renato Lo Cigno, and Alberto Montresor. Security and privacy issues in P2P streaming systems: A survey. *Peer-to-Peer Networking and Applications*, 4(2):75–91, 2011.
- [44] Mikael Goldmann and Gunnar Kreitz. Measurements on the spotify peer-assisted music-on-demand streaming system. In *2011 IEEE International Conference on Peer-to-Peer Computing*, pages 206–211. IEEE, 2011.
- [45] Dick Hardt. The OAuth 2.0 authorization framework. Technical report, 2012.
- [46] Maya Haridasan and Robbert van Renesse. SecureStream: An intrusion-tolerant protocol for live-streaming dissemination. *Computer Communications*, 31(3):563–575, 2008.

- [47] Mohammadreza Hazhirpasand and Mohammad Ghafari. One Leak Is Enough to Expose Them All. In *International Symposium on Engineering Secure Software and Systems*, pages 61–76. Springer, 2018.
- [48] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on Bitcoin’s peer-to-peer network. In *24th USENIX security symposium (USENIX security 15)*, pages 129–144, 2015.
- [49] Ping Identity and W Denniss. OAuth 2.0 Token Binding. 2017.
- [50] Michael B Jones. The emerging JSON-based identity protocol suite. In *W3C workshop on identity in the browser*, pages 1–3, 2011.
- [51] Jian Liang, Rakesh Kumar, Yongjian Xi, and Keith W Ross. Pollution in P2P file sharing systems. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 2, pages 1174–1185. IEEE, 2005.
- [52] Jian Liang, Naoum Naoumov, and Keith W Ross. The Index Poisoning Attack in P2P File Sharing Systems. In *INFOCOM*, pages 1–12. Citeseer, 2006.
- [53] Eric Lin, Daniel Medeiros Nunes de Castro, Mea Wang, and John Aycok. Spoim: A close look at pollution attacks in p2p live streaming. In *2010 IEEE 18th International Workshop on Quality of Service (IWQoS)*, pages 1–9. IEEE, 2010.
- [54] Ge Ma, Zhi Wang, Miao Zhang, Jiahui Ye, Minghua Chen, and Wenwu Zhu. Understanding performance of edge content caching for mobile video streaming. *IEEE Journal on Selected Areas in Communications*, 35(5):1076–1089, 2017.
- [55] Ming Ma, Zhi Wang, Ke Su, and Lifeng Sun. Understanding content placement strategies in smarthrouter-based peer video CDN. In *Proceedings of the 26th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 1–6, 2016.
- [56] Apostolos Malatras. State-of-the-art survey on P2P overlay networks in pervasive computing environments. *Journal of Network and Computer Applications*, 55:1–23, 2015.
- [57] Pere Manils, Chaabane Abdelberri, Stevens Le Blond, Mohamed Ali Kaafar, Claude Castelluccia, Arnaud Legout, and Walid Dabbous. Compromising Tor anonymity exploiting P2P information leakage. *arXiv preprint arXiv:1004.1461*, 2010.
- [58] Yuval Marcus, Ethan Heilman, and Sharon Goldberg. Low-Resource Eclipse Attacks on Ethereum’s Peer-to-Peer Network. *IACR Cryptol. ePrint Arch.*, 2018:236, 2018.
- [59] Xianghang Mi, Xuan Feng, Xiaojing Liao, Baojun Liu, XiaoFeng Wang, Feng Qian, Zhou Li, Sumayah Alrwais, Limin Sun, and Ying Liu. Resident Evil: Understanding residential ip proxy as a dark service. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1185–1201. IEEE, 2019.
- [60] Xianghang Mi, Siyuan Tang, Zhengyi Li, Xiaojing Liao, Feng Qian, and XiaoFeng Wang. Your Phone is My Proxy: Detecting and Understanding Mobile Proxy Networks. 2021.
- [61] Prateek Mittal and Nikita Borisov. Information leaks in structured peer-to-peer anonymous communication systems. *ACM Transactions on Information and System Security (TISSEC)*, 15(1):1–28, 2012.
- [62] Guillaume Montassier, Thibault Cholez, Guillaume Doyen, Rida Khatoun, Isabelle Chrisment, and Olivier Festor. Content pollution quantification in large P2P networks: A measurement study on KAD. In *2011 IEEE International Conference on Peer-to-Peer Computing*, pages 30–33. IEEE, 2011.
- [63] Marius Musch, Christian Wressnegger, Martin Johns, and Konrad Rieck. Thieves in the Browser: Web-based Cryptojacking in the Wild. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–10, 2019.
- [64] H Parmar and M Thornburgh. Adobe’s real time messaging protocol. *Copyright Adobe Systems Incorporated*, pages 1–52, 2012.
- [65] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *proceedings of the Network and Distributed System Security Symposium*, 2019.
- [66] A Rahmatulloh, R Gunawan, and FMS Nursuwars. Performance comparison of signed algorithms on JSON Web Token. In *IOP Conference Series: Materials Science and Engineering*, volume 550, page 012023. IOP Publishing, 2019.
- [67] Andreas Reiter and Alexander Marsalek. WebRTC: your privacy is at risk. In *Proceedings of the Symposium on Applied Computing*, pages 664–669, 2017.
- [68] J. Rosenberg. RFC 5389 - Session Traversal Utilities for NAT (STUN). <https://tools.ietf.org/html/rfc5389>, 2008.
- [69] Roberto Roverso and Mikael Höggqvist. Hive. js: Browser-based distributed caching for adaptive video streaming. In *2014 IEEE International Symposium on Multimedia*, pages 143–146. IEEE, 2014.
- [70] Henning Schulzrinne, Anup Rao, and Robert Lanphier. Real time streaming protocol (RTSP), 1998.
- [71] S Da Silva, S Ben Mokhtar, S Contiu, N Daniel, R Laurent, and R Etienne. PrivaTube: Privacy-Preserving Edge-Assisted Video Streaming. In *international middleware conference*, pages 189–201, 2019.
- [72] Atul Singh, Miguel Castro, Peter Druschel, and Antony Rowstron. Defending against eclipse attacks on overlay networks. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, pages 21–es, 2004.
- [73] Prajakta Solapurkar. Building secure healthcare services using OAuth 2.0 and JSON web token in IOT cloud scenario. In *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, pages 99–104. IEEE, 2016.
- [74] Thomas Stockhammer. Dynamic adaptive streaming over http– standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144, 2011.
- [75] Haizhou Wang, Xingshu Chen, Wenxian Wang, and Mei Ya Chan. Content pollution propagation in the overlay network of peer-to-peer live streaming systems: modelling and analysis. *IET Communications*, 12(17):2119–2131, 2018.
- [76] Jason Weil, Victor Kuarsingh, Chris Donley, Christopher Liljenstolpe, and Marla Azinger. IANA-reserved IPv4 prefix for shared address space. *IETF Request for Comment*, 6598, 2012.
- [77] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. Livesky: Enhancing cdn with p2p. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 6(3):1–19, 2010.
- [78] Wei Yu, Corey Boyer, Sriram Chellappan, and Dong Xuan. Peer-to-peer system-based active worm attacks: Modeling and analysis. In *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, volume 1, pages 295–300. IEEE, 2005.
- [79] Ge Zhang, Wei Liu, Xiaojun Hei, and Wenqing Cheng. Unreeling Xunlei Kankan: Understanding hybrid CDN-P2P video-on-demand streaming. *IEEE Transactions on Multimedia*, 17(2):229–242, 2014.
- [80] Liang Zhang, Fangfei Zhou, Alan Mislove, and Ravi Sundaram. Maygh: Building a CDN from client web browsers. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 281–294, 2013.
- [81] Mingchen Zhao, Paarijat Aditya, Ang Chen, Yin Lin, Andreas Haeberlen, Peter Druschel, Bruce Maggs, Bill Wishon, and Miroslav Ponec. Peer-assisted content distribution in akamai netsession. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 31–42, 2013.
- [82] Roverli P Ziwich, Elias P Duarte Jr, and Glaucio P Silveira. Distributed mitigation of content pollution in peer-to-peer video streaming networks. *IET Communications*, 2020.