USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

Watch Out Your TV Box: Reversing and Blocking a P2P-based Illegal Streaming Ecosystem

Jungun Ahn, Sueun Jung, Seungwan Yoo, Jungheum Park,
and Sangjin Lee, *Korea University*

https://www.usenix.org/conference/usenixsecurity25/presentation/ahn

This paper is included in the Proceedings of the
34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the
34th USENIX Security Symposium is sponsored by USENIX.

ARTIFACT EVALUATED
usenix ASSOCIATION
AVAILABLE

Watch Out Your TV Box: Reversing and Blocking a P2P-based Illegal Streaming Ecosystem

Jungun Ahn, Sueun Jung, Seungwan Yoo, Jungheum Park*, Sangjin Lee*
Korea University

Abstract

Recent developments have led to the emergence of illegal streaming services that are difficult to detect because they are offered only to those who have purchased specific set-top boxes. Even when such infringements are identified, the hardware-based nature of the service makes tracking extremely challenging. This paper focuses on analyzing the services provided through one such device, EVPAD. Upon installation, as of January 19, 2025, EVPAD allows users to watch real-time broadcasts of 1,260 channels from 18 countries and access 24,934 VoD contents, including Netflix and Disney Plus, as well as locally produced broadcasts. Through reverse engineering and detailed analysis, we identified 131,175 unique users and 78 servers dedicated to providing illegal streaming services over the course of two months. Beyond copyright infringement, EVPAD devices distributed worldwide could be misused by operators for cyberattacks. This paper proposes two blocking strategies based on EVPAD's service characteristics to curb copyright infringement and track key service nodes to dismantle illegal services and prevent potential cyber threats.

1 Introduction

The aftermath of COVID-19 has led to a steady increase in the popularity of watching broadcast content globally. Consequently, broadcasting and media companies are expanding their markets by securing or directly producing content. Paid platforms, including over-the-top (OTT) services, have diversified and their reach has expanded into the online and mobile markets.

Increasing Illegal Streaming Services. In response to these changes, illegal platforms that attract consumers with financial incentives have emerged [33, 36, 38, 41]. These platforms typically secure broadcast content and operate pirate broadcasting services, generating revenue through hardware sales or illegal advertising instead of regular subscription fees. Their operational methods are diverse, ranging from streaming via web hard drives and websites to selling devices that require the installation of illegal software.

Historically, the most well-known form of illegal service has been unauthorized websites, offering the most convenient approach for both consumers and sellers. However, these sites can be easily shut down by tracking the domain or server IP, making detection and response relatively straightforward [39, 40]. Consequently, illegal services have adopted the more elusive peer-to-peer (P2P) method, which does not directly expose their operations on the Internet. Early P2P models, such as Napster and Gnutella in the early 2000s, operated on a centralized system, followed by the emergence of distributed P2P models like BitTorrent, which facilitated the distribution of illegal content [14]. Recently, to operate even more covertly, hardware-based illegal services forming closed groups have appeared, making them difficult to locate via standard websites or external networks. These illegal hardware devices spread easily through global online marketplaces, illegally distributing video content, including OTT, while simultaneously infringing copyrights in numerous countries. Among the various illegal streaming devices (ISDs) available, EVPAD has been chosen as the focus of our analysis due to its significant global presence. Existing studies [23, 24] indicate that EVPAD is one of the most popular ISDs, particularly in regions like Southeast Asia, where a notable percentage of consumers use such devices to access pirated content.

Investigation Cases. As user consumption patterns evolve and the scale of copyright infringement grows, nations are increasingly taking action against illegal content services.

In November 2020, the U.S. Department of Justice [34] convicted five individuals for operating Jetflicks, an illegal streaming service that offered a large library of pirated movies and TV shows to its subscribers. The operation caused significant financial harm to legitimate streaming services and content creators, with damages from such illegal activities

*Co-corresponding author.

potentially reaching hundreds of millions of dollars annually.

In April 2021, Europol [17] successfully dismantled a large-scale illegal streaming service with more than 2 million subscribers around the world. This service illegally offered paid TV channels and Video-On-Demand (VOD) content, generating approximately $2 million in monthly revenue. The operation, conducted jointly by Europol and Spanish police, led to the seizure of the crime syndicate's assets and the arrest of related individuals.

In 2023, the City of London's Police Intellectual Property Crime Unit (PIPCU) [35] identified four operators in London who were illegally streaming content by installing unauthorized software on standard IPTV (Internet Protocol Television) devices, with an estimated user base of 500,000. Although the device itself is legal, the use of unauthorized software complicates efforts to shut down the service without locating the source server, as the service can be easily restored on other devices.

In 2025, the Delhi High Court [10] granted an ex parte injunction in a copyright infringement suit filed by the Football Association Premier League Limited (FAPL) against several unauthorized IPTV services, including Yoghurt TV and Nova IPTV. These services were illegally streaming live Premier League matches via rogue apps and domain-masked websites. The court ordered domain name registrars, ISPs, and Indian government agencies to block access, disclose registrant data, and assist in enforcement. The case highlights how modern IPTV piracy leverages anonymity and redirection, requiring dynamic legal and technical countermeasures.

Potential Risks. These cases illustrate the significant impact of illegal streaming services on copyright enforcement. Additionally, our analysis of the EVPAD service operation revealed that malicious apps or remote control codes could be easily introduced by service operators or external attackers. Because these devices lack basic security measures—such as antivirus software, firewall protection, and proper enforcement of Android security policies—and are often distributed with weakened permission settings, malware infections and other attacks can reach end-user devices without needing to exploit additional vulnerabilities. Furthermore, since EVPAD devices are connected to home networks through routers, additional infiltration into nearby network devices is possible through lateral movement. Typically, home or office internal networks lack security devices, making them more vulnerable to additional attacks.

Therefore, ISD devices like EVPAD, which can be remotely controlled, can easily be converted into tools for large-scale cyberattacks. A similar attack in the past involved the Mirai botnet, which first emerged in 2016 and infected hundreds of thousands of Internet-of-Things (IoT) devices worldwide, such as routers and IP cameras, to carry out distributed denial-of-service (DDoS) attacks [6]. At its peak, Mirai generated about 1TB of attack traffic using 145,000 devices [9]. Based on the Mirai attack scenario and our observations of concurrent users across all channels over 30 measurements spanning two months—with a minimum of 13,438, maximum of 19,889, and an average of 16,997 users—we estimate that approximately 17,000 EVPAD devices are active at any given time. These devices could potentially generate up to 0.12TB of malicious traffic at peak.

In terms of detection and response, attacks originating from EVPAD devices are challenging to investigate promptly since the devices are individually purchased and installed by users, making it difficult to identify the initial attack vectors and origins. Addressing the copyright infringement issues through illegal streaming is not enough; proactive investigation is needed to prevent significant global cyber threats posed by these services.

This paper provides an in-depth analysis of EVPAD service apps to estimate the global user base and collect a list of servers used in service operations. Then, by identifying vulnerabilities, we propose technical countermeasures to block illegal services. This approach helps uncover the scale of the underground economy operating illegally and proposes solutions to eradicate illegal services facilitated by various ISDs.

The structure of this paper is organized as follows: Section 2 provides a comprehensive review of related research on tracking illegal streaming services. In Section 3, we delve into an in-depth analysis of the operational mechanics of the EVPAD service. Section 4 presents a detailed examination of the global distribution of user IP addresses and server lists, building upon the characteristics identified in the preceding section. Section 5 introduces two innovative technical approaches for blocking illegal streaming from a service network perspective. In Section 6, we explore potential attack scenarios where EVPAD devices could be exploited as cyber weapons. Finally, Section 7 offers concluding remarks and insights derived from our study.

2 Background and Related Work

In the realm of unauthorized content distribution, it is crucial to investigate the current state of illegal streaming and develop countermeasures. According to ACN Newswire [32] in 2018, various ISDs such as BossTV, Ubox, EVPAD, Lingcod, and Magic Box were widely used by consumers in Hong Kong. In response, Hong Kong Customs seized over 350 ISDs in a crackdown, all of which were prosecuted for copyright violations. A survey conducted by YouGov, commissioned by the Coalition Against Piracy (CAP) during this time, revealed that many consumers who purchased ISDs had canceled their legitimate paid TV subscriptions, indicating significant copyright infringement.

To improve user awareness and block access to illegal services, research is being conducted to identify and assess the

Figure 1: Actual photo of the EVPAD 10P device, which includes a small-sized set-top box, USB cable, and charger in the box.

risks associated with threats exposed on illegal websites or devices. Delos Santos et al. [11], based on surveys of illegal users, confirmed that most users are willing to risk exposure to cyber threats, such as personal information theft, to use illegal services. They then collected illegal streaming websites and assessed the risk level based on individual usage methods and frequency. Watters [42] analyzed malware and malicious advertisements embedded in three popular ISDs and 25 illegal sports streaming sites. In ISDs, they identified 243 pieces of malware in 115 Android apps. On websites, they detected 139 malicious ads during ten visits per site. This research highlights the severe cyber threats users are exposed to when indiscriminately using illegal services. However, it also points out the limitations in identifying the scale and origin of the services to effectively block them.

Additionally, research has been conducted to develop technical countermeasures against illegal streaming services. Hsiao et al. [20] investigated the ecosystem of free illegal sports streaming websites. This study collected links from popular aggregator sites and used the OpenWPM crawling tool to gather and analyze cookies, HTTP (Hypertext Transfer Protocol) requests, and JavaScript calls on these webpages. They also used the EasyPrivacy filter list to analyze tracking behaviors and identified phishing and malicious activities through canvas, font, and WebRTC fingerprinting.

The above-mentioned existing studies can all be applied to illegal services operated as websites; however, their application is limited for services like EVPAD, which operate as closed groups. Strengthening legal measures to block illegal services faces practical limitations when servers are relocated to other countries or the service operates across multiple countries worldwide. Therefore, effective use of these measures requires them to be complemented with additional technical countermeasures.

Huang et al. [21] proposed methods to detect and block illegal streaming device related services, extending beyond just websites. They analyzed well-known illegal streaming apps to extract common features from third-party libraries, control flow graphs (CFG), and network traffic. Based on these features, they proposed an automated method to identify and block illegal apps. However, this approach is less effective for apps distributed through less conventional means, such as sideloading from private or obscure sources like EVPAD's custom channels, which are not readily accessible or detectable through traditional app distribution networks. Consequently, this method cannot block apps that are already installed or distributed through such hidden networks.

In this paper, we analyze the global proliferation of illegal streaming devices through the EVPAD platform. Our study highlights the ongoing issue of digital content copyright infringement and the potential cyber threats exposed worldwide. By examining these aspects, we underscore the severity of such illegal services.

3 Reversing EVPAD Internals

This section describes the characteristics of EVPAD, as well as the overall network components and operational mechanisms that distribute illegal streaming services.

3.1 Key Components and Characteristics

The EVPAD (https://www.evpadpro.com) device as shown in Figure 1 is a small, palm-sized set-top box manufactured by EVPADPro, a company based in Hong Kong. Notably, its official website advertises not only EVPAD products but also a range of other ISDs, including Unblock, SVICloud, DIGIBox, and SuperBox. The site presents itself as an authorized distributor endorsed by each respective vendor. It further claims that users can get lifetime access to streaming services without advertisements following a one-time purchase, and asserts that the TV boxes it sells are legal. These marketing narratives are intended to reassure consumers and legitimize the use of such devices, despite their apparent role in facilitating unauthorized access to copyrighted content. These products are easily available for purchase worldwide through various sites such as AliExpress.

The EVPAD is based on Android, and initially, the APK files required to run streaming services are not installed on the device. To access live broadcast channels and VoD streaming, users must manually install the 'StarLive' and 'StarVoD' APK files from a separate website as instructed on the official website (https://6868hx.com). The types and versions of service apps installed vary slightly depending on the device model. The specifications of the EVPAD device and the app versions primarily analyzed in this study are listed in Table 1.

Interestingly, during the installation process from such unknown sources, the service applications are installed seamlessly without requiring any additional user interaction or explicit permissions. Upon further inspection, we found that

Table 1: Device specifications of the EVPAD used in this work, along with the names and version information of two Android apps for live streaming and VoD services.

Model	EVPAD-3Max (3P)
OS	Android 7.0 (Linux kernel 3.10.65)
CPU	Cortex-A53
RAM	4GB DDR3
Interface	LAN, USB 2.0 / 3.0, Wi-Fi, Bluetooth 4.2
APP	StarLive (20230523), StarVoD (20220401)

the global system setting for package installation from non-market sources (*install_non_market_apps*) was set to 1, indicating that side-loading from unknown sources is universally permitted on this Android 7-based device. This configuration suggests that the manufacturer has turned off a default security policy [16], potentially weakening the platform's intended protections.

Although our primary focus was on analyzing and tracking the 3P devices, we also conducted additional analysis on the more recent EVPAD 10P device. Unlike the 3P version, which used two separate applications for live broadcasting and VoD services, the EVPAD 10P integrates these functions into a single application called 'StarV10' (the version analyzed in this work is 20240821). Additionally, differences were observed in the packer applied to the APK and the domain system used for server operations. However, the core mechanisms for those services (e.g., P2P) and the communication methods with servers remain nearly identical to the previous version, with the libraries responsible for key functions being exactly the same. Therefore, the following sections focus on the 3P version, with changes specific to the 10P highlighted separately.

As an example of the changes between these two versions, there is a notable difference with respect to the side-loading process of service applications. While the installation procedure via a third-party website (https://6868jx.com) remains consistent across both devices, the EVPAD 10P runs on Android 12, introducing significant changes in the security model governing the installation of applications from unknown sources. Starting from Android 8, Google enforces a per-app permission model for side-loading, requiring users to explicitly grant installation privileges to individual apps [5]. However, on the 10P, all applications are implicitly granted this permission without user interaction—even for newly installed APKs—effectively bypassing Android 12's intended security protections. This behavior is likely due to firmware-level modifications by the manufacturer.

As of January 19, 2025, users who install the StarLive can watch 1,260 live channels from 18 countries worldwide in real time. These countries include Canada, Taiwan, the United Kingdom, the United States, and several Southeast Asian nations. In addition, StarVoD provides access to 24,934 pieces of recorded video content, such as movies and TV shows from various countries, and supports OTT content from major global platforms like Netflix and Disney Plus. Unlike typical set-top boxes, EVPAD allows unlimited use of these services without any additional streaming fees once the device is purchased and the service apps are installed.

Both the StarLive and StarVoD apps, which are crucial to the illegal services, use Android app packer, SecShell [25], to obfuscate the source code and prevent analysis. For the purpose of this study, we used Frida [18] to dump the executable code from the memory area, resulting in approximately 8MB of unpacked binary code for each app. Our analysis revealed that while most processes, such as device authentication and updates, are handled by Java code within the APK, the implementation of P2P communication for streaming relies on external native libraries accessed through the Java Native Interface (JNI).

The core library for StarLive is `libtvcore`, which is a P2P-based TV streaming code distributed by binstreamio [7], supporting various operating systems such as Android, macOS, and Windows. This library enables users to share live broadcast data in real-time with other connected users, thereby enhancing the service speed.

Notably, analysis of other ISDs listed on the EVPAD website (https://www.evpadpro.com) revealed that SVICloud and DIGIBox also rely on the `libtvcore` library to facilitate live streaming. This suggests that `libtvcore` library plays a central role in the ecosystem of P2P-based ISDs, powering a range of services across different brands while potentially contributing to the decentralized distribution of unauthorized broadcast content. In the case of StarVoD, it streams recorded video and speeds up downloads using P2P via the `libp2ptrans` library, supplemented by HTTP communication with file servers to enhance speed.

3.2 Operational Mechanisms

The two core apps of the illegal service, StarLive and StarVoD, have many similarities in their operation. From authentication to verify legitimate users and devices, to the process of checking and playing the list of streaming content, they operate in a similar manner. Specifically, there are differences depending on the characteristics of the content that each app streams. Figure 2 illustrates this overall operation process.

3.2.1 StarLive: Live TV Streaming Service

StarLive broadcasts TV channels from various countries in real time. It follows a step-by-step procedure to authenticate devices and ensure service availability. First and foremost, the service operates exclusively on EVPAD devices. Although the StarLive can be installed on any Android device, streaming will only work on EVPAD devices. To achieve this, StarLive

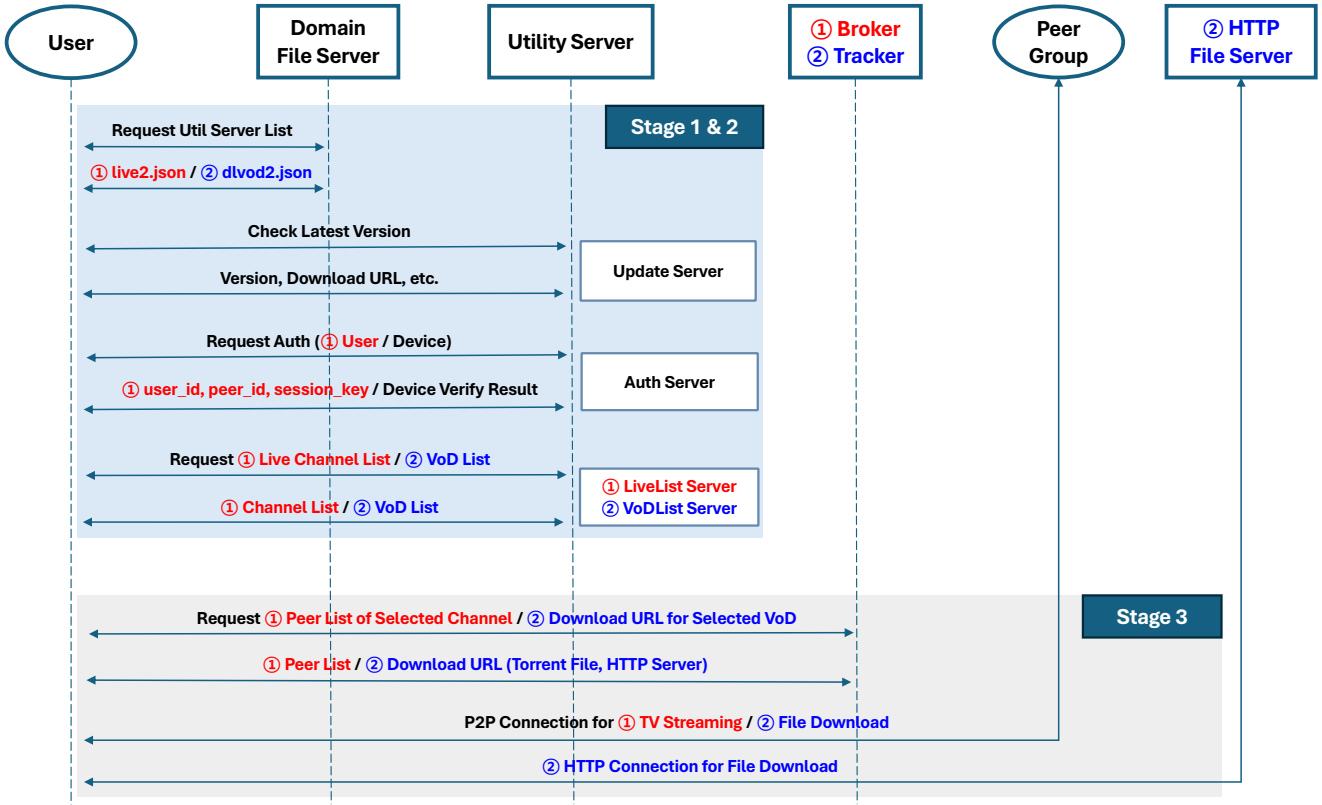

Figure 2: The overall operation of the EVPAD Services. In each process, the **common** elements of the two apps are shown in black, while elements specific to ① **StarLive** and ② **StarVoD** are shown in red and blue, respectively. Both apps operate in three main stages. **Stages 1** and **2** are executed by the Java code within each service app, and **Stage 3** is executed by referencing external native libraries in the form of JNI. StarLive uses `libtvcore.so`, while StarVoD uses `libp2ptrans.so`.

includes a device authentication process that generates data based on the unique attributes of the device.

Once device authentication is completed, the service ensures that users can always watch broadcasts in terms of availability. To achieve this, each peer node group in the P2P communication structure includes data broadcasting nodes. These nodes can be categorized as servers rather than users, with at least one broadcasting server included in the real-time user groups for each channel at any given time.

Stage 1: Checking Utility Server Lists. When the StarLive is launched, it downloads a `live2.json` file from `v6livejs.google144.com`, which acts as the Domain File Server shown in Figure 2. This JSON file contains a list of servers (hereafter referred to as Utility Servers) that are used for authentication and updates during service operation. The downloaded file is encrypted with AES-128, and by reverse engineering StarLive, we obtained a hard-coded key to decrypt it. Upon decryption, the internal DNS (Domain Name System) table mapping the domains and IP addresses of the Utility Servers can be observed.

For StarLive, the Utility Servers include an update server, an authentication server, and a management server for the streaming channel list. When the app is launched, it first compares the current information with the latest version and hash values provided by the update server to perform any necessary updates. Following this, the device undergoes an authentication process. Once authenticated, the device requests the current list of available channels by country from the management server. When a user selects a channel from the list, the streaming begins. In total, there are 25 domains used as Utility Servers, with the major ones listed in Table 3 for reference.

Stage 2: Authentication & Checking the Channel Lists. After verifying the list of Utility Servers and completing the update process, an authentication step is performed to confirm that the device is a genuine EVPAD. Once StarLive verifies that the device is an EVPAD, it proceeds with authentication to register the user for the P2P network required for streaming.

User authentication is performed by accessing the `/v1/auth` page of the authentication server. In this process, the network interface controller's MAC (Media Access Con-

```
{
    "classify": "USA & Canada",
    "classifyId": 367,
    "icon": "",
    "list": [
        {
            "alias": "bbcnews/3000",
            "classifyId": 367,
            "dname": "bbcnews",
            "epgurl": "",
            "hd": "true",
            "icon": "https://findpic.00005555.cc/Pic/Box/5cf7ec0952cab.png",
            "id": "569",
            "note": "",
            "order": "3000",
            "timeshift": 0,
            "type": "",
            "url": "[{\"path\":\"tvbus://3aUfyXnmwV.....8WY25pCgvHCULuNtm\"}]"
        },
        ..(omitted)..
    ]
}
```


Figure 3: Example of channel information for 'USA & Canada' with all channels returned in JSON format.

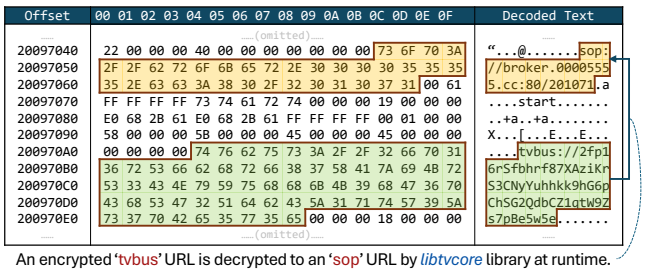

Figure 4: A running process's memory area containing an encrypted URL of the 'tvbus' protocol. Through the process, the encrypted data (`2fp16rSfbhrf87.....Zs7pBe5w5e`) is decrypted and converted into a 'sop' protocol's URL (`broker.00005555.cc:80/201071`), which include the Broker address and unique IDs for each channel.

trol) address is used to generate the *id*, while the *password* is created by concatenating the MAC address and a value labeled as 'CPUKEY' in the app, then hashing this concatenated data using the MD5 algorithm. Upon successful authentication, the server issues a `user_id`, `peer_id`, and `session_key`. These credential details are used as individual node identifiers during the subsequent P2P communication for TV streaming.

Subsequently, additional set-top box device authentication is performed through the `/stbauth` page of the authentication server to determine if the access is from an EVPAD device. The authentication data is constructed using the unique information values of the EVPAD device as follows: `Encrypt_AES( MAC | CPUID | CPUKEY | PKG )`. Here, the `MAC`, `CPUID`, and `CPUKEY` are device-dependent values, with `CPUID` and `CPUKEY` being labels assigned within the app. The `PKG` is the full package name of the currently running service app, such as `com.yby.live.v6.star` for the StarLive and `com.yby.v10.chaoneng` for the StarV10. These unique identifiers are concatenated and then encrypted using the AES-128 algorithm, with the encryption key hard-coded into the app itself. This process ensures that the device is recognized as a genuine EVPAD unit, securing the integrity of the service.

Once the user and device are successfully authenticated, the StarLive communicates with the management server to retrieve the TV channel list. The management server responds with the available channel information in JSON format as shown in Figure 3, which includes the channel name, channel address, and other details categorized by country. The channel addresses are returned in an encoded format and use a custom protocol ('tvbus') URL.

Stage 3: Streaming with P2P. When a user selects a channel from the provided list, the `libtvcore` library converts the channel address ('tvbus') using DES and Base58 algorithms. As a result, when examining the process memory, without the need for a separate detailed decryption analysis, a custom proprietary 'sop' protocol-based URL (`sop://[broker_address]/[channel_id]`) that includes the channel's unique ID is generated, as shown in Figure 4.

Based on the generated 'sop' protocol address, the user extracts `channel_id` and requests to register with the peer group for the specific channel. The Broker handles the overall tasks of registering new nodes, managing the status of nodes within each group, and more. Upon successful registration, the Broker returns the peer list for the channel in JSON format, as shown in Figure 5. This list includes information such as the `external_ip`, `external_port`, `internal_ip`, `internal_port`, `peer_id`, and the address of the broadcasting server (`sn_ip`) for the channel.

Once registered, the node uses TCP/UDP protocols to communicate with other nodes and access the streaming service. The `libtvcore` library defines the P2P communication packet structure, which includes a type field within the data structure format that specifies the role of each packet. Depending on this value, the packets facilitate various operations such as mutual authentication and data block exchange.

3.2.2 StarVoD: Video-On-Demand Streaming Service

StarVoD streams unauthorized video content, including movies and TV shows from around the world, with a focus on content from platforms like Netflix and Disney Plus, as well as various other OTT platforms. Similar to StarLive, it performs device authentication but does not require a separate user authentication process. This is because the P2P communication is implemented using the torrent protocol, which does not need individual identifiers for user nodes such as `user_id` and `session_key`.

After authentication, when a user selects a VoD title to watch, the app retrieves the address for downloading the torrent file. Simultaneously, it obtains the address of a separate video file download server, using both torrent and server download methods to ensure service availability. This approach differs from StarLive, which always includes at least one

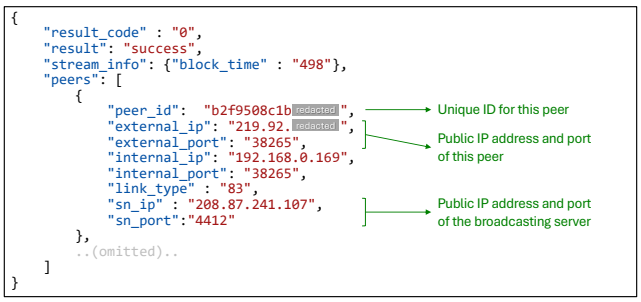

Figure 5: JSON-formatted information on each peer node that is participating in a channel. All peer information is returned as a list under the `peers` key.

broadcasting server in the peer group.

Stage 1: Checking Utility Server List. StarVoD also checks the list of Utility Servers by accessing the Domain File Server at `vodjs.google144.com` to download the `dlvod2.json` file. This JSON file is encrypted using the same method and key as used for StarLive. Decryption reveals a total of 52 domains, which include authentication servers, update servers, and video download servers. For reference, in the case of StarV10, a `dn.json` file from `jsten.googlejson.cc` contains the domains for both live and VoD services.

One notable characteristic of the Utility Servers used to provide StarVoD services is the high proportion of download servers. Out of the 52 domains, 40 were identified as download servers. A lookup of their IP addresses using a cyber threat intelligence (CTI) search engine, Criminal IP [3], revealed that most of them are cloud servers operating content delivery network (CDN) services. They distribute torrent files and video content, and the presence of multiple similar domains suggests that they are designed to quickly circumvent domain blocks. Additionally, there are domains and IP addresses for Trackers to facilitate torrent-based communication.

Other major Utility Servers include authentication servers and update servers, similar to those used by StarLive. The key servers are listed in Table 3 for reference.

Stage 2: Authentication & Checking VoD List. StarVoD, like StarLive, checks for updates and then performs device authentication. This process involves accessing the `/stbauth` page of the authentication server identified in the Utility Server list to verify the EVPAD device, without performing separate user authentication. The user node registration for the P2P network formed by torrent files is handled through the Tracker. The parameters for device authentication are similar to those of StarLive, but with a different AES key: the encryption key is randomly selected from eight possible keys that are hard-coded in the app. For authentication, the index of the chosen key and the encrypted authentication data are sent to the server.

After completing device authentication, the VoD list is retrieved. StarVoD categorizes content into main categories such as 'Movies' and 'TV', with further subcategories like 'Latest Uploads' and 'By Country'. Notably, within the Movies and TV categories, there are subcategories named '**Nflix**' for Netflix content and '**D+**' for Disney Plus. Each main and subcategory has a unique identifier (ID) used when requesting a corresponding list from the server.

To request each VoD list by subcategory, StarVoD accesses the `/quartz6/file/vodspe/` page of the VoDList Server, downloading an encrypted data file with the following filename format: `[pkg_id]_[main_id]_[sub_id].txt`. The `pkg_id` is determined by the type and version of the running service app, while the `main_id` and `sub_id` refer to the IDs for the main category and subcategory, respectively. The downloaded file is encrypted using AES-128 in ECB mode. Once decrypted, the VoD list appears in JSON format. This list includes basic information such as the video name, availability of subtitles, rating, and an ID value used as a unique identifier for each video.

By changing the ID values for the main and subcategories, we collected all VoD lists, resulting in a total of 24,934 pieces of content. Among these, 1,052 movies and TV shows were included in the 'Nflix' category. The content being illegally streamed represents approximately 15.5% of Netflix's total available titles, based on the fact that 1,052 out of approximately 6,801 titles are involved in the unauthorized activity [31]. Given that Netflix is one of the most popular OTT platforms worldwide, the inclusion of these 1,052 titles is expected to lead to significant copyright infringement.

Stage 3 : Streaming with P2P & HTTP. When a user selects a VoD to watch, a torrent file and an HTTP file server corresponding to the VoD title are identified using the VoD's ID value, as shown in Figure 2. The torrent file is encrypted using XOR, and the `libp2ptrans` library decrypts this file to perform standard torrent-based P2P communication. The Tracker is then requested to register the P2P group, and in response, it returns a list of communicable peers.

Simultaneously, the user engages in both P2P communication with other peers and HTTP communication with the file server delivering the selected VoD title. This dual approach ensures both downloading and streaming, but in practice, HTTP streaming via a dedicated file server significantly enhances service availability and playback speed, often playing a major role in video streaming.

4 Estimating the Scale of EVPAD Ecosystem

In this section, based on the operational mechanisms analyzed in Section 3, we introduce a method to estimate the global

scale of service usage by understanding the number of users and the status of servers.

4.1 Strategy of Estimation

4.1.1 Overview and Considerations

For StarLive, a Broker manages P2P communication groups for each channel and returns a list of peer node IP addresses whenever a new registration request occurs. By performing requests for lists of all channels, it is possible to determine the number of users utilizing the service at a given time. By considering time zone differences and consistently collecting user IP addresses over the course of two months, we estimated the number of users worldwide. For StarVoD, due to the higher proportion of HTTP communication with single servers compared to P2P communication, we used the user data collected only through StarLive services for our estimates.

Additionally, the data returned by the Broker includes information about the broadcasting server node for each channel. This allows us to determine the number and locations of servers used for operating StarLive. Furthermore, by aggregating information of the Utility Servers identified in Section 3.2 for both StarLive and StarVoD, we can ascertain the status of the core servers involved in the illegal services provided by EVPAD.

4.1.2 Automating Real-time Network Monitoring

To collect and analyze the real-time user data for StarLive, we developed a script for automation in the Windows environment. This script automates the peer registration requests to the Broker for all broadcasting channels. Instead of using the `libtvcore` library employed by EVPAD devices, we utilized `tvbus.exe`, an executable binary file for accessing the 'tvbus' protocol, which is a publicly available from a GitHub repository [7].

The necessary data for this process, including authenticated user session values, tvbus's channel addresses, and Broker addresses, were collected by analyzing the StarLive on an EVPAD device. By patching the memory of a running `tvbus.exe` process, we successfully communicated with the Broker in the Windows environment instead of using EVPAD devices and retrieved the list of real-time user IP addresses for the requested channel addresses.

To repeat the same operation for all channels, we developed an IDAPython script for dynamically patching memory and registers. Then the `tvbus.exe` process was executed using the IDA Pro's Debugger, and the IDAPython script patched relevant memory and registers with valid values obtained from real devices. These values corresponded to functions for user authentication and Broker requests.

After obtaining the peer list for a specific channel, the code flow was forcibly diverted back to before the channel registration request to repeat the same operation. Although the session

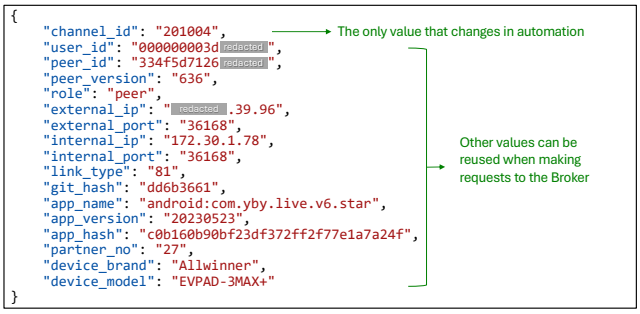

Figure 6: Request data for a specific channel to the Broker collected from an EVPAD device. As confirmed in Section 3.2, the `user_id` and `peer_id` are data issued during the user authentication process, and all field values can be reused when making requests to the Broker.

values (`user_id`, `peer_id`, etc.) of the requesting node are required each time, the verification process was weak, allowing the session data initially issued by the real device to be used indefinitely. Based on the JSON data shown in Figure 6, we sequentially proceeded with registration requests to the Broker for all channels by only changing the `channel_id` field. The pseudo code for the IDAPython script developed to collect peer lists is shown in Appendix A.

For reference, we observed that not only EVPAD but also other ISDs such as SVICloud and DIGIBox load the `libtvcore` library into the memory space of their primary streaming processes. Applying our instrumentation and analysis scripts, originally developed for EVPAD, yielded identical results on both SVICloud and DIGIBox, without any modification. This confirms that these devices share a common core architecture and exhibit uniform runtime behavior with respect to P2P-based stream distribution. Given this architectural consistency, it is likely that other TV box models leveraging similar P2P libraries may also be amenable to the same analysis and mitigation techniques, suggesting broader applicability of our findings.

4.2 Distribution of P2P Peers and Servers

In this study, we collected live streaming peer lists, covering 1,260 broadcasting channels across 18 countries. It was conducted consistently over the course of two months.

Peer Nodes. The information collected from peers includes the `external_ip`, `internal_ip`, and `peer_id`, as shown in Figure 5. Among these, the `peer_id`, as examined in Section 3.1, is dynamically generated by the server during the *user authentication* process each time the EVPAD service is started. Due to this dynamic nature, the `peer_id` changes with each application restart, making it unsuitable as a fixed identifier for estimating the number of users.

Table 2: Distribution of EVPAD users in the top 10% of countries. Notably, a significant portion of users are from East and Southeast Asian countries (✓) such as Malaysia, Taiwan, Hong Kong, and Singapore.

Service Users by Country (Top 10%)			
✓Malaysia (47,283 / 36.05%)	✓Taiwan (29,227 / 22.28%)	✓Hong Kong (11,376 / 8.67%)	✓Singapore (8,837 / 6.74%)
Canada (5,619 / 4.28%)	Australia (4,680 / 3.57%)	United States (4,145 / 3.16%)	✓Japan (3,813 / 2.91%)
✓Indonesia (3,519 / 2.68%)	United Kingdom (2,587 / 1.97%)	✓South Korea (1,924 / 1.47%)	✓Vietnam (1,540 / 1.17%)

To address this limitation, we utilized the `external_ip` and `internal_ip` to identify the number of users as accurately as possible. The `external_ip` represents the public IP, while the `internal_ip` is the private IP used within a local network. However, since public IP addresses are dynamically assigned through Dynamic Host Configuration Protocol (DHCP), they are not reliable for estimating the scale of active devices. Therefore, following the approach proposed in previous studies, we considered IP addresses sharing the same C-class prefix as belonging to a single local network [8, 29, 30].

Additionally, we utilized private IP addresses collected from individual devices to distinguish between multiple devices within the same local network, allowing us to include these devices in the user count [37]. That was because our empirical analysis demonstrated that internal IP addresses exhibit greater diversity than commonly expected. After deduplicating all internal_ip values, we identified 11,042 distinct internal addresses, with 8,109 of them falling within the 192.168.0.0/16 private address space. This level of variation indicates that internal_ip values are sufficiently heterogeneous to contribute to device-level distinction. By pairing each internal_ip with its corresponding external_ip truncated to a /24 prefix (i.e., C-class subnet), we constructed a composite identifier that enables reliable differentiation of individual devices even within the same public IP domain.

Consequently, our analysis revealed that a total of 131,175 unique peers were active in EVBOX's P2P network had over the two-month observation period.

Upon identifying the country associated with each IP address, we found that users were distributed across 116 countries [27]. The top 10% of countries by user count are listed in Table 2. Examining the user distribution by country, the majority of users were located in Asian countries such as Malaysia, Taiwan, Hong Kong, and Singapore. This aligns with the fact that most live streaming channels available were from the Southeast Asia region. Additionally, over 12,000 users were identified in countries like Canada, United States, and United Kingdom, indicating a significant number of viewers not limited to a specific continent but spread worldwide.

Broadcasting and Utility Servers. During the analysis of live streaming broadcast servers for users, a total of 35 servers were identified over a two-month period. Analysis of 35 collected server IP addresses revealed distribution across 10 distinct B-class (/16) network segments, indicating a broad and diverse address allocation strategy. IP geolocation checks confirmed that all servers were hosted in the United States. Further investigation using IP reputation services such as Criminal IP, which search for IP details like country, organization, and vulnerabilities, identified that all these servers were managed by CDN service providers: FDC Servers, Psychz Networks, OVHcloud, and ReliableSite.

From the analysis of both live and VoD services, a total of 43 Utility Server IP addresses were identified. The download servers used by StarVoD are part of cloud-based CDN services located in Japan, Singapore, Hong Kong and United States. In contrast, all other servers, including authentication and update servers, were located in Hong Kong and Singapore. Additionally, redundant servers with different IP addresses assigned to the same domain for backup purposes were located in Singapore. The detailed information, including the domains of these servers, is summarized in Table 3.

5 Blocking EVPAD Services

The services provided through EVPAD constitute a significant infringement on copyrights worldwide. The fundamental solution to blocking illegal streaming services involves identifying and seizing the operators or the physical servers through actual raids. In the case of broadcasting channel streaming, it is especially important to identify and confiscate the origin of the transmitter continuously sending data to prevent re-operation via new servers.

However, network tracking of servers distributed globally is a complex task that cannot be resolved quickly. It requires cooperation among various national agencies to map out the service network and identify the physical entities involved. Until these entities are seized through a full-scale international joint investigation, copyright infringement will continue over an extended period.

In this section, we propose two methods to block users' access to illegal services from a network perspective.

5.1 International Cooperation-based Approach for Blocking Core DNS Queries

Our analysis revealed that various Utility Servers are employed to facilitate service operations. These servers are accessed sequentially, with each server defined by its domain, and corresponding IP addresses distributed via separate DNS distribution files from the servers to replace DNS queries. Consequently, even if a domain is blocked to disable the service, updating the files can easily bypass this block.

Table 3: Operational server information identified by reversing service apps on EVPAD. Servers used for actual video distribution (i.e., live streaming and downloading) are primarily located in the United States and operate as cloud servers providing CDN services. Most of the servers that manage the services themselves are located in Hong Kong, Singapore .

Type	APP	Role	Country	Domain	Service
Streaming	**StarLive**	Live Streaming	**US**	-	CDN
Utility	**Common**	Update	**HK, SG**	evoptjapi.6868box.com, ptjapi6.ssv10dm.com	HTTP
		Authentication		evoapi.6868box.com, api6.ssv10dm.com	
	StarLive	LiveList		bslive.6868box.com, quartz6.ssv10dm.com	
		Broker		broker.00005555.cc, nbr24.googlebr01.com	
	StarVoD	VoDList		evoquartz.6868box.com, quartz6.ssv10dm.com	
		Tracker		dlt.6868nbtc.com	CDN
		Download	**JP, US, SG, HK**	nb0*.2268nb.com, s0*.23dfb.com, s0*.googlevod24.com, nbcloud0*.6868nb.com	

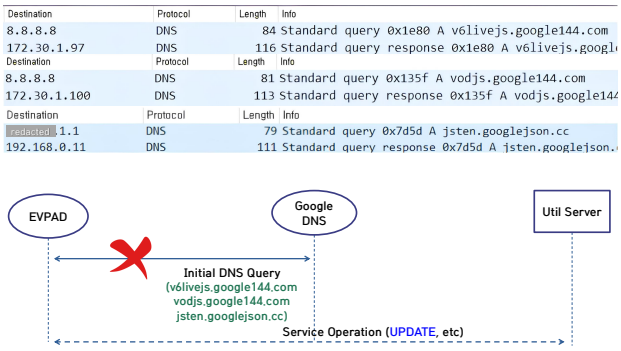

Figure 7: DNS queries for the Domain File Servers (i.e., `v6livejs.google144.com`, `vodjs.google144.com` and `jsten.googlejson.cc`) from the StarLive, StarVoD and StarV10. The initial DNS queries to retrieve DNS information for accessing the Utility Servers in the EVPAD ecosystem are directed to the DNS address configured on the network where the device is connected.

However, the initial domains required to fetch the DNS distribution file are hard-coded into the apps. As observed in Section 3.2, the domains `v6livejs.google144.com`, `vodjs.google144.com` and `jsten.googlejson.cc` are used, as illustrated in Figure 7, and DNS queries to these domains are unavoidable. By blocking these domains with the cooperation of internet service providers (ISP) in various countries, the utilization of all Utility Servers can be rendered ineffective. Although operators can update and redistribute the service apps, the updates themselves rely on the Utility Server addresses from these files, making proper updates impossible.

This method can be used to block both StarLive and StarVoD services, as they employ the same approach. However, the limitation is that it requires international cooperation and collaboration from relevant agencies, which may not be immediately feasible. Furthermore, this method is inherently limited in its effectiveness, as it can be circumvented if the application is redistributed through alternative channels between operators and users, outside the regular system update process. Therefore, we propose a second method that can immediately block users from accessing illegal streaming services.

5.2 Sybil Attack-based Aggressive Approach for Disrupting P2P Communications

Users of the service apps primarily engage in P2P communication. If the validity of the data exchanged between nodes is inadequately verified, an attacker can exploit this by launching a Sybil attack [1, 15, 28], which involves creating multiple fake identities to manipulate or disrupt the target network. Based on this hypothesis, we conducted a security analysis on the EVPAD's live streaming service and identified vulnerabilities that can be exploited to block illegal services through Sybil attack on its P2P network.

5.2.1 Vulnerabilities

Our analysis with regular user privileges revealed two critical vulnerabilities. They were verified using an Android emulator (NoxPlayer) and two actual EVPAD devices.

Bypass Authentication. The EVPAD service operates through set-top boxes, with an authentication server verifying whether each device is authorized. As noted in Section 3.2, this validation uses device-specific attributes, along with the service app type, version, and the current timestamp to form parameters for verification. As a result, installing the service apps on a typical Android-based device does not work.

During the authentication process, the service performs a single communication using fixed values based on device attributes, without incorporating random elements. Therefore, if valid attribute values required for authentication can be copied once, it is possible to bypass authentication on a regular device or emulator. We verified this vulnerability by

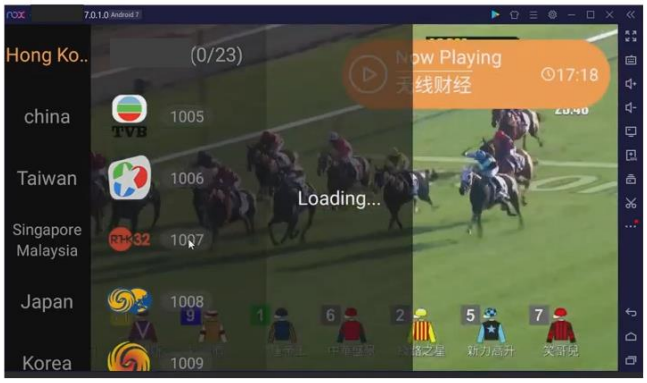

Figure 8: A screen that bypassed the EVPAD's device authentication process to run the StarLive using NoxPlayer for live broadcast streaming.

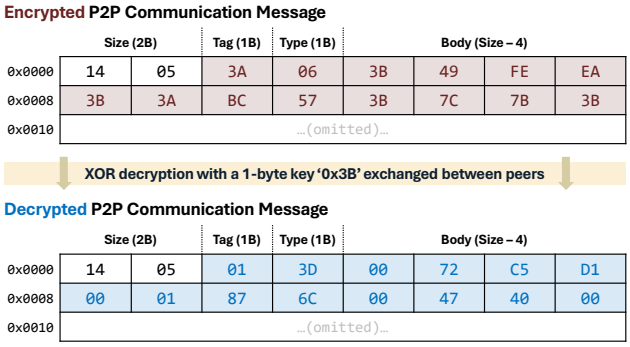

Figure 9: Structure of a P2P message. The example shown represents a data packet with `packet_type` 61 (0x3D), which corresponds to the `save_msg_put_block_data()` function in the `libtvcore` library. The upper 2 bytes of each message represent the size field, stored in plaintext. Following this, the body data, which performs specific functions based on the packet type, is sequentially arranged after the type field.

using the NoxPlayer to mimic an authenticated device. As shown in Figure 8, the emulator successfully accessed and streamed live channels from around the world. Exploiting this vulnerability allows for unlimited replication of service devices. Consequently, the number of users of the illegal service could increase exponentially, leading to significant copyright infringement.

DoS Attack. Authenticated users form peer groups with other users who are using the service in real-time. In this process, StarLive operates purely through P2P communication for streaming, while StarVoD uses HTTP communication with download servers in addition to P2P. Therefore, a P2P network attack aimed at disrupting the service would be more effective against StarLive.

While analyzing network packets during streaming via StarLive, we observed that all P2P packets were encrypted, and most packet structures exhibited similar patterns. Reverse-engineering the `libtvcore` library revealed the structure of these P2P communication messages, as illustrated in Figure 9. Each packet is encrypted using XOR-based encryption, leveraging a 1-byte key exchanged between peers beforehand. Notably, the upper 2 bytes, stored in little-endian format, represented the size of the packet and are stored in plaintext, while the remaining data is encrypted. Decrypting each packet is facilitated by the `speer_msg_decode()` function defined in the `libtvcore` library. This function takes two arguments: a `peer_tag` structure, which contains metadata about the communication node such as its IP address, port, and message type; and a `peer_data` structure, which includes the message body, its size, and the key used for encryption and decryption. Through analysis of these structures, we were able to extract the XOR key, enabling decryption of the packets. Further examination revealed that the second byte of the decrypted packet body (excluding the size field) define the type of the packet, while the remainder consists of body data specific to the functionality associated with the packet type.

Each peer validates incoming messages in the initial stage through the `speer_read()` function. Internally, `speer_read()` invokes the `recv()` function to verify the actual length of the received packet and compares it with the message field value specified in the upper 2 bytes of the packet data. If the values match, the `speer_msg_decode()` function decrypts the packet data, examines the packet type, and executes the predefined functionality if it matches one of the 11 defined types in the code. During this process, we discovered a vulnerability that enables immediate termination of the streaming service, confirming the feasibility of a DoS attack via this vector.

For experimentation, we used two EVPAD devices, both connected to the same live streaming channel. To minimize the impact on active services, we applied IP filtering via Frida, ensuring the test was confined to our devices. From one device, we crafted and sent a TCP socket-based packet to the other device, considering the previously analyzed packet reception process. Upon receiving this packet, the EVPAD streaming service on the target device immediately terminated. Further inspection revealed that this behavior was caused by a memory corruption vulnerability within the `libtvcore` library. As shown in Figure 10, a tombstone file was generated on the Android system immediately following the crash. Analysis of the file confirmed that an access violation had occurred, thereby validating the effectiveness of our DoS vector.

Given the critical nature of this vulnerability, we determined that even a single, carefully crafted TCP packet is sufficient for an individual to trigger service termination on a remote peer device. This drastically lowers the bar for potential abuse, as no significant bandwidth or coordinated effort is required. In light of the ethical implications, we have chosen

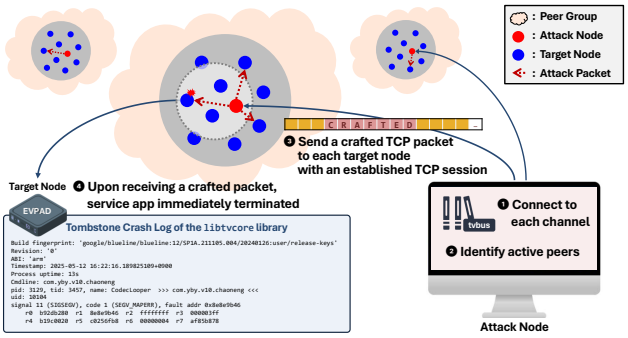

Figure 10: A scenario in which an attack node joins a specific P2P channel and sends a single crafted TCP packet to terminate the service application on a target node.

not to disclose the full technical details of the vulnerability or the exploit code in this paper. However, we are prepared to share this information with bona fide researchers and law enforcement personnel upon request, under appropriate disclosure and confidentiality agreements.

5.2.2 Procedures of the Proposed Attack

By leveraging the two identified vulnerabilities, deploying a substantial number of nodes into each channel to deliver exploit data to targeted users can now directly terminate their illegal streaming services. This leads to the development of a broad-based blocking strategy, outlined as follows:

❶ *Streaming Channel Listing:* Determine a list of TV channels available for streaming worldwide.

❷ *Attack Node Deployment in Channels:* Identify active peers present in each channel, create at least one *attack* node per channel, and connect these nodes to the corresponding channel.

❸ *Target Node Identification and Exploitation:* Obtain the IP address and port information used by the *victim* nodes for P2P communication, and then inject carefully crafted exploit packets to trigger immediate termination of the service.

❹ *Continuous Monitoring and Disruption:* Create a *monitoring* node separately from the *attack* nodes for each channel to periodically update the peer list via the broker and check whether the *victim* nodes have disappeared from the peer list as a result of the DoS attack.

By implementing this strategy, it is possible to cause significant disruption to the illegal streaming service without shutting it down entirely. Affected nodes will continue to face persistent issues even after rebooting and reconnecting, preventing normal access to streaming content.

5.2.3 Expectation

A blocking strategy that targets the P2P streaming network has several advantages over other methods, such as international cooperation to block services at the ISP level. This approach is both cost-effective and time-efficient, allowing for immediate application without incurring additional expenses. By exploiting the above-mentioned specific characteristics of the EVPAD ecosystem, this method can continuously block real-time streaming as long as it operates in its current form. Furthermore, by tracking the collected server lists and conducting on-site raids, this strategy can prevent copyright infringement of global broadcast content and contribute to the eradication of illegal services.

This proactive approach not only disrupts the current operations of illegal streaming services but also provides a scalable solution that can adapt to various types of ISD services. By maintaining pressure on these services and causing continuous disruptions, it is possible to deter users and reduce the overall demand for such illegal content. This, in turn, supports ongoing efforts to enforce copyright laws and protect the rights of content creators globally.

For reference, following the issuance of a court injunction on May 7, 2025, by the Delhi High Court (Order) [10], authorities launched an enforcement campaign to block several domains associated with EVPAD, SVICloud, and other ISDs. As a result, many of these services experienced temporary infrastructure disruptions lasting approximately four days. However, the EVPAD operator subsequently released an updated version of the StarV10 (dated May 20, 2025), which replaced all previously blocked domains with newly registered ones, effectively restoring full service. This sequence of events demonstrates that while authorities recognize the illegal nature of such services and are willing to implement aggressive ISP-level domain blocking, domain-based strategies alone face inherent limitations and are insufficient for long-term mitigation. In contrast, our proposed code-level intervention strategy offers a technically feasible and potentially more sustainable alternative that could significantly enhance enforcement capabilities if adopted by investigative agencies.

6 Discussion on Cyber Weaponization

In addition to the issue of copyright infringement through illegal streaming, EVPAD devices pose a significant and unique threat compared to traditional illegal streaming websites. Unlike websites, which can be shut down or blocked, they are customized embedded devices operating as independent network resources. This shift represents a more severe and pervasive risk that must be addressed with greater urgency, as these devices can be remotely controlled by service operators to function as *zombie* systems distributed worldwide. This section explores potential attack scenarios that could arise from EVPAD devices and examines possible countermeasures.

6.1 Threat Scenarios

To exploit each EVPAD device as a zombie system, an attacker would need to execute a backdoor process to open and listen on a port for remote control. This could be achieved during the service update process, where the operator could embed malicious code into a legitimate app package file and distribute it. The downloading and installation of service apps during updates are automatically performed by comparing the hash value and version information of the app responded by the server. Thus, the operator can install a backdoor on hundreds of thousands of devices worldwide without exploiting any additional vulnerabilities. Furthermore, as demonstrated by the SolarWinds Supply Chain Attack [4, 26], where external attackers compromised the update infrastructure to distribute malware globally, a similar hijacking of the update server here could have devastating security consequences.

To assess the feasibility of this attack scenario, we conducted an experiment in a local environment. During the EVPAD device's update process, we deliberately modified the server's response data to redirect the download URL to an attacker-controlled server. Interestingly, the update proceeded without any issues. Although the process includes a prompt asking the user to approve the installation, this behavior mirrors the legitimate EVPAD update flow and does not raise significant concerns. Notably, the update process lacked any mechanism to verify the integrity or authenticity of the downloaded APK—no signature or hash validation was performed. As discussed in Section 3.1, the system imposes no restrictions on the installation of additional applications, with all apps implicitly granted permission to install from unknown sources. Furthermore, our analysis revealed that the EVPAD device is distributed in a pre-rooted state, allowing any application to obtain superuser privileges. Combined with the fact that the device operates with SELinux in permissive mode—which allows policy violations to occur without enforcement—this significantly amplifies the potential impact of malware infections compared to standard Android environments.

This attack scenario involves not only the malicious distribution of data from the update server but also allows a third-party attacker to intercept update packets through a man-in-the-middle (MITM) attack. While the initial communication to receive update information is encrypted, the actual update file is downloaded over an unencrypted HTTP connection. Since there are no signature verification, integrity checks, or validity checks on the downloaded application software, an attacker could inject malicious code into the update package via MITM, thereby compromising the device with root privileges. These characteristics make the attack scenario not just theoretically possible, but particularly realistic and impactful compared to typical update mechanisms in more securely configured devices.

Once attackers have secured a global network of zombie resources, they can execute various attacks through remote control, with a large-scale DDoS attack being the most impactful. To expand the node resources for a DDoS attack, they can employ lateral movement to compromise other resources within the same network as the hijacked EVPAD devices. In typical households, where these devices are connected to a router hub for internet access, attackers can exploit vulnerabilities such as default password settings on IoT devices or other publicly known vulnerabilities to gain further access. Similar to the Mirai botnet [2, 19, 22], which utilized IoT devices for DDoS attacks on the Dyn DNS provider, an attack leveraging EVPAD devices could be constructed. Using an significant pool of resources, an attack through EVPAD could cause similar or greater international disruption, targeting major institutions and services. The overall attack scenario illustrating these steps is shown in Figure 11.

Furthermore, EVPAD devices could be weaponized as potent tools in state-level cyber conflicts, like the cyber warfare between Ukraine and Russia, by crippling command and control systems and critical infrastructure.

6.2 Countermeasures

Attacks utilizing EVPAD devices are challenging to detect and trace back to their origin due to the distribution through illegal service operation servers and the connection of purchased devices to users' private networks. The attacks are carried out within closed groups formed during service operations, rendering typical methods like blocking harmful websites ineffective against malware infections. Furthermore, DDoS traffic originates from the public IP addresses of users' networks, making it difficult to conduct practical investigations into the nodes connected to these home networks.

The effective response to such attack scenarios is to analyze detailed operations of the EVPAD services and block access to its self-hosted DNS server domains. As examined in Section 5.1, blocking these domains will prevent the service apps from performing updates, thereby hindering the creation of additional botnets. However, it remains practically impossible to remove malware from already infected individual devices.

While domain blocking can disrupt the service, it is not a fundamental solution unless their operation servers are shut down and seized. It can always be relaunched with a new domain, making domain blocking a temporary measure. Therefore, to prevent the worst-case global cyber threat scenarios, it is crucial to pursue an international collaborative investigation aimed at tracking and shutting down relevant servers and identifying operators behind the EVPAD ecosystem.

7 Conclusion

Most ISD devices like EVPAD, which are widely used globally, are purchased through overseas intermediaries, making it difficult to track down sellers or operators. Unlike legitimate

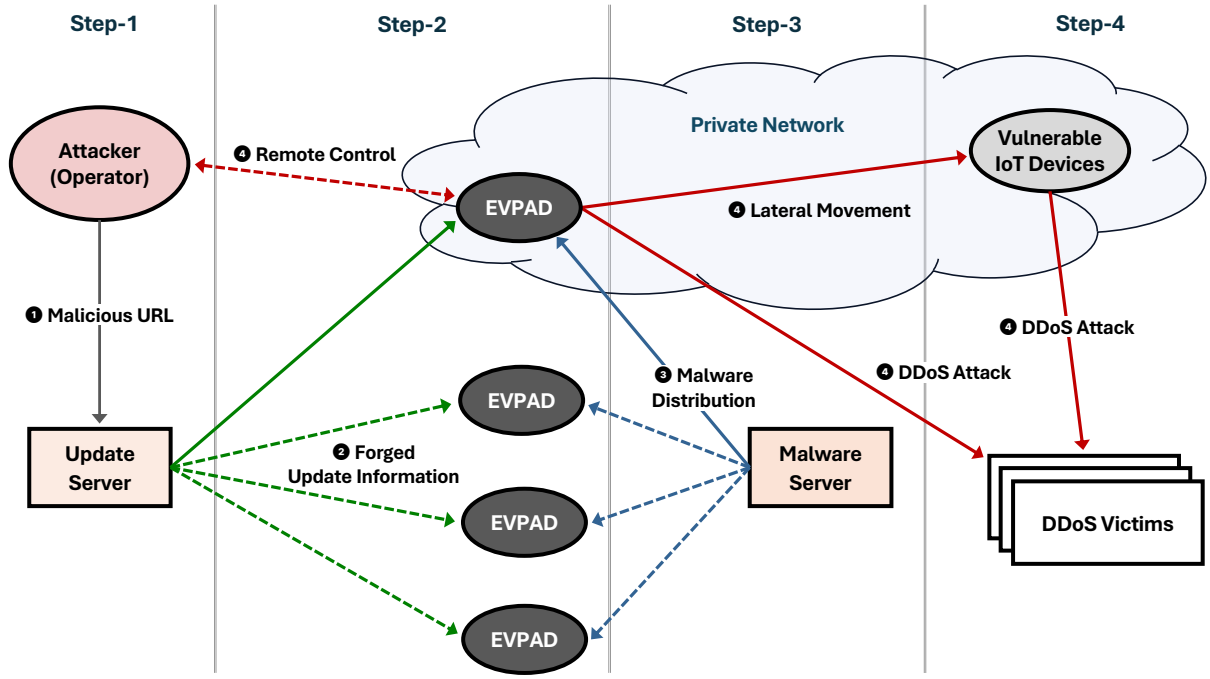

Figure 11: The attack scenario progresses through four main stages. **Step-1**: The attacker registers a malicious installation URL on the update server. **Step-2**: When the EVPAD service app runs, it references the tampered URL during the automatic update process. **Step-3**: Malware is distributed to all EVPAD devices from the attacker's server. **Step-4**: The attacker gains remote control through a backdoor, allowing further penetration into other nodes (IoT) connected to the home network via various publicly known vulnerabilities (CVEs). Ultimately, a DDoS attack is executed using all the compromised zombie nodes.

streaming services, these devices do not charge a subscription fee, making it hard to identify and trace payment accounts. Even if these devices are blocked, new devices can always be resold, allowing their services to continue operating. Recognizing these challenges, global copyright protection organizations have been actively working to address these issues, but illegal services that distribute broadcast content continue to emerge. To completely eradicate ISD-based illegal services, it is essential to identify and physically seize the core servers through coordinated raids. Particularly for addressing global copyright infringements like those involving EVPAD, international cooperative investigations are crucial, along with efforts to raise user awareness.

This paper analyzed two core EVPAD services, estimating the global user base and collecting the actual IP addresses of operational servers located in various countries. Based on the characteristics of the service operations, we proposed technical measures to effectively block illegal services. Other ISD services that adopt a P2P streaming model can also be analyzed and blocked using the methodologies proposed in this study. This research can serve as a foundation for the eradication of widely distributed ISD-based services globally.

With EVPAD devices, we identified 131,175 users across 116 countries and 78 operational servers located in the United States, Japan, Singapore, Hong Kong, and other countries. IP-based queries revealed that most servers operate as CDN services in a cloud configuration. Therefore, it is necessary to trace the network paths used by StarLive, StarVoD, and StarV10 for illegal content distribution, in cooperation with national ISPs and cloud hosting providers. More specifically, for StarLive, it is essential to identify the receivers located in various countries that transmit real-time broadcast content. For StarVoD, the focus should be on tracing operational servers responsible for recording and distributing content, especially considering that it also distributes content from national OTT platforms that restrict access to external countries.

Moreover, the issue with ISD devices extends beyond copyright infringement. Despite the absence of additional revenue sources such as advertising, EVPAD provides large-scale illegal streaming services, suggesting potentially malicious intentions behind the service. Even if malicious intent is not the primary motive, the lack of commitment to user security by such illegal operators places users in a vulnerable position, making them susceptible to attacks.

If services operating within closed groups in the underground world are not eradicated, they can easily be repurposed for various criminal cyber activities beyond streaming. Therefore, it is imperative to swiftly shut down these operations through coordinated efforts among relevant agencies and nations.

Acknowledgments

We sincerely thank the anonymous reviewers and our shepherd for their valuable feedback. This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2024-00460321; Development of Digital Asset Transaction Tracking Technology to Prevent Malicious Financial Conduct in the Digital Asset Market, No.RS-2023-00227165; Development of Binary-based Automatic SW Vulnerability Detection and Analysis Technique Applicable to Weapon-system Platform). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor.

Ethics Considerations

This research fully adheres to the USENIX Security's ethical guidelines. Throughout the study, we incorporated necessary ethical considerations and made a conscious effort to mitigate potential risks to third-party services.

The study analyzes an illegal streaming ecosystem that contributes to widespread copyright infringement and explores the potential for cyber-attacks within this ecosystem.

The experiments conducted were passive in nature, focusing on observing a single user's interactions with the service. While we recognized potential risks to actual services and users, within our controlled experimental environment, we conservatively verified that a packet generated by one device could minimally impact another device.

In our analysis, we encountered multiple vulnerabilities within the ecosystem, each presenting different levels of severity and implications. For example, the denial-of-service (DoS) attack can terminate illegal streaming sessions with a single crafted packet. However, it cannot escalate beyond this scope (e.g., arbitrary code execution). As such, it reveals a structural weakness that authorized entities could leverage to disrupt illegal services in a targeted and minimally invasive manner.

In addition, we discovered an authentication bypass vulnerability that enables unauthorized access to EVPAD's streaming services when emulated on Android-based platforms. This vulnerability allows full-service functionality without requiring the physical device, thereby directly undermining the service's access control mechanisms.

Based on our analysis, the manufacturer appears to be actively involved in the distribution and facilitation of unauthorized streaming content. As a result, we concluded that disclosing this vulnerability would be unlikely to mitigate harm and could result in the quiet removal of the exploit without addressing the illicit operation. We acknowledge that opinions may differ on the appropriate course of action, and we remain open to discussion regarding ethical alternatives.

To adhere to the Open Science Policy, we decided not to publicly release the JavaScript code used to exploit these vulnerabilities. This decision was made to minimize the risk of widespread misuse while still supporting the advancement of academic knowledge and responsible security research. Accordingly, we have decided to share our findings through restricted-access platforms that allow controlled dissemination to verified academic and legal stakeholders.

The solutions presented in Section 5 are not intended for individual use; rather, they serve as theoretical and practical tools that law enforcement agencies may leverage to combat illegal streaming activities. We have carefully designed these approaches to minimize collateral damage and avoid negatively impacting the quality of third-party services. Although the identified vulnerabilities do not pose immediate risks of large-scale disruption, they provide practical entry points for lawful interventions against illegal streaming networks.

The large-scale attack scenarios described in Section 6 are not directly related to the vulnerabilities discussed in Section 5. The vulnerabilities in Section 5 may support lawful operations targeting illegal services, but they do not pose immediate threats to users or infrastructure. By contrast, the scenarios in Section 6 focus on the risk of supply chain compromise, in which attackers could gain control of devices by targeting key infrastructure components. Even if the vulnerabilities were to be disclosed and patched, this would not fundamentally prevent the scenarios outlined in Section 6.

In conclusion, this study balances ethical and academic responsibilities while recognizing the potential risks of misuse. We have taken precautions to ensure that the insights shared contribute constructively to the academic community while minimizing risks to the public and the broader ecosystem.

Open Science

In accordance with the USENIX Security's Open Science policy, we have decided to release key artifacts to support future research efforts, provided they meet ethical and legal requirements. These include the source code used for data collection, processing, and analysis; measurement datasets where permissible; and scripts or configuration files necessary to replicate key experiments.

To ensure long-term accessibility and compliance with the artifact hosting policy, we have published our artifacts on Zenodo—a platform that supports permanent storage and access control. The artifacts consist of two types: ❶ *public* artifact that contains publicly shareable materials for reversing and understanding EVPAD ecosystem [12] and ❷ *restricted* artifact that contains sensitive materials including potentially dangerous exploits and privacy-sensitive peer data [13]. Recognizing the potential risk of misuse, we have adopted Zenodo's "Restricted Access" model for the *restricted* artifact. This model allows access requests to be reviewed and approved on a

case-by-case basis, enabling us to provide access to vetted researchers and authorities while reducing the risk of abuse.

Consistent with the ethical considerations discussed in our paper, we have chosen not to publicly release functional exploit code for the discovered vulnerabilities. While our paper includes sufficient technical detail to understand the nature, cause, and potential impact of the issue, we believe that releasing working exploit code would pose undue risks by enabling unauthorized access to similar services. This decision reflects a careful balance between openness and responsibility, aligned with both the spirit of open science and the need to prevent real-world harm.

Through these measures, we aim to contribute high-quality, reproducible research to the community, while maintaining our ethical responsibilities and complying with legal and policy constraints. We believe this approach appropriately supports both academic rigor and responsible disclosure.

References

[1] Gideon Adele, Abinash Borah, Anirudh Paranjothi, and Mohammad S Khan. A survey and comparative analysis of methods for countering Sybil attacks in VANETs. In *2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0178–0183. IEEE, 2024.

[2] Antonia Affinito, Stefania Zinno, Giovanni Stanco, Alessio Botta, and Giorgio Ventre. The evolution of Mirai botnet scans over a six-year period. *Journal of Information Security and Applications*, 79:103629, 2023.

[3] AI Spera Inc. Criminal IP: Cyber Threat Intelligence Engine. https://www.criminalip.io/.

[4] Rahaf Alkhadra, Joud Abuzaid, Mariam AlShammari, and Nazeeruddin Mohammad. Solar winds hack: In-depth analysis and countermeasures. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, 2021.

[5] Android Developers. Publish your app, 2023.

[6] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the Mirai Botnet. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1093–1110, 2017.

[7] BinStream. binstreamio. https://github.com/binstreamio, January 2018.

[8] Xue Cai and John Heidemann. Understanding block-level address usage in the visible internet. In *Proceedings of the ACM SIGCOMM 2010 Conference*, pages 99–110, 2010.

[9] Cloudflare. Inside Mirai: The Infamous IoT Botnet – A Retrospective Analysis. https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis, December 2017.

[10] Delhi High Court. The Football Association Premier League Limited vs. Yoghurt TV & Ors. https://delhihighcourt.nic.in/app/showFile/1746712133_681cb64548639.pdf/2025, May 2025. CS(COMM) 427/2025, Order dated May 7, 2025.

[11] Midsi Shreya V Delos Santos, Andreia D Etorma, Helisha A Ocampo, Adriel E Panjaitan, Juan Miguel B Romualdo, and Eric B Blancaflor. Risk Analysis of Home User's Vulnerability to Illegal Video Streaming Platform. In *Proceedings of the 4th International Conference on Management Science and Industrial Engineering*, pages 365–372, 2022.

[12] Digital Forensic Research Center (DFRC), Institute of Cyber Security & Privacy (ICSP), Korea University. Watch Out Your TV Box [Public Artifact]. https://doi.org/10.5281/zenodo.15646588.

[13] Digital Forensic Research Center (DFRC), Institute of Cyber Security & Privacy (ICSP), Korea University. Watch Out Your TV Box [Restricted Artifact]. https://doi.org/10.5281/zenodo.15602938.

[14] Choon Hoong Ding, Sarana Nutanong, and Rajkumar Buyya. P2P Networks for Content Sharing. Technical Report GRIDS-TR-2003-7, Grid Computing and Distributed Systems Laboratory, University of Melbourne, December 2003. https://ar5iv.org/abs/cs/0402018.

[15] Seda Dogan-Tusha, Saud Althunibat, and Marwa Qaraqe. Doppler-Shift-Based Sybil Attack Detection for Mobile IoT Networks. *IEEE Internet of Things Journal*, 11(1):1136–1147, 2023.

[16] André Egners, Björn Marschollek, and Ulrike Meyer. Hackers in your pocket: A survey of smartphone security across platforms. *RWTH Aachen, Tech. Rep. AIB-2012-07*, 2012.

[17] Europol. Illegal Streaming Service with Over 2 Million Subscribers Worldwide Switched Off. https://www.europol.europa.eu/media-press/newsroom/news/illegal-streaming-service-over-2-million-subscribers-worldwide-switched, April 2021.

[18] GuoQiang1993. GuoQiang1993/Frida-Apk-Unpack. https://github.com/GuoQiang1993/Frida-Apk-Unpack, April 2020.

[19] Ebu Yusuf GÜVEN and Zeynep GÜRKAŞ-AYDIN. Mirai Botnet Attack Detection in Low-Scale Network Traffic. *Intelligent Automation & Soft Computing*, 37(1):419–437, 2023.

[20] Luke Hsiao and Hudson Ayers. The price of free illegal live streaming services. *arXiv preprint arXiv:1901.00579*, 2019.

[21] Kong Huang, Ke Zhang, Jiongyi Chen, Menghan Sun, Wei Sun, Di Tang, and Kehuan Zhang. Understanding the Brains and Brawn of Illicit Streaming App. In *International Conference on Digital Forensics and Cyber Crime*, pages 194–214. Springer, 2021.

[22] Ping Identity. What Is a Botnet Attack and How to Prevent It. https://www.pingidentity.com/en/resources/cybersecurity-fundamentals/threats/botnet-attack.html, June 2023.

[23] Collin Jerome, Ting Su Hie, Ahmad Junaidi Ahmad Hadzmy, and Humaira Raslie. Accessing News in the Digital Era: The Case of Sarawak, Malaysia. *International Journal of Business and Technology Management*, 5(2):20–29, 2023.

[24] Ramon Lobato and Pradip Sarkar. The OTT TV box as a diasporic media platform. *Media Industries Journal*, 6(2), 2019.

[25] lxzh. lxzh/SecShell. https://github.com/lxzh/SecShell, March 2021.

[26] Jeferson Martínez and Javier M Durán. Software supply chain attacks, a threat to global cybersecurity: SolarWinds' case study. *International Journal of Safety and Security Engineering*, 11(5):537–545, 2021.

[27] MaxMind, Inc. GeoLite2 City Database. https://www.maxmind.com, 2024. Licensed under Creative Commons Attribution-ShareAlike 4.0 International License.

[28] Abolfazl Mehbodniya, Julian L Webber, Mohammad Shabaz, Hamidreza Mohafez, and Kusum Yadav. RETRACTED ARTICLE: Machine Learning Technique to Detect Sybil Attack on IoT Based Sensor Network. *IETE Journal of Research*, 69(10):clxx–clxxviii, 2023.

[29] Vikas Mishra, Pierre Laperdrix, Antoine Vastel, Walter Rudametkin, Romain Rouvoy, and Martin Lopatka. Don't count me out: On the relevance of IP address in the tracking ecosystem. In *Proceedings of The Web Conference 2020*, pages 808–815, 2020.

[30] Giovane CM Moura, Carlos Ganán, Qasim Lone, Payam Poursaied, Hadi Asghari, and Michel van Eeten. How dynamic is the isps address space? towards internet-wide dhcp churn estimation. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9. IEEE, 2015.

[31] Netflix. What We Watched the First Half of 2024. https://about.netflix.com/en/news/what-we-watched-the-first-half-of-2024, September 2024.

[32] ACN Newswire. The Upsurge in Hong Kong of Pirated TV Boxes Poses a Major Threat to the Subscription Video Industry. https://www.acnnewswire.com/press-release/english/44292/the-upsurge-in-hong-kong-of-pirated-tv-boxes-poses-a-major-threat-to-the-subscription-video-industry, July 2018.

[33] Alexios Nikas, Efthimios Alepis, and Constantinos Patsakis. I know what you streamed last night: On the security and privacy of streaming. *Digital Investigation*, 25:78–89, 2018.

[34] U.S. Department of Justice. Five Men Convicted of Operating Major Illegal Streaming Service. https://www.justice.gov/opa/pr/five-men-convicted-operating-major-illegal-streaming-service, November 2020.

[35] City of London Police. Four arrested during nationwide crackdown on the supply of illegal streaming services. https://www.cityoflondon.police.uk/news/city-of-london/news/2023/february/four-arrested-during-nationwide-crackdown-on-the-supply-of-illegal-streaming-services/, February 2023.

[36] M Zubair Rafique, Tom Van Goethem, Wouter Joosen, Christophe Huygens, and Nick Nikiforakis. It's Free for a Reason: Exploring the Ecosystem of Free Live Streaming Services. In *NDSS*, 2016.

[37] Philipp Richter, Florian Wohlfart, Narseo Vallina-Rodriguez, Mark Allman, Randy Bush, Anja Feldmann, Christian Kreibich, Nicholas Weaver, and Vern Paxson. A multi-perspective analysis of carrier-grade NAT deployment. In *Proceedings of the 2016 Internet Measurement Conference*, pages 215–229, 2016.

[38] Damjan Jugovic Spajic. Piracy Is Back: Piracy Statistics for 2024. https://dataprot.net/statistics/piracy-statistics/, February 2024.

[39] Alissa Starzak. The unintended consequences of blocking IP addresses. https://blog.cloudflare.com/consequences-of-ip-blocking/, December 2022.

[40] Hugh Stephens. Appeal Against Canada's First Successful Pirate Site-Blocking Order is Dismissed. https://hughstephensblog.net/2021/06/01/appeal-against-canadas-first-successful-pirate-site-blocking-order-is-dismissed/, June 2021.

[41] Ruoyu Wang, Yan Shoshitaishvili, Christopher Kruegel, and Giovanni Vigna. Steal This Movie: Automatically Bypassing DRM Protection in Streaming Media Services. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 687–702, 2013.

[42] Paul Watters. Scams, Cyber Threats and Illicit Sports Streaming in Singapore. *Available at SSRN 4709637*, 2024.

Appendix

A Pseudocode for Peer IP Address Collection

We used `tvbus.exe` to iteratively extract peer lists for all channels on the Windows platform. The binary file `tvbus.exe` provides access to the 'tvbus' protocol. Using IDA Pro's Debugger, we executed the `tvbus.exe` process as IDAPython scripts to patch memory and registers with values obtained from a real device. These values are associated with user authentication and broker request functions. Consequently, we successfully retrieved peer lists for each channel. Figure 12 shows the IDAPython scripts we developed.

```python
def RequestBroker(channel_id):
  global req_data # sample data shown in Figure 6.
  RefreshDebuggerMemory()
  ..(omitted)..
  BrokerURL = f"sop://broker.00005555.cc:80/{channel_id}"

  # Patch Memory for HTTP URL buffer.
  for i, char in enumerate (BrokerURL):
    patch_byte(broker_url + i, ord(char))
  patch_byte(broker_url + len (BrokerURL), 0x00)

  # change channel_id field in sample data.
  req_data['channel_id'] = channel_id
  continue_process()
```

```python
def GetPeerList():
  RefreshDebuggerMemory()
  ..(omitted)..

  # get returned peer_info data as shown in Figure 5
  BrokerResponse = get_string(get_reg_value("esi"))

  if "channel offline" in BrokerResponse:
    print("[+] Channel Offline")

  elif "external_ip" in BrokerResponse:
    # extract peer_nodes ip & server_node (sn) ip.
    peer_pat = re.compile(r'"external_ip": \s*" ([\d\.]+)"')
    sn_pat = re.compile(r'"sn_ip": \s*" ([\d\.]+)"')
    Peer_IP = peer_pat.findall(BrokerResponse)
    SN_IP = sn_pat.findall(BrokerResponse)

  # repeat RequestBroker Function.
  set_reg_value(ebp_backup, "ebp")
  set_reg_value(esp_backup, "esp")
  set_reg_value(RequestBroker, "eip")
  continue_process()
```

Figure 12: A simplified Python code excerpted from our developed IDAPython script that is executed in a debugger attached to the `tvbus.exe`. It automates peer registration requests (`RequestBroker`) and peer list collection (`GetPeerList`) to the Broker. Both functions are breakpoint handlers set in the code responsible for *Send* and *Receive* functionalities with the Broker, patching memory and register values using valid communication data collected from the EVPAD device.