# Watching You Watch: The Tracking Ecosystem of Over-the-Top TV Streaming Devices

Hooman Mohajeri Moghaddam, Gunes Acar, Ben Burgess, Arunesh Mathur, Danny Yuxing Huang, Nick Feamster*, Edward W. Felten, Prateek Mittal, Arvind Narayanan
Princeton University and University of Chicago*

## ABSTRACT

The number of Internet-connected TV devices has grown significantly in recent years, especially Over-the-Top ("OTT") streaming devices, such as Roku TV and Amazon Fire TV. OTT devices offer an alternative to multi-channel television subscription services, and are often monetized through behavioral advertising. To shed light on the privacy practices of such platforms, we developed a system that can automatically download OTT apps (also known as channels), and interact with them while intercepting the network traffic and performing best-effort TLS interception. We used this smart crawler to visit more than 2,000 channels on two popular OTT platforms, namely Roku and Amazon Fire TV. Our results show that tracking is pervasive on both OTT platforms, with traffic to known trackers present on 69% of Roku channels and 89% of Amazon Fire TV channels. We also discover widespread practice of collecting and transmitting unique identifiers, such as device IDs, serial numbers, WiFi MAC addresses and SSIDs, at times over unencrypted connections. Finally, we show that the countermeasures available on these devices, such as limiting ad tracking options and adblocking, are practically ineffective. Based on our findings, we make recommendations for researchers, regulators, policy makers, and platform/app developers.

## CCS CONCEPTS

• **Security and privacy** → *Privacy protections.*

## KEYWORDS

privacy; OTT; third-party tracking; measurement; Internet TV; automation

## 1 INTRODUCTION

The number of Internet connected TV users has increased steadily over the past few years and an estimated 65.3% of Internet users in the United States (US)—close to 182.6 million people—used an Internet connected TV device in 2018 [20]. In fact, "cord-cutting" patter, where users replace their traditional cable TV subscription with content delivered through Internet connected TV platforms to avoid long-term multi-channel subscription commitments, has been a very popular trend in recent years [77].


*CCS '19, November 11–15, 2019, London, United Kingdom*

ACM ISBN 978-1-4503-6747-9/19/11.
https://doi.org/10.1145/3319535.3354198

However, many Internet-connected TVs introduce privacy risks. These devices often have access to sensitive user data, e.g., microphone input, viewing history, and personal information. Not only could such data be exposed to developers who build applications (hereafter, *channels*) for these devices, but it could also be used for behavioral advertising. For example, the Federal Trade Commission (FTC) recently fined Vizio—a smart TV manufacturer—for collecting individual users' demographics and viewing histories for targeting advertising without consent [60]. Moreover, for manufacturers such as Roku, advertising has surpassed device sales as the primary source of income [38, 52, 56, 73]. In fact, we have found evidence of trackers collecting user identifying information and viewing behavior in Roku's network traffic, as shown in Figure 1.

In this paper, we examine the advertising and tracking ecosystem of Over-the-Top ("OTT") streaming devices, which deliver Internet-based video content to traditional TVs/display devices. OTT devices refer to a family of services and devices that either directly connect to a TV (e.g., streaming sticks and boxes) or enable functionality within a TV (e.g., smart TVs) to facilitate the delivery of Internet-based video content [25].

Specifically, we are interested in identifying endpoints that OTT channels contact to serve advertisements and/or track users; the entities that these endpoints are associated with; what information OTT channels send to them; and how they potentially track users. To this end, we build an automated system that collects channel information from OTT channel stores, loads individual channels on an OTT device, and attempts to play a video clip and trigger a video ad while capturing network traffic. In particular, we study two of the most popular OTT streaming devices in the market: the Roku Channel Store [67] and the Amazon Fire TV channel store [3]. We examine the network traffic of 1,000 channels from the Roku Channel Store and 1,000 channels from the Amazon Fire TV channel store. Automated analysis of these channels at scale presents two key challenges that previous studies examining third-party tracking at scale on the web [23] and mobile devices [59] have not encountered:

**Challenge 1: Automated Interaction:** Unlike web browsers and mobile devices, where visiting, launching and interacting with applications is enabled by existing automation software (e.g., Selenium [71]), to the best of our knowledge there is no such solution for OTT devices. Even though many OTT devices have a remote control API, the interaction is virtually limited to sending commands and coarse-grained feedback, such as the current channel viewed. The lack of fine-grained feedback, such as what text is shown on the TV screen or whether a video clip or video advertisement (ad) is being played on the TV, makes it difficult to execute the complex interaction required to play videos on channels. To mitigate this lack of feedback, we build a system that uses audio

```
HTTP outbound to 192.35.249.124:80(DNS: search.spotxchange.com)(channel name: asiancrush)

GET./vast/3.0/146141?VPI[]=MP4&VPI[]=ROKU&app[name]=asiancrush&app[domain]=asiancrush.com&
app[bundle]=com.dmr.asiancrush&player_width=1280&player_height=720&device[devicetype]=7&de
vice[make]=Roku&device[model]=Roku&device[ifa]=39fc6352-aede-53f6-b3e3-58bf562bd074&ip_add
r=128.112.139.195&cb=1557313464653&custom[movie_title]=So%20Young%202%3A%20Never%20Gone&cu
stom[content_id]=3417&token[device_id]=39fc6352-aede-53f6-b3e3-58bf562bd074&token[connecti
on]=wifi&token[category_ID]=241&token[category_Title]=Romance&device[dnt]=0&max_bitrate=70
00 HTTP/1.1
Host: search.spotxchange.com
User-Agent: Roku/DVP-9.0 (519.00E04142A)
Accept: */*
```

**Figure 1: AsianCrush channel on Roku sends device ID and video title to online video advertising platform `spotxchange.com`.**

and pixel content to infer the state of video playback on the TV, as we will explain in detail in Section 3.

**Challenge 2: Intercepting Traffic:** Unlike web browsers and mobile devices which expose capabilities to install root certificates to intercept HTTPS traffic, OTT streaming devices are largely closed and proprietary, making it harder to intercept and analyze encrypted traffic. To this end, we modify mitmproxy to intercept and decrypt HTTPS traffic via *best-effort*. Where possible, we deploy our own TLS certificate to the device and use external toolkits (e.g., Frida [29] for the Amazon Fire Stick TV) to bypass certificate pinning.

**Contributions:** We make the following contributions:

- We conduct the first large-scale study of privacy practices of OTT streaming channels. Using an automated crawler that we developed in house, we crawled more than 2,000 channels and found widespread practices of tracking.
- We build the first system that can automate the interaction of OTT channels and the interception of network activities. We will open-source the system, which can be used by other researchers to study similar OTT and smart TV platforms.
- We discover collection of persistent identifiers (such as WiFi MAC addresses and SSIDs) for tracking, at times over unencrypted connections. We find connection to at least one tracker on the traffic of 691 of the most popular 1,000 channels on Roku and 894 of the most popular channels on Amazon Fire TV.
- Analyzing the local remote control APIs of the OTT devices, we find a vulnerability that allows a malicious web script to extract a Roku user's location, retrieve installed channels, install new channels and access device identifiers. We reported the vulnerability to the vendor, who has rolled out a fix.

## 2 RELATED WORK AND BACKGROUND

In this section we review the related work, and we describe the OTT streaming devices we studied in our work.

### 2.1 Related Work

Numerous studies have examined different aspects of online tracking, ranging from understanding how tracking works, what entities engage in tracking, and the implications of tracking for consumer privacy. In this section, we summarize the related literature and place our study in context.

*2.1.1 Web Tracking: Techniques and Measurement.* Some of the earliest studies of online tracking began examining the presence and practices of third-party entities embedded on websites [36, 42, 63], finding pervasive use of third-party cookies to track users across websites. Successive studies discovered novel third-party tracking techniques that are difficult to counter. For example, studies have shown how users can be tracked on the Web using Flash cookies [74], browser fingerprinting [18], canvas fingerprinting [45], performance characteristics [44], font metrics [27] and installed browser extensions [31, 76].

While these studies illuminated several types of tracking mechanisms, only recently have researchers begun measuring and examining their behavior at scale. For example, Nikiforakis, et al. measured the prevalence of third-party JavaScript on the Alexa Top 10,000 websites, and identified new ways an adversary could compromise websites such as by registering stale domains that victim websites load scripts from [51]. In another instance, Libert used an instrumented headless browser to study third-party tracking and discovered that websites that leak user data contact an average of nine different third parties [39]. Finally, Acar et al. built FPDetective [2] to measure the prevalence of various fingerprinting techniques on the web over the Alexa Top 1 Million websites.

More recently, Englehardt and Narayanan designed and built OpenWPM—an instrumented web crawler based on Selenium—and used it to measure the prevalence of various kinds of browser fingerprinting on the Alexa Top 1 Million pages [23], discovering that a few large third parties are responsible for most tracking on the web. Other studies using OpenWPM have shown how web tracking exposes users to network surveillance by sending identifiers in the clear [24], and how nearly 30% of all emails leak users' email addresses to third parties [22]. More recently, Das et al. [10] found that several third parties in the Top 100K Alexa websites use sensor APIs—such as motion, orientation proximity and light—available in mobile browsers for tracking and analytics.

*2.1.2 Mobile Tracking.* Numerous studies have documented the third-party entity landscape of mobile applications. Privacy implications of mobile apps have been extensively studied in the literature [21, 32, 61]. Xia et al. showed that up to 50% of the smartphone traffic can be attributed to users' real names [82]. Reyes et al. analyzed 5,855 popular free children's apps and found widespread violation of the Children's Online Privacy Protection Act ("COPPA") [62]. Other work explored ways to track users on mobile

platforms by fingerprinting smartphone configurations [37], acoustic components [11], sensors [12, 13, 48], battery readings [54], and ambient light level [53]. Ren et al. presented ReCon [61], which detects potential PII leaks by inspecting network traffic and allows users to control dissemination. They ran a study with 92 participants to measure PII exposure on the 100 most popular iOS, Android, and Windows Phone apps.

*2.1.3 Privacy and Security of Smart Devices.* Analyzing 20 IoT devices, Loi et al. proposed a systematic method to identify the security and privacy issues of various IoT devices including home security, energy management and entertainment devices [40]. Fernandes et al. studied security of 499 SmartThings apps and 132 device handlers on Samsung's SmartThings platform and found security flaws in the framework [26]. Wood et al. studied four medical devices and found that one device occasionally sends sensitive health data in clear text [81]. Acar et al. used DNS Rebinding [14] to gather sensitive information and control IoT devices with local HTTP interfaces [1]. Finally, Malkin et al. [41] surveyed current and prospective smart TV users regarding their privacy understanding and expectations with regards to these platforms and concluded that there is very little transparency and understanding of privacy practices on these platforms.

## 2.2 Platforms and Channel Stores

We examined the advertising and tracking ecosystem of services present on two of the most popular OTT streaming device families: **Roku** and **Amazon Fire TV**. We specifically chose these two, since together they account for 59% to 65% of the market share globally [8, 19, 20]. Both device families consist of various external devices that users connect to displays (such as TVs) via HDMI. These devices stream video content over the Internet through channels (like apps). Users can download and install channels on their devices from the Roku Channel Store [67] and the Amazon Fire TV channel store [3].

Roku streaming devices run a proprietary operating system developed by Roku, Inc. The Roku channels are packaged, signed and encrypted to ensure confidentiality of the source code. Only Roku devices have the ability to decrypt the channels. As a privacy option, Roku allows users to "Limit Ad Tracking" to disable identifiers used for targeted advertising and Roku's developers documentation states that channels should not use the data collected from the device to serve personalized advertisements when this option is enabled [64].

Amazon Fire TV devices run a custom version of the Android operating system. Amazon Fire TV channels are packaged in the Android Application Package (APK) format, and—as with most Android devices—developers can interact with the Fire TV devices using the Android Debug Bridge (`adb`) tool[1]. Similar to Roku, Amazon allows users to "Disable Interest-based Ads" and limit behavioral profiling and targeting [4].

We performed our crawls on both Roku and Amazon Fire TV family of devices, more specifically using Roku Express [69] and the Amazon Fire TV Stick [5] since these are the most popular and least expensive options within each family. Note that while we crawled the channels from the channel stores using these specific devices, our crawler was built to be largely agnostic to the underlying device type. In future work, researchers can replace a small portion of our Roku- and Amazon-specific code with APIs of other OTT platforms.

[1] `https://developer.android.com/studio/command-line/adb`

# 3 SMART CRAWLER AND DATA COLLECTION

In this section, we describe our data collection pipeline: the list of channels we crawled, and our crawler infrastructure. We also discuss different settings and preferences we used for all the crawls we performed in our study.

## 3.1 Compiling Channel Lists

We compiled a list of channels from the Roku and Amazon Fire TV channel stores. We compiled these lists in May 2019.

*3.1.1 Roku Channel Lists.* Roku channels are organized by category on the Roku Channel Store website, with each channel belonging to only one category. To compile a list of channels, we extracted all the channels within each category. Each category page on the website (e.g., `channelstore.roku.com/browse/movies-and-tv`) contains a list of channels in that category along with each channel's metadata information—its ID, description, Roku's internal channel popularity ranking, and the identity of its developer—all of which we recorded. This resulted in a list of 8,660 channels across 23 categories.

To keep our crawls tractable while analyzing channels that users are more likely to encounter, we did not extract all the 8,660 channels. Instead, we created a new list, sorting the list of 8,660 channels by rank and retaining the top 1,000 channels (**Roku-Top1K**). In addition, to test various features of the OTT devices (e.g., privacy controls), we created a list of 100 channels (**Roku-Categories-Top100**) by selecting the top 10 channels by rank from the following categories: "Movies & TV", "Kids & Family", "Sports" , "Fitness", "Religious", "Food", "Shopping", "Educational", "Special Interest", and "News & Weather". We chose these categories since they contained the most channels overall.

*3.1.2 Amazon Fire TV Channel Lists.* Like Roku, Amazon Fire TV channels are organized as a list on the Amazon Fire TV channel store website, with some channels belonging to multiple categories. We recorded all the channels from this list, including each channel's metadata—its ID, description, Amazon Fire TV's internal channel popularity ranking, and the identity of its developer—resulting in a list of 6,782 channels across 29 categories.

As with the Roku channel lists, we retained the top 1,000 channels by rank (**FireTV-Top1K**) for crawling. In addition, we also created a list of 100 channels (**FireTV-CategoriesTop100**) by selecting the top 10 channels by rank from the following categories: "News & Weather", "Movies & TV", "Sports", "Lifestyle", "Health & Fitness", "Food & Drink", "Kids", "Shopping", and "Education". We chose these categories since they contained the most channels overall. Because some of these channels belonged to multiple categories, this list contained 86 channels in total.

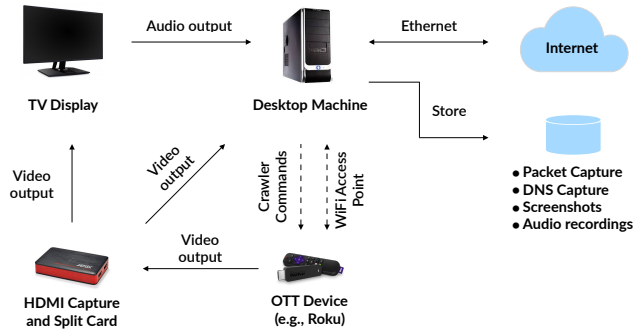



**Figure 2: Overview of our smart crawler.**

## 3.2 Smart Crawler Infrastructure

*3.2.1 Overview and Setup.* Figure 2 illustrates our smart crawler setup. Our crawler consists of four physical devices: a desktop machine, a TV display, an HDMI split and capture card, and the OTT device. The desktop machine executes the crawler code, orchestrates the crawl and stores the resulting data.

The desktop machine acts as a WiFi access point (AP) and its wireless network interface is bridged to the Internet. The OTT device connects to this AP, which allows us to capture the OTT device's network traffic. The OTT device outputs its video to both a TV display and a desktop machine by means of an HDMI capture card. The TV display allows us to visually inspect the crawler's behavior and we use the screenshot captures on the desktop machine to validate our findings visually and for further debugging. Finally, the TV display's audio output connects to the desktop machine's audio input, which we capture into files using `arecord`. Thus, the desktop machine—and thus the crawler—receives both audio and video signals emitted from the OTT device.

The crawler interacts with the OTT devices using their remote control APIs. Roku and Amazon Fire TV expose their remote control functionality via web APIs and `adb` respectively, both of which can receive keystroke commands to interact with the device. For example, "`adb shell input keyevent 21`" sends the "left" key to Amazon Fire TV devices, and an HTTP GET request to "`http://ROKU_DEVICE_IP_ADDRESS:8060/keydown/left`" does the same for all Roku devices.

The crawler uses a combination of such device commands to launch and install channels starting from the home screen of each OTT device. Operating one channel at a time from the list of channels, the crawl installs a channel, launches it, and interacts with it to play video. Soon after launching the channel, the crawler captures and attempts to decrypt various network and application level data. Finally, the crawler uninstalls the channel to free space on the OTT device.

The crawler's channel installation process is platform specific. On Roku, the crawler starts from the home page; visits the Roku Channel Store; opens the channel's page using its channel ID; and presses "Install" to install the channel. On Amazon, we discovered that delays in having channels appear on the device from the Amazon Fire TV channel store made the crawl prohibitively long and unrepeatable. Therefore, while compiling the list of channels to crawl, we installed each channel, waited for it to appear on the device, and then extracted the APK files from the device using `adb`. For successive crawls, we installed the channels from the APK files using `adb` as opposed to the channel store. The crawler then uses `adb` commands to install the channel APK files, which we retrieve from the Amazon Fire TV channel store ahead of running any crawls.

*3.2.2 Triggering Video Playback.* While merely launching a channel might reveal initial insight into its advertising and tracking ecosystem, triggering video playback and watching content, like an actual user, provides a more thorough and complete view. While our crawler can interact with channels using the OTT device's remote control APIs, triggering video playback in an automated fashion is challenging because channels' user interfaces vary widely. Therefore, we developed our crawler to maximize the probability of triggering video playback in channels.

To maximize the probability, we first analyzed how a human would interact with a channel to play a video. We randomly selected 100 channels from the Roku-Top1K list. For each channel, we manually recorded the shortest sequence of keystrokes that leads to video playback (e.g., the "Down" button followed by the "OK" button). The three sequences that triggered video playback on the most channels were: *[OK, OK, OK]*, *[Down, OK, OK]* and *[OK, OK, Down, OK, Down, OK]*.

The crawler employs these three key sequences by first opening a channel, executing one key sequence, and then checking the audio output signal (as a proxy for video playback). The crawler detects the audio signal by comparing the amplitude of the last five seconds of audio to the noise, and then checking if the difference between the two is greater than a certain threshold; we determined this threshold through iterative testing. If the crawler does not detect an audio signal, it assumes the key sequence did not work, restarts the channel, and then proceeds to play the remaining key sequences.

We tested the efficacy of our audio detection method by manually labelling screenshots from a sample of 150 channels with whether or not they achieved video playback. We compared this "ground truth" to the crawler's audio detection log to determine the accuracy, noting false positives and false negatives. Overall, the audio detection method was accurate in 144 of the 150 (96%) channels. We discovered only six cases of false positives (out of 40 detections) where the crawler erroneously concluded video playback had taken place. These were due to menu animations, audio guides and background music. We found no false negatives.

*3.2.3 Collecting Network Data and Intercepting Encrypted Communications.* The crawler collects the entire network level data as a PCAP file from the time of launching each channel to the time it is uninstalled. For each channel, the PCAP dump contains information about all DNS queries, HTTP requests, and TLS connections made during the crawl. The crawler also keeps track of all DNS queries in a Redis[2] key-value database for immediate retrieval and further analysis. The crawler attempts to decrypt TLS traffic with `mitmproxy` which we will describe next.

[2]https://redis.io

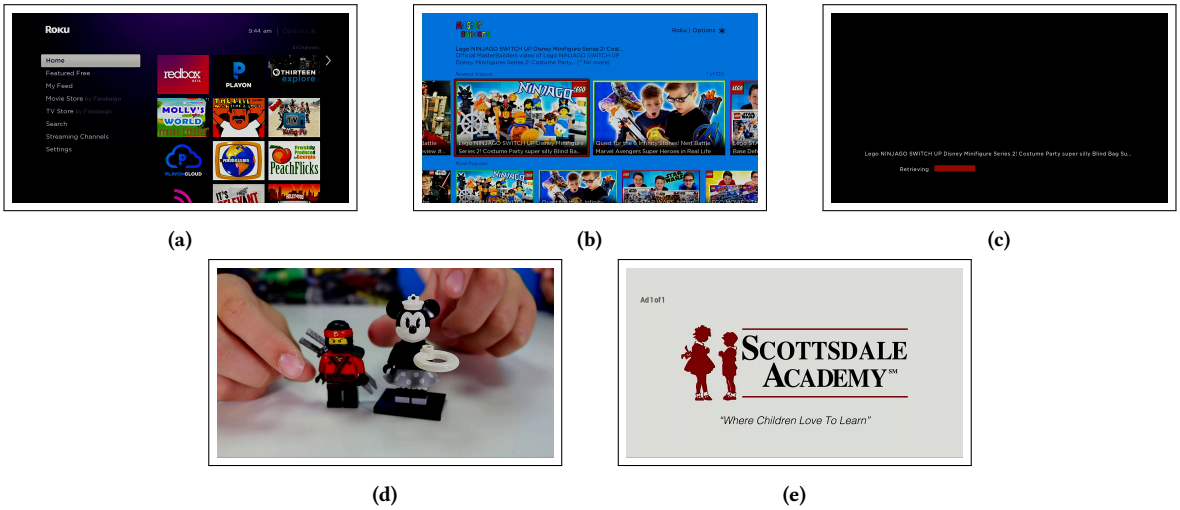



**Figure 3: A run of our smart crawler on a channel on Roku. The smart crawler launches the channel from the home screen (a); navigates to a video on the channel menu (b); waits for the video to start playing (c); plays and forwards the video (d); and encounters an advertisement (e).**

**Decrypting TLS Traffic with `mitmproxy`:** An open-source tool for intercepting and decrypting TLS traffic, `mitmproxy` [43], replaces the server's certificate with a different certificate transparently. If the channel does not properly validate the certificate, we can successfully intercept the TLS session and decrypt the captured traffic using the private key associated with the injected certificate. In practice, however, implementing this technique for OTTs involves multiple challenges. An OTT channel may establish TLS connections using a number of TLS implementations and settings, some of which may reject the certificate used for interception and the channel may fail to load at all. To overcome this, we wrote a TLS intercept companion script for `mitmproxy`. After a channel contacts a new TLS endpoint, the script learns whether or not the connection can be intercepted for that channel by examining the TLS alerts, and then adds the endpoints to the *"no-intercept list"* if TLS interception fails. However, TLS failures caused during this learning phase regularly interfere with the loading of a channel. In order to avoid such failures, we perform a warm-up stage in which we launch each channel multiple times beforehand to learn un-interceptable endpoints.

We also perform a number of optimizations to minimize the chance of TLS interception interfering with channel loading and performance. First, since the validation of the server's TLS certificate is performed on the client side, we assume all domains mapped to the same IP address behave similarly with respect to TLS interception. Therefore, our TLS intercept script uses the DNS capture daemon to obtain all other IP addresses mapped to the same domain and add them to the "no-intercept list". Second, we use the Public Suffix List [28] to extract base domains from hostnames, and we treat a base domain and its subdomains in the same way. Third, the script signals to the crawler when we learn no new un-interceptable endpoint during a warm-up launch, so the crawler can finish the warm-up phase earlier to save time. See Appendix A for more information on how we chose our warm-up parameters such as the number of warm-up crawls. Finally, for each channel, the crawler dumps all the TLS session keys for each intercepted session into a file, which can be used by Wireshark/tshark[3] to decrypt sessions. We then generate a list of all successfully intercepted endpoints.

**Certificate Injection and Pinning**: On Roku, our TLS interception rate is bounded by the number of channels with incorrect validation of certificates, since we cannot deploy our own certificate to the device. We compared the success rate of TLS interception using different X.509 certificates (see Appendix A for details) and used a self-signed certificate generated by `mitmproxy` with a common name matching the original certificate's common name.

On the Amazon Fire TV, however, we gained access to the system certificate store by rooting the device and installed our own certificate [70]. This allowed us to intercept more TLS endpoints beyond the ones that had a faulty certificate validation implementation. However, many channels and system services use certificate pinning [55]. We built a script which uses the Frida toolkit [29] to bypass channel level certificate pinning for all of the running channels, which we discuss in detail in Appendix B.

## 3.3 List of Crawls

Next, we summarize the list of crawls we conducted, and we describe the configurations for each crawl. As shown in Table 1, we conducted five different crawls on each platform.

*3.3.1 Smart Crawls.* In the following crawls, we used the smart crawler to interact with and trigger video playback on channels compiled using the method described in Section 3.1:

- **Top1K Crawls:** We crawled the top 1,000 channels—Roku-Top1K and FireTV-Top1K—on both the Roku and Amazon Fire TV channel stores with two configurations. In the first configuration, we intercepted encrypted traffic on both Roku and the Amazon Fire TV channels using the techniques

[3] https://www.wireshark.org/docs/man-pages/tshark.html

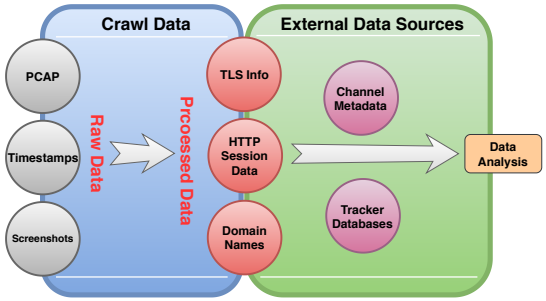



**Figure 4: Data processing pipeline**

we described in Section 3.2.3. We call these crawls **Roku-Top1K-MITM** and **FireTV-Top1K-MITM** respectively. As a point of comparison, we crawled the same channels without intercepting any encrypted traffic to analyze the success of our interception. We call these crawls **Roku-Top1K-NoMITM** and **FireTV-Top1K-NoMITM**.

- **Privacy Settings Crawls:** Next, we examined the efficacy of the privacy settings provided by Roku and the Amazon Fire TV. To do so, we first crawled the top 100 channels—Roku-CategoriesTop100 and FireTV-CategoriesTop100—picked across ten different categories from the Roku and Amazon Fire TV channel stores, intercepting encrypted traffic as before. We call these crawls **Roku-CategoriesTop100-MITM** and **FireTV-CategoriesTop100-MITM** respectively. We then repeated the same crawls, this time enabling the "Limit Ad Tracking" (Roku) and the "Disable Interest-based Ads" (Amazon Fire TV) settings. We call these crawls **Roku-Categories-Top100-LimitAdTracking** and **FireTV-CategoriesTop100-DisableInterestAds** respectively.

*3.3.2 Manual Crawls.* While our smart crawler is optimized to play videos on many kinds of channels, it fails to navigate channels that require human input (e.g., account registration and payment). To overcome this limitation, we manually interacted with the top 30 channels by rank from the Roku-Top1K and FireTV-Top1K channel lists. This included 1) signing up for accounts where possible, and 2) paying for premium and paid services using a credit card where necessary . We call these crawls **Roku-Top30-Manual-MITM** and **FireTV-Top30-Manual-MITM** respectively.

We crafted a protocol to ensure consistency in the manual crawls. When the channel required signing up on the web or visiting a website to enter a one time code, we launched an instance of OpenWPM tool [23] to perform the required authentication steps on the web and to collect the HTTP traffic for further analysis. We then attempted to view videos on the channel and fast forwarded to maximize observing ads. We spent no more than three minutes playing videos on each channel. The manual crawls collected the same network-level data as the smart crawls.

# 4 PROCESSING CRAWL DATA

In this section, we describe the raw data our crawler collected and the post-processing steps we performed after data collection.

## 4.1 Data Files

Our crawler generated a number of different files for each channel, as discussed below:

- PCAP Dumps: These contain network traffic for each channel crawled, filtered by the OTT device's IP address. These files constituted the basis of our analyses.
- SSL Key Log Files: These files contain the SSL keys used to intercept TLS sessions, all stored in NSS key log format [46]. These files can be used with `wireshark` or `tshark` to decrypt PCAP dumps[4].
- Events and Timestamps: For each channel it crawled, the crawler recorded a list of associated events in chronological order. These events included channel install, channel launch, channel uninstall, and any key presses (using the remote control API). We used these event timestamps to map events to network traffic patterns.
- TLS Artifacts: The crawler recorded a list of TLS endpoints that it successfully and unsuccessfully intercepted, along with the corresponding domain names. It used these mappings in subsequent crawls to minimize the warm-up period and to speed up crawling.
- Screenshots: The crawler captured screenshots from each channel it crawled and stored them in a folder. We used these screenshots to visually verify the crawler's interaction with each channel.
- Audio Recordings: The crawler recorded any audio that each channel played. It used these files to detect video playback.

## 4.2 Data Processing

After gathering the raw data for each crawl, we performed a processing step to extract relevant information. For instance, since we selectively attempted to intercept TLS connections, we needed a way to detect connections that we attempted to intercept. To do this we looked for certificates issued by `mitmproxy` using the following `tshark` filter: `x509sat.uTF8String==mitmproxy`. We labeled TLS connections as successfully intercepted if there was at least one TLS record containing payload (i.e. with `ssl.record.content_type` equal to 23) that was sent from the OTT device. We labeled the remaining TLS connections, which were attempted but had no payload originating from the OTT device, as interception failures.

We also used TCP connections throughout the paper for various measurements, since they contain both encrypted and unencrypted connections. For certain analyses, including measuring tracker prevalence, we needed a way to determine the hostname that corresponded to the destination IP address of a TCP connection. DNS queries collected during the crawls can be used for this purpose, but IP addresses may be shared by different domains (e.g. in a Content Delivery Network (CDN) setting), making it possible to have one IP address mapping to several domains. Indeed, we observed such collisions during our preliminary analysis. To reliably map IP addresses to hostnames, we followed a layered approach that made use of HTTP, TLS or DNS data, in this particular order. For a given TCP connection in a channel's network traffic: *(i)* we first checked whether there were any HTTP requests with a matching TCP stream index. If there were, we assigned the `Host` header of

[4] https://docs.mitmproxy.org/stable/howto-wireshark-tls/

| Device and Crawl Name | Channel Count | TLS Intercept? | Limit Ads Enabled? | Channels Completed | Video Playback (%) | Unique Domains |
|---|---|---|---|---|---|---|
| Roku-Top1K-NoMITM | 1,000 | No | No | 981 | 69 | 1,017 |
| Roku-Top1K-MITM | 1,000 | Yes | No | 982 | 62 | 1,043 |
| Roku-CategoriesTop100-MITM | 100 | Yes | No | 100 | 55 | 266 |
| Roku-CategoriesTop100-LimitAdTracking | 100 | Yes | Yes | 100 | 56 | 294 |
| Roku-Top30-Manual-MITM | 30 | Yes | No | 30 | 90 | 135 |
| FireTV-Top1K-NoMITM | 1,000 | No | No | 956 | 59 | 1,019 |
| FireTV-Top1K-MITM | 1,000 | Yes | No | 955 | 51 | 1,014 |
| FireTV-CategoriesTop100-MITM | 86 | Yes | No | 80 | 53 | 268 |
| FireTV-CategoriesTop100-DisableInterestAds | 86 | Yes | Yes | 80 | 56 | 262 |
| FireTV-Top30-Manual-MITM | 30 | Yes | No | 29 | 86 | 140 |

**Table 1: Overview of the crawls we conducted in this study using our smart crawler. In the crawls where "TLS intercept?" is "Yes", we either did warm-up launches during the crawl or loaded warm-up domain information from previous crawls.**

the request. If this failed, *(ii)* we searched for a TLS handshake sent over this TCP connection and used its TLS Server Name Indication (SNI) [6] field to determine the hostname (since SNI field is used to indicate which hostname a client is attempting to connect over TLS). If both of these failed, *(iii)* we used the DNS queries made during the crawl of this channel to determine the hostname corresponding to the server IP address.

In addition to HTTP 1.1 traffic, we extracted headers and payload from HTTP/2 frames present in the Fire TV crawls. We mapped HTTP/2 specific headers to their HTTP/1.1 counterparts (e.g. `authority` header to `host` header), and combined data from both HTTP versions in a unified format for ease of processing.

Finally, we used channel metadata, as discussed in Section 3.1, to retrieve channel ID, category, and rank to label the network traffic we observed. We also made use of existing tracking databases to classify domains as potential tracking domains, discussed further in Section 5.2.1. Figure 4 illustrates our data processing pipeline.

We used five popular ad-blocking/tracking protection lists to check the blocked status of HTTP requests and TCP connections: EasyList [16], EasyPrivacy [17], Disconnect [15], Ghostery [30] and Pi-hole [57]. The first four lists are widely used by millions of users, and they provide good coverage to block ads and prevent tracking on the web. Pi-hole, on the other hand, is a network-level ad blocker which can be used to block a wider set of trackers, including those present in IoT devices and Internet-connected TVs. We refer to hosts and domains detected by these five lists as *tracking domains*, acknowledging that the term may be too general for different types of practices such as analytics, audience measurement, and ad impression verification.

# 5 FINDINGS

This section covers our findings from smart crawls. We present an overview of the data we collected, followed by an analysis of tracking on channels and effectiveness of tracking countermeasures. Finally, we discuss some of the security properties and practices on the two OTT platforms we studied.

## 5.1 Data and Crawl Overview

As summarized in Table 1, we used a few different metrics to quantify the success of each crawl. First, we counted the number of "completed" channels in a crawl. We define "completed" channels as those that were successfully installed, interacted with, and then uninstalled by our crawler. Some channels may fail during install or launch for different reasons (e.g., network issues). Second, we counted the number of channels for which our smart crawler was able to reach video playback by using the remote control API in a crawl. Some channels may not have resulted in video playback because of complex interfaces (e.g., logging into an account, as explained in Section 3.3.2). Third, and finally, we counted the number of unique domains contacted by all channels in the crawl. This number indicates the range of endpoints the device contacted during the crawl. Together, these metrics provide a basis for sanity check of the different crawls—each with different parameters—on the same channel list.

We note two important observations about the overall results of our crawls (Table 1). First, on both Roku and the Amazon Fire TV, the crawls with and without TLS interception have similar success rates based on the metrics we define above. Thus, we can conclude that our TLS interception does not break channels' functionality. Second, our video playback rates are similar to our initial experiments and trials detailed in Section 3.2.2.

## 5.2 Tracker prevalence

We first present an overview of the trackers the Roku and Amazon Fire TV channels contact using data from the Roku-Top1K-MITM and FireTV-Top1K-MITM crawls. We refer to tracking domains, which we described in Section 4, as *"trackers"* hereafter for the sake of brevity.

*5.2.1 Trackers that are present on most channels.* Table 2 lists the top 10 trackers in order of prevalence across all the channels in the Roku-Top1K-MITM and FireTV-Top1K-MITM crawls. Across the Roku channels, `doubleclick.net`—owned by Google—was by far the most prevalent tracker, appearing on 975 of the 1,000 channels. Across the Amazon Fire TV channels, Amazon's own advertising tracking domain, `amazon-adsystem.com` was most prevalent, appearing in 687 of the 1,000 channels.

| Tracker Domain | Channel Count |
|---|---|
| doubleclick.net | 975 |
| google-analytics.com | 360 |
| scorecardresearch.com | 212 |
| spotxchange.com | 212 |
| googlesyndication.com | 178 |
| imrworldwide.com | 113 |
| tremorhub.com | 109 |
| innovid.com | 102 |
| 2mdn.net | 88 |
| vimeo.com | 86 |

| Tracker Domain | Channel Count |
|---|---|
| amazon-adsystem.com | 687 |
| crashlytics.com | 346 |
| doubleclick.net | 307 |
| google-analytics.com | 277 |
| facebook.com | 196 |
| d3a510xmpll7o6.cloudfront.net | 180 |
| app-measurement.com | 179 |
| googlesyndication.com | 145 |
| imasdk.googleapis.com | 129 |
| gstatic.com | 127 |

**Table 2: Most prevalent trackers from the Roku-Top1K-MITM (left) and the FireTV-Top1K-MITM (right) crawls.**

| Channel Name | Rank | Category | Tracker Count |
|---|---|---|---|
| StarGazer | 1012 | Special Interest | 50 |
| Rock Paper Scissors Free | 437 | Games | 42 |
| Falling Down Free | 421 | Games | 42 |
| Marble Blast Free | 738 | Games | 41 |
| Ping Pong Free | 447 | Games | 41 |
| Basketball Shots Free | 211 | Games | 41 |
| Swing Hero Free Game | 504 | Games | 40 |
| Pop Lock Free | 489 | Games | 40 |
| Pulse Free | 844 | Games | 37 |
| Soccer Shots Free | 509 | Games | 37 |

**Table 3: Top 10 channels from the Roku-Top1K-MITM based on the number of trackers they contact.**

Three Google domains associated with advertising and analytics (`doubleclick.net`, `google-analytics.com`, and `google-syndication.com`) are present in the top ten trackers list of both platforms. While `facebook.com` appeared in traffic from 196 Amazon Fire TV channels, it was only present in one Roku channel. This finding is in line with the results from tracking studies on web [23, 42, 63] and mobile [59] platforms that indicate that Facebook and Google advertising and tracking services are prevalent in web and mobile ecosystems as well as the OTT platforms, allowing them to create a more complete profile of users.

*5.2.2 Channels that contact the most trackers.* To further examine the distribution of trackers beyond prevalence, we examined their distribution across channel categories and channel ranks for both the Roku and Amazon Fire TV channels. The faceted plots in Figure 5a and Figure 5b illustrate this distribution for the Roku and Amazon Fire TV channels respectively. To complement the plots, Table 3 and Table 4 order channels by the number of trackers they contact. As expected, overall across the Roku and Amazon Fire TV channels we observe that channels that played video (orange dots) contain more trackers than those that failed to play video (blue dots).

Examining the distribution of trackers across channel categories, we notice several interesting patterns. For example, the "Games" category of Roku channels contact the most trackers, with 9 of the top 10 channels ordered by the number of trackers (Tables 3) being all from this category. On further inspecting these game channels, we discovered that these channels contained a similar number of trackers (37–42) and were published by the same developer (StuffWeLike). Also, five of the ten Fire TV channels with the most trackers are "News" channels (Table 4), where the top three channels contact close to 60 tracker domains each. This result implies that tracking behavior of channels differs depending on the category of the channel and that the users of certain categories are more susceptible to tracking.

## 5.3 Identifier and Information Leakage

Next, we investigate what pieces of information each domain that the channels in our dataset contact collect about users. This information reveals behaviors of existing trackers, as well as potentially new and previously unknown trackers.

| Channel Name | Rank | Category | Tracker Count |
|---|---|---|---|
| WNEP - Proud to Se... | 739 | News | 64 |
| ABC7 News San Fran... | 503 | News | 61 |
| WTTV CBS4 Indy | 1157 | News | 58 |
| Midnight Pulp | 933 | Movies & TV | 32 |
| Xtreme Vegas - Cla... | 341 | Games | 30 |
| Xtreme Slots - FRE... | 406 | Games | 28 |
| WVTM 13 -Birmingha... | 1022 | News | 25 |
| IP Tools: Network ... | 1029 | Utilities | 22 |
| ABC11 Raleigh-Durh... | 595 | News | 20 |
| FreeCell Solitaire | 894 | Games | 19 |

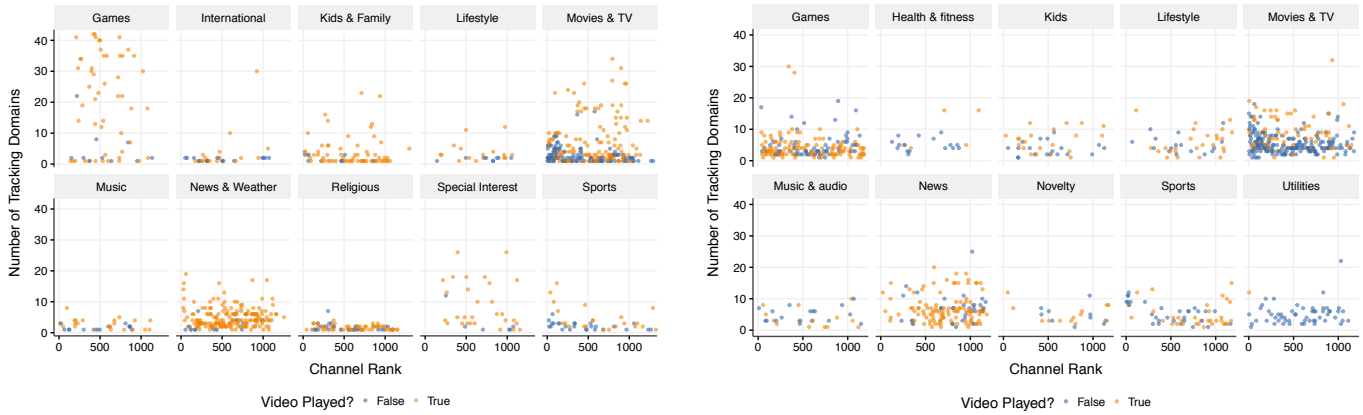
**Table 4: Top 10 channels from the FireTV-Top1K-MITM based on the number of trackers they contact.**

For this analysis, we compiled a list of possible identifiers and information that trackers could possibly collect about users. We based this list on pieces of device and user identifiers already available on the device interfaces (e.g., account information) and also on our preliminary analysis of the PCAP files. Table 5 lists these identifiers. We searched for these identifiers and pieces of information in the HTTP traffic resulting from the Roku-Top1K-MITM and FireTV-Top1K-MITM crawls.

Since these identifiers may be transmitted as encoded or hashed, we searched for various encoding and hashing combinations using the method described by Englehardt et al. [22]. This allowed us to detect URL and Base64 encoded identifiers, as well as hashed IDs. We performed the search on the path section of the URL, referrer and cookie headers, and post body of the HTTP requests. The detection results are shown in Table 6.

On Roku, we discovered that 4,452 of the 6,142 (nearly 73%) requests containing one of the two unique IDs (AD ID, Serial Number) are flagged as trackers. On Amazon 3,427 of the 8,433 (41%) unique identifiers are sent in cleartext.

In order to further understand the implication of subscription services, we manually crawled 30 channels on both Roku and Amazon platforms (as mentioned in Section 3.3.2). We collected network traffic of the OTT device and the website used to register and sign-in to the service. We used our tool to observe cross-device tracking and sharing of identifier with third-party trackers during these steps. We found four channels, two on each platform, that share the




**(a) Roku-Top1K-MITM** **(b) FireTV-Top1K-MITM**

**Figure 5: Distribution of trackers in the Roku-Top1K-MITM and FireTV-Top1K-MITM across channel ranks and channel categories.**

| Identifier Name | Persistence | Source |
|---|---|---|
| Serial No. | Lifetime of the device | Roku,Amazon |
| AD ID | Until reset by user | Roku,Amazon |
| MAC Addr | Lifetime of the device | Roku,Amazon |
| Device Name | Until reset by user | Roku,Amazon |
| Software Version | Lifetime of S/W update | Amazon |
| FireTV Home Version | Lifetime of S/W update | Amazon |
| Android ID | Lifetime of the device | Amazon |
| Bluetooth MAC Addr | Lifetime of the device | Amazon |
| Device ID | Lifetime of the device | Roku |
| Device Account Email | Until reset by user | User |
| Device Account Pwd | Until reset by user | User |
| Zip Code | Until device is moved | — |
| City | Until device is moved | — |
| State | Until device is moved | — |
| WiFi SSID | Until reset by user | Network |

**Table 5: The various device and user identifiers we checked for leaks in our study.**

| Identifier | Leak Count | Channel Count |
|---|---|---|
| AD ID | 2650 | 320 |
| Channel name | 2331 | 197 |
| Serial No | 996 | 110 |
| City | 64 | 11 |
| State | 33 | 6 |
| Zip | 61 | 10 |

| Identifier | Leak Count | Channel Count |
|---|---|---|
| Android ID | 3856 | 394 |
| MAC | 138 | 52 |
| Serial No | 377 | 105 |
| Device name | 64 | 40 |
| AD ID | 953 | 221 |
| Zip | 190 | 28 |
| City | 285 | 26 |
| Wifi SSID | 204 | 21 |
| Channel name | 5248 | 223 |
| State | 67 | 12 |

**Table 6: Overview of identifier and information leakage detected in the Roku-Top1K-MITM (left) and the FireTV-Top1K-MITM (right) crawls. We excluded leaks to platform domains `roku.com` and `amazon.com` from this table.**

email address of the profile used for account creation with third-party trackers on the web. We also found one Amazon channel sharing the zip code with a third-party on the web. Finally, for each channel, the tracking domains that are contacted both on the web (during registration or sign-in) and by the device, are potentially able to track users across web and OTT. We measured the presence of such domains in the 30 channels we manually crawled on each platform. On Roku, 4 channels had overlapping tracking domains on their device and web traffic, while on Amazon we found 7 such channels.

*5.3.1 Detecting New Tracking Domains.* Our analysis of trackers so far is based on existing adblocking lists. In this subsection, we use domains' behavior to detect previously unknown trackers. To this end, we used a combination of automated processing and manual false positive removal on the Roku-Top1K-MITM. We first built a list of domains that received unique identifiers, namely AD ID and device serial number. We removed known trackers from this list using the five adblocking lists that we used throughout the study. We then eliminated domains that were contacted only in one channel. Finally, we manually removed false positives using a WHOIS service, marketing materials, and other publicly available information. We provide a list of these domains in Appendix C.

*5.3.2 Title Leak Detection.* To determine whether trackers collect information about users' viewing habits, we checked whether channels shared video titles with the trackers. To this end, we randomly selected 100 channels on each platform from the subset of channels for which we detected video playback. The screenshots from each of these channels' menus were then manually reviewed to determine the title of the video. If a channel had a generic title such as "Live News" displayed in the menu, the screenshots of the video playback itself would be viewed to try to determine the most accurate title. Finally, we searched for different encodings of each video title in the network traffic, following a similar approach to the ID leak detection method explained in Section 5.3.

| Requests and Identifier leaks | Limit Ad Tracking? | |
|---|---|---|
| | No (default) | Yes |
| No. of HTTP requests | 4120 | 3880 |
| No. of contacted tracker domains | 96 | 128 |
| No. of contacted domains | 266 | 294 |
| AD ID (# instances/ # channels) | 390/30 | **0/0** |
| Serial No. (# instances/ # channels) | 135/14 | 118/13 |

**Table 7: Choosing to "Limit Ad Tracking" on the Roku Express (Roku-CategoriesTop100-LimitAdTracking crawl) seemingly makes no difference to the number and types of leaks to trackers except the AD ID is not leaked at all.**

We found 9 channels on Roku and 14 channels on the Fire TV, among the 100 channels we randomly selected on each device, that leaked the title of the video to a tracking domain. The majority of these channels were news channels with 8 channels on Roku and 7 channels on Fire TV being news related. We list the channel name, video title, and tracking domain the title was leaked to in the Appendix G (Table 13 for Roku and Table 14 for Fire TV). On Roku, all video titles are leaked over unencrypted connections, exposing users' viewership preferences to eavesdroppers. On Fire TV, only two channels (NBC News and WRAL) used an unencrypted connection when sending the title to tracking domains.

The study by Malkin et al. [41] shows only 29% of Smart TV users surveyed believed it was acceptable for the advertiser to collect their viewing behavior, and our findings show the gap between user expectation and current tracking practices. Furthermore, as we will discuss in Section 6, this type of tracking may be subject to certain legislation such as the Video Privacy Protection Act ("VPPA") in the United States which prohibits collecting the viewing history of clients for video rental services.

## 5.4 Effect of countermeasures

Both Roku and Amazon Fire TV provide privacy options to users that purport to limit tracking on their devices. On Roku, this option is called "*Limit Ad Tracking*" and on Amazon Fire TV it is called "*Disable Interest based Ads*". Both options are off by default.

**"Limit Ad Tracking" on Roku:** To find out the effect of this option, we ran a crawl with "Limit Ad Tracking" on and compared the results to a crawl with the same parameters, except with "Limit Ad Tracking" turned off. For comparison, we measured the AD ID and serial number leaks; number of tracker domains contacted; and the number of HTTP requests. Turning on "Limit ad tracking" reduced the number of AD ID leaks from 390 to zero, but did not affect the number of trackers contacted by the channels (Table 7). Moreover, the number of serial number leaks stayed the same. The increase in the number of contacted domains (96 to 128) also points to the limited efficacy of the "Limit Ad Tracking" option.

**"Disable Interest based ads" on Amazon:** Next, we evaluated the effect of the "Disable Interest based ads" option that is available on the Fire TV. Comparing the domains contacted in FireTV-CategoriesTop100-DisableInterestAds and FireTV-CategoriesTop100-MITM crawls, we found that the number of requests to `amazon--adsystem.com` decreased from 65 to just 10 channels, meaning Amazon seemingly reduced access to its own advertising system.

| Requests and Identifier leaks | Disable Interest Based Ads? | |
|---|---|---|
| | No (default) | Yes |
| No. of HTTP requests | 8895 | 8269 |
| No. of contacted tracker domains | 119 | 115 |
| No. of domains | 268 | 262 |
| AD ID (# instances/ # channels) | 198/35 | 144/16 |
| Android ID (# instances/ # channels) | 607/69 | 638/65 |
| Serial No. (# instances/ # channels) | 151/33 | 127/24 |
| MAC Addr. (# instances/ # channels) | 22/3 | 26/3 |

**Table 8: Choosing to "Disable Interest Ads" on the Amazon Fire TV Stick (FireTV-CategoriesTop100-DisableInterestAds crawl) seemingly makes no difference to the number and types of leaks to trackers except nearly 50% fewer channels leak the AD ID when the option is enabled.**

We also observed that the trackers collecting the AD ID reduced by nearly half. However, as Table 8 shows, the remaining traffic, including the other identifiers communicated to trackers, remained largely the same.

Both Roku and Amazon state in their developer documentation that if the privacy option is enabled, the AD ID "should not be used for targeted advertising" [64] or to "collect information about the user's behavior to build user profiles for advertising purposes". Nevertheless, developers can use the AD ID for activities such as frequency capping, contextual advertising, conversion tracking, reporting, and security and fraud detection [4, 66]. Our data, however, reveals that even when the privacy option is enabled, there are a number of other identifiers that can be used to track users, bypassing the privacy protections built into these platforms[5].

**Pi-hole's network-level blocking.** Pi-hole uses a modified version of `dnsmasq` to block DNS queries containing unwanted hosts. Pi-hole blocks DNS queries based on hostnames, although a wildcard blocking feature was recently added[6]. Pi-hole comes with a default list of blocked hostnames, but does not come with wildcard patterns at the install time.

Reading through the Pi-hole's source code, we compiled a list of hostnames that it blocks by default. Pi-hole's installation wizard offers seven different lists that are turned on by default[7]. In our simulations we assumed the user would select all lists offered to them, the most conservative approach. We then simulated Pi-hole blocking on the data we collected.

Pi-hole blocked 4,226 of the 18,075 HTTP requests made in the Roku-Top1K-MITM crawl. This corresponds to 71% of the 5,983 requests blocked by four other blocking lists (EasyList, EasyPrivacy, Disconnect and Ghostery). Inspecting the hostnames of these missed requests, we found that Pi-hole's blocklist actually contained hostnames from 38 of the 70 domains that were missed. The missed requests were commonly due to a different subdomain of the same tracker.

Simulating Pi-hole's blocking of the information leaks we find that 26.7% of the AD ID leaks and 44.6% of the serial number leaks

[5] In fact, to comply with the European Union's General Data Protection Regulation ("GDPR"), Roku sets the Limit Ad Tracking value to "true" on all devices in the EU [68], which seems inadequate given our result.
[6] https://docs.pi-hole.net/ftldns/regex/overview/
[7] https://git.io/fjNsi

| Domain | Channel Count | Domain | Channel Count |
|---|---|---|---|
| doubleclick.net | 266 | amazon-adsystem.com | 678 |
| google-analytics.com | 175 | scorecardresearch.com | 108 |
| scorecardresearch.com | 145 | ifood.tv | 50 |
| roku.com | 145 | images-amazon.com | 45 |
| ifood.tv | 90 | cloudinary.com | 32 |
| tremorhub.com | 79 | titantv.com | 29 |
| stickyadstv.com | 74 | wsi.com | 27 |
| irchan.com | 74 | cdn01.net | 24 |
| monarchads.com | 73 | lightcast.com | 24 |
| 1rx.io | 66 | demdex.net | 22 |

**Table 9: Most prevalent domains contacted over unencrypted connections in the Roku-Top1K-NoMITM (left) and the FireTV-Top1K-NoMITM (right) crawls.**

| Channel Name | Rank | Hostname |
|---|---|---|
| DIRECTV NOW | 7 | api.cld.dtvce.com |
| VUDU | 15 | apicache.vudu.com |
| VUDU | 15 | vudu.d1.sc.omtrdc.net |
| NBC | 19 | ws-cloudpath.media.nbcuni.com |
| Fox News Channel | 35 | api.segment.io |
| fuboTV Watch Live Sports & TV | 43 | api.fubo.tv |
| fuboTV Watch Live Sports & TV | 43 | api.segment.io |
| fuboTV Watch Live Sports & TV | 43 | app.launchdarkly.com |
| Newsy | 45 | cloudapi.imrworldwide.com |
| MTV | 54 | auth.mtvnservices.com |

**Table 10: Top channels where TLS Connections were intercepted and decrypted by our smart mitmproxy (Roku-Top1K-MITM crawl).**

are missed by Pi-hole. Finally, we list the domains that received these IDs and were missed by Pi-hole in Table12 in Appendix E.

## 5.5 Network Connection Security

*5.5.1 Unencrypted Connections.* Analyzing the requests sent over port 80 we found that 794 of the 1000 Roku channels sent at least one request in cleartext. Those channels contacted 191 distinct hosts from 123 distinct domains without encryption.

Similarly, 762 of the 1000 Fire TV channels sent at least one unencrypted request. The top 10 domains contacted over insecure connections by Roku and Amazon are shown in Table 9. Notably, Google and Amazon own the domains that are contacted by most channels without encryption.

*5.5.2 Certificate validation.* On Amazon Fire TV, we were able to install our own cert on the device which allowed us to intercept HTTPS requests on 957 of the 1000 channels in the FireTV-Top1K-MITM crawl. On Roku, a total of 43 channels failed to properly verify the server's certificate and allowed the smart proxy to decrypt the TLS traffic. Table 10 shows the top ten TLS connections (by channel rank) that our smart proxy was able to decrypt in the Roku-Top1K-MITM crawl. Finally, we performed an analysis of the TLS cipher suites used by different channels which can be found in Appendix D.

## 5.6 Remote Control API Vulnerabilities

To investigate other ways OTT devices may compromise user privacy and security, we analyzed local API endpoints of Roku and Fire TV. OTT devices expose such interfaces to enable debugging, remote control, and home automation by mobile apps and other automation software [65, 78]. In the past, security researchers identified several vulnerabilities of these interfaces, including Cross-Site Request Forgery [75] and DNS Rebinding [1, 14]. Recently, Roku's External Control API, which we used in this study to automate our crawls, was found to be vulnerable to a DNS Rebinding attack [50, 65]. Moreover, similar attacks were found to be exploited in the wild by exploit kits for malicious advertising purposes [34].

We limited our analysis of local interfaces to attacks by malicious web scripts considering the large attack surface and limited resources needed by an attacker. To perform this attack, the attacker may run a malicious script (e.g. by running ads) on websites visited by a user browsing from the same network that the OTT device is connected to (e.g. a home network).

To discover available API endpoints, we installed mobile remote control apps and reverse engineered the HTTP traffic while sending available remote control commands to OTT devices. We also made use of the API documentation available for the devices to discover endpoints that were missing in the captured traffic [65, 78].

**Roku**: Roku has an extensive external control API that enables automation by third-party software [65]. The API allows sending commands to install/uninstall/change channels and retrieve device information. The device information returned by the API includes the complete list of installed channels, unique device identifiers (e.g. MAC address, Ad ID, serial no), and SSID of the wireless network that the device is connected to.

Analyzing the headers sent by Roku to remote control requests, we discovered that Roku sends "*" in the "Access-Control-Allow-Origin:" header. The "*" value relaxes cross-origin restrictions in the browser and allows scripts from an arbitrary domains to read cross-origin resources without being limited by the Same Origin Policy [47]. This exposes Roku devices to attacks from all web pages visited by Roku users.

We set up a page to demonstrate the attack and verified that a malicious web page visited by Roku users (or third-party scripts embedded on them) can abuse the External Control API to:

- Send commands to install/uninstall/launch channels and collect unique identifiers from Roku devices - even when the connected display is turned off.
- Geolocate Roku users via the SSID of the wireless network and WiFi SSID & Geolocation databases (such as WiGLE [80]).
- Extract MAC address, serial number, and other unique identifiers to track users or respawn tracking identifiers (similar to evercookies [35]).
- Get the list of installed channels and use it for profiling purposes. For instance, the existence of health/religion/children focused channels can be used to infer victims' sensitive personal attributes and their lifestyles.

We reported the vulnerability to Roku in December 2018. Roku addressed the issue by changing the value in the "Access-Control-Allow-Origin" header. They finalized rolling out their security fix by March 2019.

**Fire TV**: Fire TV provides a custom remote control API over the network which allows a user to use their phone as the remote control, which we analyzed in Appendix F.

# 6 DISCUSSION

Past research on user privacy expectations about smart TVs finds that viewers find sharing of their data with advertisers unacceptable [41]. Our findings show that such concerned users have limited options at their disposal. As emerging platforms, OTT services lack tools, controls, and countermeasures available on the web and mobile platforms. Users who enjoy installing adblockers for their web browsers with just a few clicks often lack usable defenses to protect their video privacy on OTT platforms. Moreover, widespread collection of persistent device identifiers like MAC addresses and serial numbers disables one of the few defenses available to users: resetting their advertising IDs. Trackers who collect persistent IDs can link users' activity even if they reset their AD IDs.

In addition, the tendency of companies to monetize users' data makes it difficult to incentivize privacy friendly practices. It is often the role of regulators to keep platform and application developers accountable. Our tool is an apt solution that can be used by regulators to inspect channels and devices to enforce privacy regulations on OTT platforms.

## 6.1 Recommendations

Based on our findings, we believe OTT vendors should borrow ideas from other platforms, such as web, to provide a more privacy friendly and secure experience to their users. Specifically, we make the following recommendations based on our findings:

- OTT platforms should offer better privacy controls, similar to Incognito/Private Browsing Mode of modern web browsers. Platforms should ensure that linking of private and non-private profiles of the same user is not possible by partitioning identifiers and their storage along with denying access to existing platform identifiers.
- Platforms should make it possible to monitor the content of the network traffic, similar to browsers and operating systems allowing intercepting one's own traffic by installing a self-signed certificate in the browser. This would allow tech-savvy users and security researchers to analyze channel behavior, leading to a more transparent ecosystem. These features should require explicit user consent and display their status prominently to mitigate the risk of use by one user to spy on another user.
- To complement limited technical protections available to users, regulators and policy makers should ensure the privacy protections[8] available for brick and mortar video rental services are updated to cover emerging OTT platforms, where—as our research shows—users are constantly and pervasively tracked.
- OTT platforms should introduce policies to disincentivize insecure connections, for example, by blocking clear-text connections unless an exception is requested by the channel (similar to Apple's App Transport Security feature[33]).

[8] e.g., the US Video Privacy Protection Act ("VPPA")

## 6.2 Limitations and Future Work

Below we discuss the limitations of our work and potential future work to address these limitations.

*6.2.1 Channel Login.* Our crawler cannot go beyond login or signup pages. This limitation, which is common among similar automation tools, prevents us from reaching video playback on some channels. As future work, we plan to integrate a real-time signal from an image recognition script to detect, fill, and submit login forms during crawls in order to increase the video playback rate. During the manual crawl, we noticed some of the channels require sign up through a web browser.

*6.2.2 Background Traffic.* Our traffic captures may contain background traffic from Roku and Amazon platforms. Based on our preliminary analysis of the background traffic when no channel is active, the effect on the results should be minimal. Distinguishing background traffic for closed platforms like Roku is a problem we plan to address in future studies.

*6.2.3 Smart TV Platforms.* We studied two of the most popular OTT streaming devices, leaving out the smart TV platforms such as Samsung, Vizio and Apple TV. We believe our automated crawler can be easily modified to run experiments on these platforms, as long as they have similar remote control APIs.

# 7 CONCLUSION

In this paper, we presented the first large scale study of tracking by OTT streaming channels. Our measurement of the more than 2,000 OTT streaming channels revealed widespread user tracking and data collection. To perform the study, we built a smart crawler that automatically installs, launches, and interacts with the OTT channels.

Our measurements showed that tracking is prevalent on the OTT platforms we studied, with traffic to known trackers present on 69% of Roku channels and 89% of Amazon Fire TV channels. We also observed that certain OTT channels contact more than 60 tracking domains and the data shared with the trackers include video titles, WiFi SSIDs, MAC addresses and device serial numbers. Analyzing the network data collected by our crawler, we discovered that 79% of the Roku channels and 76% of the Fire TV channels send at least one unencrypted HTTP request. Finally, we identified new tracking domains not seen in previous tracking studies.

Our analysis of the available privacy countermeasures showed that they are ineffective at preventing tracking. Such weak countermeasures should be supported by policies and regulations to ensure that users' viewership information remains private.

# 8 ACKNOWLEDGEMENTS

This paper was supported in part by NSF awards CPS-1739809, CNS-1553437 and CNS-1704105. We thank Kevin Borgolte and the anonymous CCS reviewers for their comments.

## REFERENCES


[1] Gunes Acar, Danny Yuxing Huang, Frank Li, Arvind Narayanan, and Nick Feamster. 2018. Web-based Attacks to Discover and Control Local IoT Devices. In *Proceedings of the 2018 Workshop on IoT Security and Privacy*. ACM, 29–35.

[2] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. 2013. FPDetective: dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC conference on Computer and Communications Security (CCS)*. 1129–1140.

[3] Amazon. 2019. Fire TV Apps. Retrieved August 25, 2019 from https://www.amazon.com/Fire-TV-Apps-All-Models/b?ie=UTF8&node=10208590011

[4] Amazon. 2019. Fire TV Documentation. Retrieved August 25, 2019 from https://developer.amazon.com/docs/fire-tv/advertising-id.html

[5] Amazon. 2019. Fire TV Stick. Retrieved August 25, 2019 from https://www.amazon.com/dp/B0791TX5P5?ref=ODS_v2_FS_SMP_TVStick

[6] Simon Blake-Wilson, Magnus Nystrom, David Hopwood, Jan Mikkelsen, and Tim Wright. 2006. *Transport layer security (TLS) extensions*. Technical Report. Retrieved August 25, 2019 from https://tools.ietf.org/html/rfc3546#section-3.1

[7] Piergiovanni Cipolloni. 2017. Universal Android SSL Pinning bypass with Frida. https://techblog.mediaservice.net/2017/07/universal-android-ssl-pinning-bypass-with-frida/

[8] Comscore. 2016. Roku Leads OTT Streaming Devices in Household Market Share. Retrieved August 25, 2019 from https://www.comscore.com/ita/Public-Relations/Blog/Roku-Leads-OTT-Streaming-Devices-in-Household-Market-Share

[9] Dave Cooper. 2008. Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile. (2008). https://tools.ietf.org/html/rfc5280

[10] Anupam Das, Gunes Acar, Nikita Borisov, and Amogh Pradeep. 2018. The Web's Sixth Sense: A Study of Scripts Accessing Smartphone Sensors. In *Proceedings of the 25th ACM Conference on Computer and Communication Security (CCS)*. ACM. https://doi.org/10.1145/3243734.3243860

[11] Anupam Das, Nikita Borisov, and Matthew Caesar. 2014. Do You Hear What I Hear?: Fingerprinting Smart Devices Through Embedded Acoustic Components. In *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 441–452.

[12] Anupam Das, Nikita Borisov, and Matthew Caesar. 2016. Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses. In *Proceeding of 23rd Annual Network and Distributed System Security Symposium (NDSS)*.

[13] Anupam Das, Nikita Borisov, and Edward Chou. 2018. Every Move You Make: Exploring Practical Issues in Smartphone Motion Sensor Fingerprinting and Countermeasures. *Proceedings on Privacy Enhancing Technologies (PoPETs)* 1 (2018), 88–108.

[14] Drew Dean, Edward W Felten, and Dan S Wallach. 1996. Java Security: From HotJava to Netscape and Beyond. In *Proceedings 1996 IEEE Symposium on Security and Privacy*. IEEE, 190–200.

[15] Disconnect. 2018. Disconnect defends the digital you. https://disconnect.me/.

[16] EasyList. 2019. Overview. https://easylist.to/.

[17] EasyPrivacy. 2019. EasyPrivacy. https://easylist.to/tag/easyprivacy.html.

[18] Peter Eckersley. 2010. How Unique is Your Web Browser?. In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies (PETS)*. 1–18.

[19] eMarketer. 2018. US Connected TV Users, by Brand, 2018 & 2022. Retrieved August 25, 2019 from https://www.emarketer.com/Chart/US-Connected-TV-Users-by-Brand-2018-2022-of-connected-TV-users/220767

[20] Jasmine Enberg. 2018. Roku is winning the connected TV race, and Amazon probably won't catch up anytime soon. Retrieved August 25, 2019 from https://www.businessinsider.com/roku-is-winning-the-connected-tv-race-ahead-of-amazon-2018-7

[21] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. 2014. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. *ACM Transactions on Computer Systems (TOCS)* 32, 2 (2014), 5.

[22] Steven Englehardt, Jeffrey Han, and Arvind Narayanan. 2018. I never signed up for this! Privacy implications of email tracking. *Proceedings on Privacy Enhancing Technologies* 2018, 1 (2018), 109–126.

[23] Steven Englehardt and Arvind Narayanan. 2016. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of ACM CCS 2016*.

[24] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W Felten. 2015. Cookies That Give You Away: The Surveillance Implications of Web Tracking. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 289–299.

[25] Susan Engleson. 2018. State of OTT: An in-depth look at today's over-the-top content consumption and device usage. Retrieved August 25, 2019 from https://www.aaaa.org/wp-content/uploads/2017/07/ComScore-State-of-OTT.pdf

[26] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security analysis of emerging smart home applications. In *2016 IEEE symposium on security and privacy (SP)*. IEEE, 636–654.

[27] David Fifield and Serge Egelman. 2015. Fingerprinting Web Users Through Font Metrics. In *International Conference on Financial Cryptography and Data Security*. Springer, 107–124.

[28] Mozilla Foundation. 2019. Public Suffix List. Retrieved August 25, 2019 from https://publicsuffix.org/

[29] Frida. 2019. A world-class dynamic instrumentation framework. https://www.frida.re/

[30] Ghostery. 2019. Ghostery Makes the Web Cleaner, Faster and Safer! Retrieved August 25, 2019 from https://www.ghostery.com/

[31] Gábor György Gulyás, Gergely Acs, and Claude Castelluccia. 2016. Near-optimal fingerprinting with constraints. *Proceedings on Privacy Enhancing Technologies* 2016, 4 (2016), 470–487.

[32] Peter Hornyack, Seungyeop Han, Jaeyeon Jung, Stuart Schechter, and David Wetherall. 2011. "These Aren't the Droids You're Looking For": Retrofitting Android to Protect Data from Imperious Applications. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 639–652.

[33] Apple Inc. 2019. Preventing Insecure Network Connections | Apple Developer Documentation. https://developer.apple.com/documentation/security/preventing_insecure_network_connections.

[34] KAFEINE. 2016. Home Routers Under Attack via Malvertising on Windows, Android Devices. https://www.proofpoint.com/us/threat-insight/post/home-routers-under-attack-malvertising-windows-android-devices.

[35] Samy Kamkar. 2019. evercookie - virtually irrevocable persistent cookies. https://samy.pl/evercookie/.

[36] Balachander Krishnamurthy and Craig Wills. 2009. Privacy diffusion on the web: a longitudinal perspective. In *Proceedings of the 18th international conference on World wide web*. ACM, 541–550.

[37] Andreas Kurtz, Hugo Gascon, Tobias Becker, Konrad Rieck, and Felix Freiling. 2017. Fingerprinting Mobile Devices Using Personalized Configurations. *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2016, 1 (2017), 4–19.

[38] Adam Levy. 2018. Roku's Advertising Revenue Will Surpass Its Player Revenue This Year. Retrieved August 25, 2019 from https://www.fool.com/investing/2018/03/26/rokus-advertising-revenue-will-surpass-its-player.aspx

[39] Timothy Libert. 2015. Exposing the Hidden Web: An Analysis of Third-Party HTTP Requests on One Million Websites. *International Journal of Communication* (2015).

[40] Franco Loi, Arunan Sivanathan, Hassan Habibi Gharakheili, Adam Radford, and Vijay Sivaraman. 2017. Systematically evaluating security and privacy for consumer IoT devices. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. ACM, 1–6.

[41] Nathan Malkin, Julia Bernd, Maritza Johnson, and Serge Egelman. 2018. "What Can't Data Be Used For?" Privacy Expectations about Smart TVs in the US. In *European Workshop on Usable Security (Euro USEC)*.

[42] Jonathan R Mayer and John C Mitchell. 2012. Third-party web tracking: Policy and technology. In *Proceedings of the 33rd IEEE Symposium on Security and Privacy (SP)*. 413–427.

[43] mitmproxy. 2019. How mitmproxy works. https://docs.mitmproxy.org/stable/concepts-howmitmproxyworks/.

[44] Keaton Mowery, Dillon Bogenreif, Scott Yilek, and Hovav Shacham. 2011. Fingerprinting Information in JavaScript Implementations. In *Proceedings of W2SP*, Vol. 2.

[45] Keaton Mowery and Hovav Shacham. 2012. Pixel perfect: Fingerprinting canvas in HTML5. In *Proceedings of Web 2.0 Security and Privacy Workshop (W2SP)*.

[46] MDN Web Docs Mozilla. 2019. NSS Key Log Format. Retrieved August 25, 2019 from https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/Key_Log_Format

[47] MDN Web Docs Mozilla. 2019. Same-origin policy. Retrieved August 25, 2019 from https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

[48] Sashank Narain, Triet D Vo-Huu, Kenneth Block, and Guevara Noubir. 2016. Inferring User Routes and Locations using Zero-Permission Mobile Sensors. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 397–413.

[49] Inc. Netflix. 2018. DIAL. http://www.dial-multiscreen.org/.

[50] Lily Hay Newman. 2019. Millions of Google, Roku, and Sonos Devices Are Vulnerable to a Web Attack | WIRED. https://www.wired.com/story/chromecast-roku-sonos-dns-rebinding-vulnerability/.

[51] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 2012. You are what you include: large-scale evaluation of remote javascript inclusions. In *Proceedings of the 19th ACM SIGSAC conference on Computer and Communications Security (CCS)*. 736–747.

[52] Evan Niu. 2018. Roku Is Beefing Up Ad Targeting in a Big Way. Retrieved August 25, 2019 from https://www.fool.com/investing/2018/06/27/roku-is-beefing-up-ad-targeting-in-a-big-way.aspx

[53] Lukasz Olejnik. 2017. Stealing sensitive browser data with the W3C Ambient Light Sensor API. https://blog.lukaszolejnik.com/stealing-sensitive-browser-data-with-the-w3c-ambient-light-sensor-api/.

[54] Lukasz Olejnik, Gunes Acar, Claude Castelluccia, and Claudia Diaz. 2015. The leaking battery A privacy analysis of the HTML5 Battery Status API. In *International Workshop on Data Privacy Management*. 254–263.
[55] Open Web Application Security Project (OWASP). 2019. Certificate and Public Key Pinning. Retrieved August 25, 2019 from https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning
[56] Nilay Pate. 2019. Taking the smarts out of smart TVs would make them more expensive. Retrieved August 25, 2019 from https://www.theverge.com/2019/1/7/18172397/airplay-2-homekit-vizio-tv-bill-baxter-interview-vergecast-ces-2019
[57] Pi-hole®. 2019. Pi-hole®: A black hole for Internet advertisements. https://pi-hole.net/.
[58] Andrey Popov. 2015. *RFC 7465: Prohibiting RC4 cipher suites*. Technical Report. https://tools.ietf.org/html/rfc7465
[59] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, and Christian Kreibich Phillipa Gill. 2018. Apps, Trackers, Privacy, and Regulators. (2018).
[60] FTC Press Releases. 2017. VIZIO to Pay $2.2 Million to FTC, State of New Jersey to Settle Charges It Collected Viewing Histories on 11 Million Smart Televisions without Users' Consent.
[61] Jingjing Ren, Ashwin Rao, Martina Lindorfer, Arnaud Legout, and David Choffnes. 2016. ReCon: Revealing and Controlling PII Leaks in Mobile Network Traffic. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 361–374.
[62] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, and Serge Egelman. 2018. "Won't Somebody Think of the Children?" Examining COPPA Compliance at Scale. *Proceedings on Privacy Enhancing Technologies* 2018, 3 (2018), 63–83.
[63] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. 2012. Detecting and Defending Against Third-Party Tracking on the Web. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 12–12.
[64] Roku. 2019. Developers Documentation. Retrieved August 25, 2019 from https://developer.roku.com/docs/references/brightscript/interfaces/ifdeviceinfo.md#getrida-as-dynamic
[65] Roku. 2019. External Control Protocol (ECP). Retrieved August 25, 2019 from https://developer.roku.com/docs/developer-program/discovery/external-control-api.md
[66] Roku. 2019. Integrating the Roku Advertising Framework. Retrieved August 25, 2019 from https://developer.roku.com/docs/developer-program/advertising/integrating-roku-advertising-framework.md
[67] Roku. 2019. Roku Channel Store. Retrieved August 25, 2019 from https://channelstore.roku.com/browse
[68] Roku. 2019. Roku Compliance. Retrieved August 25, 2019 from https://developer.roku.com/develop/platform-features/compliance
[69] Roku. 2019. Roku Express. Retrieved August 25, 2019 from https://www.roku.com/products/roku-express
[70] Elias Saba. 2019. Amazon Fire TV Stick 2 has been Rooted for the first time. Retrieved August 25, 2019 from http://www.aftvnews.com/amazon-fire-tv-stick-2-has-been-rooted-for-the-first-time/
[71] Selenium. 2019. Selenium - Web Browser Automation. Retrieved August 25, 2019 from https://www.seleniumhq.org/
[72] Sensepost. 2019. sensepost/objection. https://github.com/sensepost/objection
[73] Mike Shields. 2018. Inside Roku's battle to control the future of TV advertising — and why it better watch out for Amazon. Retrieved August 25, 2019 from https://www.businessinsider.com/roku-wants-to-control-the-future-of-tv-ads
[74] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. 2010. Flash cookies and privacy. In *2010 AAAI Spring Symposium Series*.
[75] Sid Stamm, Zulfikar Ramzan, and Markus Jakobsson. 2007. Drive-by pharming. In *International Conference on Information and Communications Security*. Springer, 495–506.
[76] Oleksii Starov and Nick Nikiforakis. 2017. Extended Tracking Powers: Measuring the Privacy Diffusion Enabled by Browser Extensions. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1481–1490.
[77] Brett Tingley. 2018. Over One Third of US Households Will Cut the Cord by 2020. Retrieved August 25, 2019 from https://www.streamingobserver.com/over-one-third-of-us-households-will-cut-the-cord-by-2020/
[78] Amazon Fire TV. 2019. Remote Control Input. Retrieved August 25, 2019 from https://developer.amazon.com/docs/fire-tv/remote-input.html
[79] Daniel Veditz. 2011. *Rizzo/Duong chosen plaintext attack (BEAST) on SSL/TLS 1.0 (facilitated by websockets -76)*. Technical Report. https://bugzilla.mozilla.org/show_bug.cgi?id=665814
[80] WiGLE.net. 2019. WiGLE: Wireless Network Mapping. Retrieved August 25, 2019 from https://wigle.net/
[81] Daniel Wood, Noah Apthorpe, and Nick Feamster. 2017. Cleartext data transmissions in consumer iot medical devices. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. ACM, 7–12.
[82] Ning Xia, Han Hee Song, Yong Liao, Marios Iliofotou, Antonio Nucci, Zhi-Li Zhang, and Aleksandar Kuzmanovic. 2013. Mosaic: Quantifying Privacy Leakage in Mobile Networks. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*. ACM, New York, NY, USA, 279–290. https://doi.org/10.1145/2486001.2486008

# A TLS INTERCEPTION SETTINGS

## A.1 TLS Interception Certificate

When we cannot deploy our own certificate to the device, our TLS interception rate is bounded by the number of channels with incorrect validation of certificates. We compared the success rate of TLS interception using two different X.509 certificates on Roku. First, a self-signed certificate generated by `mitmproxy` with a common name matching the original certificate's common name was used. If a channel does not validate the chain of certificate authorities (CAs), the client will complete the TLS handshake using this certificate and proceed with the communication. Second, a certificate issued by Let's Encrypt[9] for a domain we own (**"3016sale.xyz"**) was tried. In this case, the channel proceeds with the connection if it does not validate the fields "Subject" or "Subject Alternative Name" in the certificate [9]. Using the self-signed certificate, we attempted intercepting sessions on top 100 channels on Roku. We were able to intercept 12 distinct hosts from 9 channels using this certificate; whereas, interception was successful only on 5 distinct hosts from 4 channels with the Let's Encrypt certificate. Thus, we use the self-signed certificate for larger crawls.

## A.2 TLS Interception Learning Rate

To measure how fast our TLS interception learns new endpoints that cannot be intercepted and adds them to the no-intercept list, we measure the number of endpoints where we attempted to intercept the TLS connection but failed for each warm-up launch of a channel. The resulting histogram is shown in Figure 6 and Figure 7 shows the corresponding CDF.

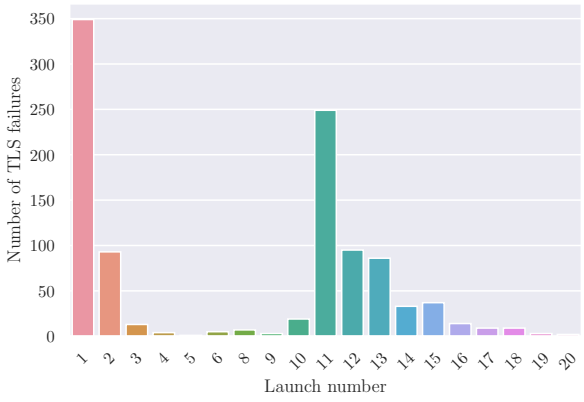



**Figure 6: Number of TLS failures in each warm-up launch for Roku platform.**

[9] https://letsencrypt.org/

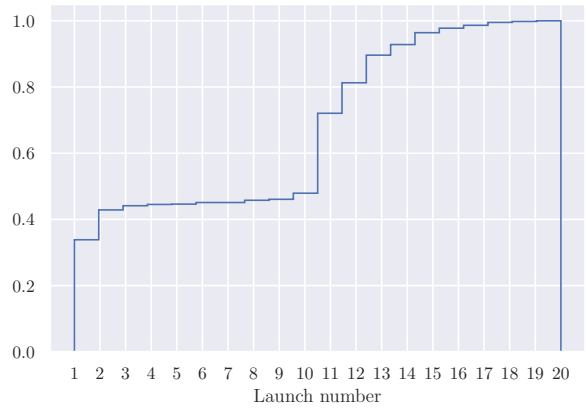



**Figure 7: CDF of TLS failures in each warm-up launch. Launches up to ten do not feature any interaction with the channel; launches after ten include smart crawls.**

## B BYPASSING TLS CERTIFICATE PINNING

In order to bypass application level certificate pinning for all of the running applications, we built a script which uses the Frida toolkit [29]. On initialization, the script collects all of the running processes and uses Objection [72] to generate an API port linked to each process. We then send a POST request to the Objection API instructing it to send the Universal Android SSL Pinning Bypass gadget [7] to the Frida server on the Fire TV as can be seen in Figure 8.

A listener is also spawned which monitors the "ActivityManager" process in Android using "`adb logcat`" which allows us to detect any new processes which spawn and create an Objection instance for them. Once the Objection instance is spawned we use a POST request as explained above to instruct Objection to send the gadget to Frida. The operation on the watchdog can be seen in Figure 8.

To prevent the system from crashing if a process is incompatible with the Frida gadget, we maintain a blacklist of processes. A process is added to the blacklist if Frida returns an error code while trying to insert the gadget or if an app crashes within ten seconds of a gadget being injected after two attempts are made.

## C OTT SPECIFIC TRACKING DOMAINS

We list the domains that we found to engage in advertising and tracking in OTT services and not listed in previous work in Table 11.

## D WEAK TLS CIPHERSUITES

*Method.* Using the pcaps from the Roku-Top1K-NoMITM and FireTV-Top1K-NoMITM crawls we extracted all TLS Client Hello messages that the OTT device sent. From these Client Hello messages, we extracted the non-ephemeral parameters: the TLS handshake version, along with a list of ciphers, extensions, compression methods, elliptical curves (supported groups), EC point formats, and signature algorithms. We concatenated these parameters — while preserving the order of all the lists — and generated a SHA-256 hash as a fingerprint for the Client Hello message.

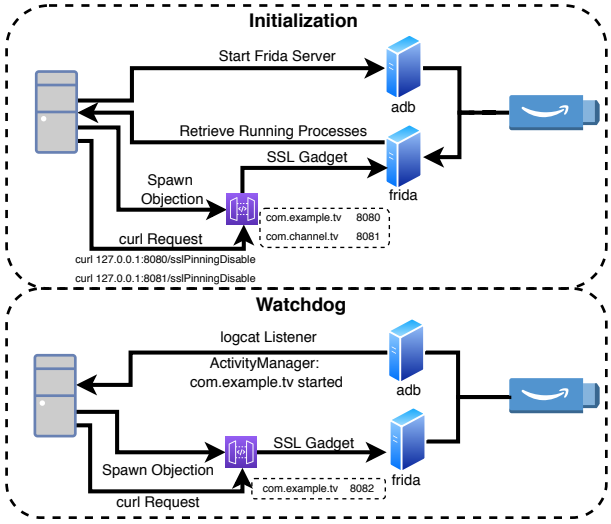



**Figure 8: Frida script pipeline.**

| Domain | Channel Count |
|---|---|
| monarchads.com | 74 |
| ewscloud.com | 31 |
| kargo.com | 25 |
| adrise.tv | 18 |
| aragoncreek.com | 7 |
| lightcast.com | 7 |
| mtvnservices.com | 7 |
| myspotlight.tv | 6 |
| brightline.tv | 3 |
| junctiontv.net | 2 |

**Table 11: Domains that collect AD ID and are not blocked by the five adblocking lists in Roku-Top1K-MITM crawl.**

We use Client Hello fingerprints to identify potentially the same TLS implementations and configurations. In other words, if two channels send out Client Hello messages with the same fingerprints, the channels are likely to have used the same TLS libraries and settings. A Client Hello fingerprint associated with insecure settings, such as using outdated TLS versions or advertising weak ciphers (e.g., RC4), implies that all channels with this fingerprint are likely affected.

*Roku Result.* Across the 1,000 Roku channels, we identify 16 distinct fingerprints. Multiple fingerprints may be associated with the same channel. On average, a channel is associated with 3.00±0.19 distinct fingerprints. Two of the 16 fingerprints appear in all 1,000 channels, and one fingerprint appears in 991 channels. This observation suggests that the majority of the channels use some standard TLS implementations provided by the platform. One of these fingerprints which appears in all 1,000 channels advertises RC4 as one of its supported ciphers. Even though no RC4 ciphers were actually used (i.e., because the Server Hello picked a non-RC4 cipher), the presence of RC4 ciphers may still face potential vulnerabilities. In particular, past years have seen multiple attacks that targeted RC4, and in February 2015, RFC 7465 prohibits the use of RC4 in TLS [58].

Additionally, there is also a long tail of fingerprints that appear in a small number of channels. Three fingerprints each appear in two different channels, and another 10 fingerprints each appear in exactly one channel. This long tail of fingerprints suggests the use of custom TLS implementations and/or configurations in channels. Of the ten single-channel fingerprints, five of them advertise the RC4 cipher across three channels: Sling TV (ranked #6), Vudu (ranked #17), and Spotify Music (ranked #25). Furthermore, two of these ten single-channel fingerprints use TLS 1.0: Amazon Prime Video (ranked #3) and Vudu — whereas all other fingerprints in our dataset use TLS 1.2. Researchers have shown in the past that TLS 1.0 can potentially be subject to attacks such as BEAST [79]. Amazon Prime Video, for instance, communicated with amazon.com and amazonvideo.com over TLS 1.0 using this particular fingerprint, although the channel also uses four other fingerprints — all over TLS 1.2 — to communicate with Amazon, Roku, and other services.

*Amazon Result.* Across the 1,000 channels, we identify 203 distinct fingerprints. One fingerprint appears in all 1,000 channels, followed by a fingerprint in 797 channels and another fingerprint in 774 channels. These top three fingerprints advertise the RC4 cipher and they use TLS 1.2. The fingerprint that is used in the most number of channels and which negotiates TLS 1.0 appears in 49 channels. In all cases, these 49 channels communicate with amazonvideo.com using this fingerprint. A total of 67 channels use fingerprints that are not used in other channels, which suggest either (i) the channels use custom implementations or configurations of the TLS library, or (ii) channels use random parameters in the Client Hello, which appears as multiple fingerprints even though only one TLS implementation/configuration is used. Unfortunately, we are unable to distinguish these two cases.

## E TRACKER DOMAINS MISSED BY PI-HOLE

| Domain | Channel Count |
|---|---|
| tremorhub.com | 66 |
| irchan.com | 42 |
| bfmio.com | 41 |
| monarchads.com | 38 |
| adrise.tv | 15 |
| digitru.st | 12 |
| bidswitch.net | 12 |
| sharethrough.com | 9 |
| adsrvr.org | 6 |
| lightcast.com | 6 |

**Table 12: Domains that receive AD ID and Serial number after filtering requests with Pi-hole (Roku-Top1K-MITM)**

## F FIRE TV REMOTE API

The Fire TV remote control app uses a version of the Discovery and Launch (DIAL) protocol [49] to locate the Fire TV on the network. Once the Fire TV's IP is located, the app queries a web server running on the Fire TV which returns the device's name along with a secure and insecure port to continue communicating on. The remote control continues the communication on the secure port using TLS by default.

In order to authenticate new clients, the Fire TV prompts the user to enter a code displayed on the TV within the app. The app uses this to generate an authentication key which is included with all the future requests the remote sends to the TV. The remote control app does not verify the TLS certificate the server provides which allowed us to recover the authentication key using a man in the middle attack. Once the authentication key is recovered, an attacker can forge remote control interactions using POST requests to either the insecure or secure port on the Fire TV's web server. This allows the attacker to install and uninstall channels, change channels, and retrieve device information. However, we believe this is not a practical attack since the attacker would need to be on the local network in order to perform the man in the middle attack and recover the authentication key.

## G VIDEO TITLE LEAKS

The following tables summarize our video title leak detection results (for a subset of channels).

| Channel Name | Video Title | Tracking Domain |
|---|---|---|
| Newsy | Newsy's Latest Headlines | google-analytics.com |
| WCJB TV-20 News | Lets Go with Livestream | scorecardresearch.com |
| CBS News | CBSN Live Video | scorecardresearch.com |
| 1011 News | Live Newscasts | scorecardresearch.com |
| WEAU News | Live Newscasts | scorecardresearch.com |
| FilmRise Kids | Barnum | spotxchange.com |
| KJRH 2 Works for You Tulsa | Sunday Night Forecast | google-analytics.com |
| News 5 Cleveland WEWS | Freddie Kitchens makes surprise appearance | google-analytics.com |
| NewsChannel 5 Nashville WTVF | Live: NewsChannel 5 This Morning at 4 | google-analytics.com |

**Table 13: Title leaks in 100 random Roku channels from the Roku-Top1K-MITM crawl.**

| Channel Name | Video Title | Tracking Domain |
|---|---|---|
| KSAT TV | KSAT-TV Livestream | google-analytics.com |
| WRAL | Severe storms batter central U.S. | scorecardresearch.com |
| WRAL | Severe storms batter central U.S. | google-analytics.com |
| Yuyu - Movies & TV | Mood Indigo | spotxchange.com |
| WTMJ TODAY's TMJ4 Milwaukee | Partly cloudy, cool Saturday | google-analytics.com |
| Hillsong Channel NOW | The Jesus Trek | litix.io |
| KJRH 2 Works For You Tulsa | In the Kitchen with Fireside Grill: Caribbean Jerk Chicken | google-analytics.com |
| WPTV NewsChannel 5 West Palm | To The Point | google-analytics.com |
| WKBW 7 Eyewitness News Buffalo | Graffiti Patio officially opens at Tappo Pizza | google-analytics.com |
| NBC News | TODAY's Headlines | omtrdc.net |
| NBC News | TODAY's Headlines | google-analytics.com |
| Popcornflix Kids | The Tuxedo | google-analytics.com |
| Popcornflix Kids | The Tuxedo | youboranqs01.com |
| Cooking Channel | Raising the Heat | conviva.com |
| Cooking Channel | raising the heat | google-analytics.com |
| Travel Channel | The Dead Files | conviva.com |
| Travel Channel | the dead files | google-analytics.com |
| PopcornflixTM- Movies.TV.Free | Planes Trains and Automobiles | youboranqs01.com |
| PopcornflixTM- Movies.TV.Free | Planes Trains and Automobiles | google-analytics.com |
| Pluto TV - It's Free TV | The Adventures of Tintin | youboranqs01.com |

**Table 14: Title leaks in 100 random Fire TV channels from the FireTV-Top1K-MITM crawl.**