Blocking Resistant Communication for Censorship Circumvention using Push Notification

Anonymous Author(s)

ABSTRACT

The rapid increase in global censorship events has stimulated a substantial growth in users relying on circumvention tools. Fighting against censors requires tool maintainers to frequently update client-side configurations and proxy IPs. However, existing methods for doing so require clients to explicitly query for updates. Further, this client-initiated communication relies mostly on ad-hoc and out-of-band channels.

This work demonstrates the utility of push notification services as an efficient and sustainable communication channel between tool maintainers and their clients. A push notification channel allows tool maintainers to update client configurations automatically without the need for clients to initiate a query themselves. We develop a general-purpose design for integrating push notifications as a signaling channel in circumvention tools. We utilize the design to integrate and implement push notifications for use in the popular circumvention tool Tor and demonstrate their utility to push bridge line updates to Tor clients. We believe such a channel creates a paradigm shift in how circumvention tools communicate with the clients and design their tools.

1 INTRODUCTION

The past decade has seen a noteworthy and disturbing surge in censorship and surveillance activities across the globe. Recent reports [18, 38] show a dangerous precedent where censorship is no longer limited to the usual suspects such as China, Russia, and Iran, but has also risen dramatically in other countries [32, 47, 57]. Freedom House [29] reports a consistent decline in the number of "free countries", with only 17% of the world's population having uncensored access to the Internet.

Growing restrictions on access to information have in turn forced citizens of censored nations to utilize circumvention technologies. As a result, circumvention tool developers have reported high user growth. For instance, the popular circumvention tool Tor had about 20k daily average concurrent users in 2016, while by the end of 2023, this number had increased to 200k.[1] Further, the conflict between censors and their citizens often results in political unrest and dissent, which causes temporary surges in the usage of circumvention tools. For example, just before the Russian invasion of Ukraine [9], there was a notable increase in the number of circumvention tool users in Russia, with Tor reporting a rise from 12.5k to over 40k [5]. An even more significant spike was observed during the September 2022 protests in Iran, with the number of users temporarily increasing from 1k to around 180k [10].

Fundamental issues with a control channel in circumvention systems. The cornerstone for circumvention tools' long-term and efficient functioning is a blocking-resistant communication channel for conveying important control information between the tool maintainers and their users. A client typically relies on such a communication channel at various stages of the tool usage, ranging from first contact (or bootstrapping) to continuous subsequent communication for obtaining updates to the circumvention protocol and configuration details such as proxy IP addresses or connection parameters (refer to Section 2.1 for details). While domain fronting [28] has long served as one of the solutions for such a communication channel, its support has been discontinued by many major providers [7, 14]. Other channels for facilitating such communication in popular tools are primarily ad-hoc and unorganized, relying on out-of-band communication such as emails, querying on forums, or visiting a website (refer to Table 1). Moreover, existing solutions for these control channels require the client to initiate requests for proxy or configuration updates. This process becomes highly challenging in environments with extensive censorship, where client requests are aggressively filtered. If tool developers could update clients' configurations without needing client-initiated connections, it would enable a more robust, sustainable circumvention approach.

Novel use of push notifications for control communication. We present the design and implementation of the first server-initiated and blocking-resistant communication channel for automated updates of circumvention clients by leveraging push notification services. Application servers traditionally use push notification services to deliver short, time-sensitive messages to mobile and desktop client applications. Push notification services use a separate connection for sending notifications, which works independently of the direct client-to-server connection. For example, when a client receives an email in its mailbox, the mail server requests the push notification service to send the user a notification informing them of the new email. The service then delivers the notification to the client device on behalf of the mail server application (as demonstrated in Figure 1). The client then directly connects with the email server to retrieve the email. Thus, even if the direct client-to-server communication channel is blocked, the applications can still receive push notifications. Moreover, push notification services are highly centralized as individual applications mostly use the same providers to send notifications—Android applications depend on Google Firebase Cloud Messaging (FCM), while iOS applications rely on Apple Push Notification Service (APNS). Therefore, blocking the notifications of individual applications would also require blocking the push notification services themselves, preventing all applications on the same platform from receiving any notifications.

[1] Note that this is a conservative estimate as the number represents only the users that rely on Tor bridges or pluggable transports (which are explicit methods for using circumvention), with the actual number reaching as high as 3-4 million (including users directly connecting via the public Tor relays).

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–14

https://doi.org/XXXXXXX.XXXXXXX

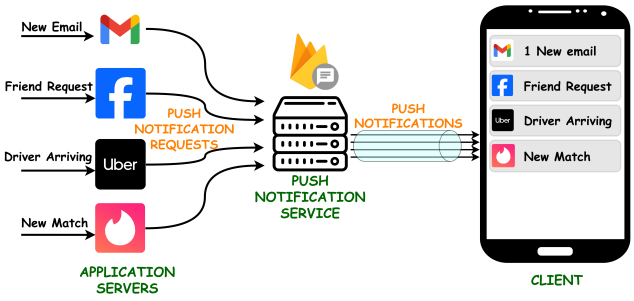


Figure 1: Architecture of Push Notification Services: On receiving updates or activity on their client accounts, application servers (e.g. Gmail) request a push notification provider to immediately send a notification to the client regarding it. The service provider sends a push notification corresponding to each such request (over a single channel) to the client.

Thorough assessment of push notification as control channel. We evaluate the suitability of push notifications to serve as an efficient control channel by testing it for multiple requisite properties. These properties include high collateral for blocking, resistance to fingerprinting, reliability, sufficient data-carrying capacity, ease of integration, and low cost. To evaluate collateral, we first note that push notification has become a widely integrated feature in mobile and desktop applications, with over 700 billion messages sent to 660 million monthly active users by a single platform in a year [8]. Further, there is currently no alternative mechanism that offers a similar functionality. Some providers (*e.g.*, Apple) even use the same originating domains for both their push notification service and other services (*e.g.,* iMessage). As a result, blocking push notification services outright can cause significant collateral for the censors.

To affirm this further, we performed large-scale and longitudinal measurements across regions and networks to test for potential censorship practices against push notification services. We used the remote HTTPS censorship measurement technique Hyperquack [46] to test the reachability of push notification domains across 188 countries. The technique relies on crafting and sending HTTP(s) requests to the target network's open web servers and observing the response for any interference by network intermediaries (such as middleboxes). Our analysis of over 3.6 million measurements indicates that only a few ASes ($< 0.01\%$ measurements) occasionally interfere with push notification domains with no evidence of blocking over the measurement period of a year by popularly known censored countries such as China and Russia. We also evaluated push notifications for other requisite properties (reliability, ease of integration, cost, etc.) for an effective control channel and found it to be a suitable channel for delivering control information (refer to Section 4).

Integration and deployment efforts in popular circumvention tools. While the use of push notification as a full-fledged circumvention channel has been preliminarily studied [56] and found to be unrealistic for practical purposes, in this work, we establish the efficacy of push notifications as a control channel and lay down the design for integrating it as a transport for sending control information in circumvention tools. We then demonstrate the practicality of such a transport by integrating it for use with Tor. We modify the popular and open-source Tor client application, Orbot [17], and provide support for receiving and automatically updating bridge lines using push notifications. We are working with the Tor anti-censorship team and Orbot developers to deploy push notifications transport as a control channel soon. We also highlight various challenges encountered in the integration and realistic deployment considerations (e.g. privacy implications) along with their solutions in Section 5.1. Further, we collaborated with another popular circumvention tool CTZ,[2] which is currently in the advanced stages of integrating push notification into their ecosystem.[3]

Applicability and potential beyond popular circumvention tools. Beyond direct contribution towards alleviating control channel challenges of popular and widely used circumvention tools, the push notification transport has immense potential for mitigating problems that cannot be addressed with existing client-initiated control channels and for catalyzing the latest circumvention strategies. For example, the emerging circumvention designs such as Proteus [53] and WATER's [25] one of the major challenges of communicating protocol updates can be addressed using push notifications (refer Section 6 for details).

Overall, the integration into popular circumvention tools clearly demonstrates the potential and utility of a server-initiated communication channel built using push notifications. We believe such a channel advances how circumvention tools communicate with clients, significantly enhancing their capability.

2 BACKGROUND

2.1 Circumvention Tool Life Cycle

In this work, we focus on circumvention tools that rely on infrastructure located outside the censored region to provide access to blocked content. In the typical life cycle of such a circumvention tool, the client first performs a bootstrapping step, which we generalize here as being facilitated by a single configuration server. This bootstrapping step can include the distribution and exchange of proxy IP addresses, encryption keys, or connection configuration parameters. The connection to the configuration server happens using one or several highly censorship-resistant out-of-band channels, sometimes referred to as signaling channels [52]. The client then connects to the circumvention tool to access censored content until their connection fails due to censorship or a service outage. At this point, the client must repeat the bootstrapping step by contacting the configuration server using the same out-of-band channels. This process repeats indefinitely during the circumvention tool life cycle.

We give an example of the circumvention tool life cycle for a proxy-based circumvention tool architecture in Figure 2. During the bootstrapping step, the client receives connection information for a proxy from the configuration server. When the proxy becomes blocked, the client contacts the configuration server again to receive a new proxy. This life cycle generalizes to other architectures as well. For a circumvention tool like meek, which uses domain fronting [28], the bootstrapping step requires sending the front domain and destination URL of the proxy server to the client. If the front domain becomes blocked by its TLS SNI or switches cloud providers, the

[2]Anonymized for submission.
[3]We could not directly integrate push notification in CTZ as its source code is not publicly accessible.

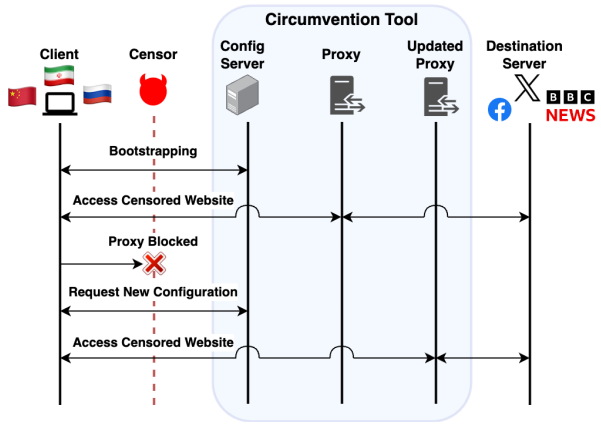


Figure 2: Circumvention Tool life cycle: A tool undergoes bootstrapping, accessing censored content, getting the proxy blocked, and obtaining new configurations for subsequent connections.

client's connection will cease to work and the client will need to ask the configuration server for an updated configuration.

Previous work shows that determined censors can detect and block circumvention tools despite the measures taken by the tool maintainers [1, 19, 23, 27, 31], effectively requiring continuous innovation in the obfuscation methods, updated configurations, or new proxies. The success of the existing tools greatly depends on the effectiveness of the communication channel between the client and the configuration server to convey such updates. However, the current process of obtaining updates and other relevant information from a tool is primarily ad-hoc and unorganized, with popular tools relying on channels such as emails or instant messaging chat bots. For example, Tor users may request bridge information through either email, Telegram, the domain-fronted service Moat, or visiting Tor's bridge distribution website. Some of these methods require the user to manually update their Tor configuration, and may be blocked or ineffective in their country [15, 16].

2.2 Push Notification

Push notifications enable application servers to deliver important or time-sensitive messages (*e.g.,* notification of a new email) to client devices, even when the user is not actively accessing the client application. Essentially, push notification services function through a separate channel independent of direct client-to-application communication. The push messages are sent through push notification service providers such as Google Firebase Cloud Messaging (FCM) [33], Apple Push Notification Service (APNS) [20], and Microsoft Push Notification Services (WNS) [41]. While many other push notification service providers also exist, most of them utilize the same underlying popular providers.

Push notifications have seen significant growth in their usage across the globe. Currently, FCM and APNS dominate the mobile push notification landscape [12]. FCM also serves the vast majority of browser push notifications, delivering notifications to Chrome and other Chromium-based browsers, accounting for over 95% of all web-based push notifications [6]. Regarding usage, a third-party provider reported sending over 700 billion Android and iOS push notifications to about 660 million monthly active users [8]. In terms of revenue, a study projects $\approx 4\times$ increase from about 2 billion to 8 billion USD by 2032 [13]. WNS and APNS are responsible for push notifications to all Windows and Mac OS systems.

The implementation of push notifications generally conforms to a publish-subscribe model. First, a newly installed application must undergo an enrollment procedure by contacting a push notification provider. Following this, the client device acquires a device and app-specific registration token. Next, the client sends this token to the application server. To send a notification, the app server sends a request to the push notification provider. This request includes a custom payload and identifies the set of clients that should receive the notification by their registration tokens. The push notification provider then delivers the notification to the target devices.

3 NEED FOR EFFECTIVE SERVER TO CLIENT CHANNELS

Censorship circumvention systems regularly face blocking attempts by censors. When successful, these attacks disconnect tool clients from their proxies and the tool back-end. Numerous studies [1, 19, 21] have exposed the different ways in which censors identify the use of circumvention services, including attacking the bootstrapping process [1], fingerprinting the usage of the tool by their differentiated implementation of standard TLS libraries [1, 4], and actively probing for the presence of a circumvention service [27]. Such consistent efforts by censors force tool maintainers to come up with workarounds and regular updates to their protocols and systems. While part of the resolution is to develop systems that support on the fly updates of parameters and protocols, the main challenge for the tool maintainers is to disseminate these updates and associated information to their end users. Solving such challenges requires the presence and availability of an unblocked channel between the tool developers and the tool users.

However, currently, users rely on temporary and mostly ad-hoc methods of communicating with the back end. These methods involve using the out-of-band channels to query for information by visiting websites, sending emails, or inquiring around in online forums and groupchats [15, 16]. Not only is the current process cumbersome, where users at times have to request updates out-of-band and manually load the configurations into their software client, but in some instances can also pose risk, when users have to ask around in public online forums.

For a concrete understanding of the existing methods, we surveyed popular circumvention tools for the techniques they employ for such communication. Table 1 highlights the communication methods and entities for initial (first) contact, subsequent communication to the circumvention transport, and communication post-blocking of five popular circumvention tools. For example, Snowflake users rely on built-in configurations or a domain fronted control channel called Moat to learn how to configure their client and communicate with the Snowflake broker. Users then repeatedly poll the Snowflake broker and connect to the bridge over currently available Snowflake proxies. When a censor blocks Snowflake by targeting the connection to the broker or blocking access to STUN servers, the client must obtain a new configuration to circumvent the block. Overall, we

Circumvention Tool	Control Channel (First Contact/Subsequent Post-block Contact)	Circumvention Transport
Tor (obfs4)	Email, Telegram bot, bridgedb, bridgedb website, and Domain Fronting	obfs4 server
Tor (Snowflake)	Built-in configuration, or domain-fronted Moat request	Broker rendezvous and WebRTC connection to one or more proxies
Lantern	Domain Fronting	obfuscated/unobfuscated proxies
Psiphon	Email responder	Simultaneous connections to different proxies
Conjure	TLS requests to benign websites within an ISP hosting Conjure	Phantom hosts (proxies within the ISP)

Table 1: Communication channels for various stages of the circumvention tool usage for five popular tools. The table lists the methods for the first contact to obtain the initial configuration and the successive contacts post-blocking of the circumvention tool along with the circumvention transports used by them.

observe that popular tools mostly rely on email or domain fronting to contact and receive updates, with some of them relying on instant messaging bots, or sending steganographed web requests to a supportive Internet service provider. Note that while domain fronting has been extensively used in the past decade, its usage has considerably declined because of decommissioning by many fronting service providers [7], leaving a void for effective control channels.

Another key observation from analyzing the communication methods of existing tools is that the client has to initiate a query to the server for all tasks. These tasks include obtaining new configurations or proxy IPs as well as informing the tool maintainers of the intent to use their service (registration). Initiating queries can be difficult for the users, especially after large-scale blocking events. Such events tend to leave the users stranded and clueless as the default modes of communication are generally severed (refer to Section 5 for details). However, if there was a way that allowed tool developers to (periodically) update the configuration of the clients without the need for the clients to connect to them, a systematic and sustainable process of circumvention could be achieved.[4] The presence of an uninterrupted server-to-client channel not only helps solve existing problems, but can also provide tool maintainers the flexibility of choosing the set of users and the kind of configurations they would want to send at select time instances. For instance, in Conjure [30], the client does not receive any confirmation that its registration was successful. With an independent server-to-client channel, the client could be more reliably informed about the success of its registration. In a nutshell, a server-to-client channel effectively switches the burden of ensuring sustainable connections from normal users to tool maintainers, who are usually more experienced and technically capable of handling such situations and taking appropriate actions.

3.1 Required/Relevant Properties

For the success of a channel aimed at ensuring sustained one-way communication, there are specific properties that it should satisfy:

High Collateral The communication channel should not be trivial for the censor to completely block. There should be pervasive usage of the channel by benign and legitimate services, such that blocking it carries a high collateral damage for the censor.

Resistance to Detectability It should be difficult for the censor to detect the usage of the channel for providing circumvention related information by passively analyzing the traffic patterns or actively probing for extracting out information.

Reliability The channel should be able to deliver the required information reliably to the receiver. In case of potential information loss, there should be overlay mechanisms in place for ensuring reliable delivery.

Data Carrying Capacity The channel should be able to support sufficient data carrying capacity without leaving an observable trace of not using the channel for its intended purpose. For example, if the channel supports extremely low data transfer in the order of a few bytes, sending a complete configuration may require transferring it in multiple iterations, making it prone to passive traffic analysis.

Ease of Integration Integrating the channel functionality in a circumvention tool should not be very complex. This promotes streamlined integration and wider adoption.

Low Cost Using the channel should not be very expensive as it would then become difficult for the tool maintainers to provide circumvention services to a large user base.

4 PUSH NOTIFICATION FOR AIDING CIRCUMVENTION

We now investigate the suitability of push notification as an effective control communication channel by testing it for the desirable properties laid down in the previous section.

4.1 High Collateral

Push notifications have been widely adopted and integrated into mobile, desktop, and web applications as the only means of timely communicating relevant information to users. Currently, push notifications are not only an integral part of users' daily interactions with their laptops and mobile phones but also provide a competitive business edge to app developers for delivering quality service. Importantly, no alternative mechanism offers similar functionality and is thus hard to replace if blocked. Further, some providers even use the same endpoints as the push notification service to provide other services. For instance, Apple piggybacks its iMessage service using the APNS systems employed for push notifications. Blocking a push notification service can thus cause significant collateral damage and catastrophic effects on the end-users and businesses that rely on it.

To strengthen this claim we performed active measurements to study the availability and reachability of push notification services across regions and over time.

[4] Note that, while the analysis is performed on popular circumvention tools, the observations apply to any existing or future tool that requires communication between users and tool maintainers.

4.1.1 Push Notification Current Blockability. To effectively leverage push notification services to bypass censorship, it is essential that they remain accessible and are not blocked, especially in regions with strict censorship controls. Thus, we perform extensive measurements to identify any potential censorship targeting push notification services on a global scale. Our study specifically examines Google FCM, the world's leading push notification provider, and evaluates its accessibility across different regions.

Methodology To measure the blocking of push notification domains, we rely on an efficient HTTPS blocking detection technique termed Hyperquack [46]. The technique builds upon the original Quack [51] system that is used to remotely measure the blocking of URLs by sending unsolicited HTTP requests with target URLs to Echo servers and analyzing the corresponding response to check for blocking behavior. Quack is limited to detecting only HTTP filtering and relies on echo servers, which can be scarce and difficult to find. Hyperquack goes beyond this limitation by crafting HTTP(s) requests and sending them to actual open web servers in target networks. The web requests are crafted to contain the domains under test in either the TLS SNI field or the HTTP Host header. After sending the requests, the Hyperquack system monitors the response. A response from the middlebox as an action to enforce URL/keyword blocking would be anomalous and is characterized by TCP RST(s)/FIN, censor blockage *etc.*

We conducted two extensive Hyperquack measurements: a short-term global scan from October 18, 2024 to November 18, 2024, and a longitudinal scan of China and Russia—both of which have previously interfered with Google FCM but lifted blocking due to the significant disruption caused [3, 56]—from October 1, 2023 to October 31, 2024. We selected Google FCM domains as target domains for both scans. The full list of these domains can be found in Table 2 in the appendix.

Results In our one-month global scan, we collected 2,818,901 measurements to public web servers distributed across 188 countries. We then aggregate results by AS. We consider and analyze the anomaly results of the ASes where we were able to send and successfully receive Hyperquack measurements to at least more than 20 servers within the AS. In our analysis, we find negligible interference of connections to the FCM endpoints, with majority of the ASes ($> 99\%$) returning either no anomalies or anomalous responses for less than 1% of all measurements within the AS. We show the top 10 ASes with maximum anomalous response in our measurements in Figure 3. Interestingly, the high anomaly ASes ($> 8\%$) all belonged to countries that lack pervasive censorship, potentially due to transient network and policy updates in those ASes. These measurements indicate that push notifications are currently (as of November 2024) functional worldwide, with no existing nation-state censors engaging in their blocking. These results also corroborate with those of Xue et al. [56], who similarly did not find evidence of targeted censorship of FCM endpoints.

Our longitudinal analysis scan comprised 864,079 Hyperquack measurements to China and Russia over one year. The anomaly rates over the year for both countries are shown in Figure 4. Overall, the results clearly show that Russia and China did not engage in any large-scale blocking attempts over the year, with overall anomalies staying close to 1-2% on average for China and $< 1\%$ for Russia.

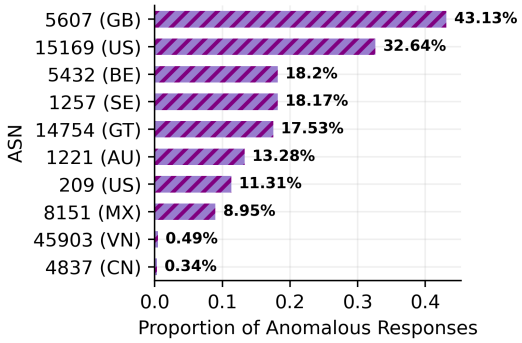


Figure 3: *Hyperquack* Results: Top 10 ASes (among 1,085 analyzed) in decreasing order of the fraction of anomalies observed. The anomaly rates fall below 1% starting with 9th ranked AS (45903). More than 99% of the ASes reported anomaly rates of zero or close to zero in our measurements.

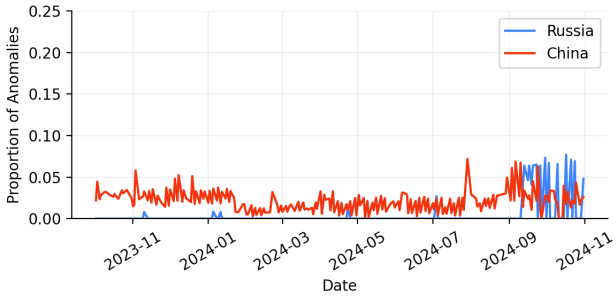


Figure 4: *Hyperquack* Results for Russia and China: Proportion of anomalous Hyperquack measurements for Russia and China from October 2023 through October 2024.

Specifically, in China, only 5 ASes account for 72% of all anomalies, while in Russia, 3 ASes account for over 95% of all anomalies. Given that a small number of ASes are responsible for most of the anomalies in these countries, we assume that there have been no attempts to interfere with push notifications from Google FCM.

4.2 Resistance to Detectability

Clients receive push notification messages from a fixed set of centralized URLs, irrespective of the application generating the request (FCM for all Android-related and APNS for Apple-related). A typical client, on average, receives 46 such push notifications per day [45]. This allows a single (or even a few) push messages of the circumvention tool to easily blend in with the existing set of push notifications received by the client—making it hard for the censor to identify circumvention tool notifications based on passive traffic analysis. Further, push notifications naturally defend against any active probing mechanisms, as the service provides one-way communication from the server side. Lastly, the TLS-encrypted communication between the push service provider and the client eliminates the possibility of keyword-based blocking using DPI.

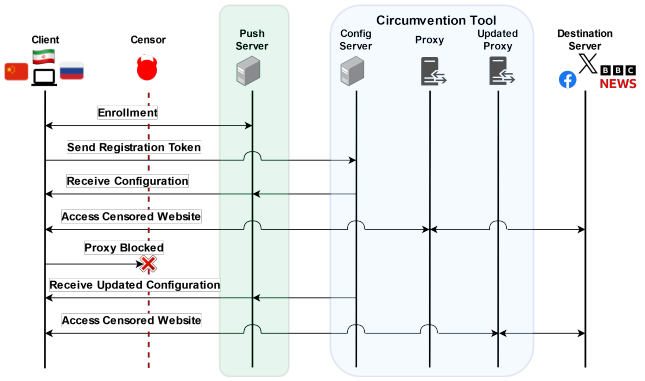


Figure 5: Circumvention Tool Life Cycle with push notification integration for tools with direct proxy connection.

4.3 Data Carrying Capacity & Reliability

Push notification services operate in real-time to deliver time-sensitive information that requires the user's attention. Both FCM and APNS allow a single push message to carry up to 4 KiB of payload, while WNS allows for 5 KiB. A maximum of a few messages are sufficient to deliver the required circumvention tool information. For instance, Tor obfs4 would require sending a bridge line with a transport name, IP, fingerprint, and connection parameters, requiring less than a KiB. Additionally, push notifications as a transport automatically ensure the delivery of content. Even if the client is not connected to the Internet, the push notification message is queued and stored for a default of four weeks. Once the client regains Internet access, all the queued messages are automatically delivered [11].

4.4 Cost & Ease of Integration

The first-party push notification services are offered at no additional monetary cost, [5] a significant advantage over other high-availability systems using cloud providers or blockchain [28, 34]. Moreover, integrating and customizing the functionality of push notifications is reasonably easy (as demonstrated in Section 5). The ability to process push notification messages by the client application without any visible prompts to users allows for a seamless user experience.

5 DEPLOYMENT & CASE STUDIES

We now present the steps for integrating push notifications into circumvention tools as a transport to facilitate sending control information. We update the original circumvention tool life cycle and highlight the change in steps when using such a transport (Figure 5 depicts the modified version). The client first registers with a push notification provider and obtains a device token for receiving push notifications. Subsequently, the client, in its initial contact to the configuration server, also sends the push notification token which the server can later use to send notifications via the push notification service. Next, the configuration server sends the configuration (proxy IP and other relevant details) to the client via a push notification update. The client then connects to the proxy (directly or with the help of multiple intermediaries, depending on the tool design) and accesses the censored content. During this stage, the configuration server can preemptively update the client's configurations or proxy via push notification updates as and when required. When proxy connections are blocked by the censor, the client's configuration can be automatically updated by the configuration server, which pushes updates via the push notification service.

It must be noted that the current steps outline the process to utilize push notifications during all phases of the client and circumvention tool communication. However, tool maintainers can freely choose it for specific steps or purposes according to their requirements and the tool architecture.

Case Studies: Having laid down the foundations of using push notifications to send unidirectional updates to clients, we will now demonstrate concrete instances of integrating and implementing them in circumvention tools. To that end, we collaborated with popular circumvention tool developers and explored the integration of push-notification-based transport while highlighting the practical challenges associated with realistic deployments.

We start by detailing how we integrated push notifications for use with Tor, one of the oldest and most popular circumvention tools. Subsequently, as a collaborative effort, we present the case study of a popular tool CTZ.[6] We describe how CTZ currently handles out-of-band communication with clients and highlight the challenges and motivations CTZ sees in integrating push notifications along with their integration efforts.

5.1 Tor Integration

Tor [26] is a volunteer-operated network of relays that provides anonymity and anti-censorship. In regions with extensive censorship, Tor users rely on the usage of bridges [15], which function as non-public entry relays to the Tor network. These bridges support one or more pluggable transports (PTs) [16], which wrap the connection between the user and the bridge, as a means to evade blocking.

Bridge and PT configurations are communicated to the user in the form of *bridge lines*. Some PTs require the bridge lines to be kept private from censors because knowledge of the information contained in them would enable a censor to block access to that bridge. For example, bridges that support the obfs4 PT [44] include the IP address of the bridge in the bridge line, which may be easily added to a block list if discovered. Other PTs leverage trade-offs rather than secrecy to provide blocking resistance. Snowflake [23] and meek [28] both use domain fronting as part of their anti-censorship strategy, forcing the censor to make a choice between blocking the popular, high-profile front domain in order to block access to the tool. Snowflake additionally relies on the large and ephemeral pool of temporary Snowflake proxies, which are not disclosed in the bridge line but provided to the user as part of an interactive protocol upon use of the system. As a result, Snowflake and meek bridge lines may be public, but they still need to be distributed to the user.

Bridge lines are distributed through a variety of methods, dependent on the level of secrecy required by the PT, and the extent to which Tor is censored. To obtain bridge lines for PTs that require secrecy, users may request bridges from Tor's bridge distribution system through several channels. Both Orbot and Tor Browser have an integrated interface for requesting bridges through Moat [49], a domain fronted connection to the distribution server. Users may also

[5] A developer subscription is needed for APNS

[6] Anonymized for review.

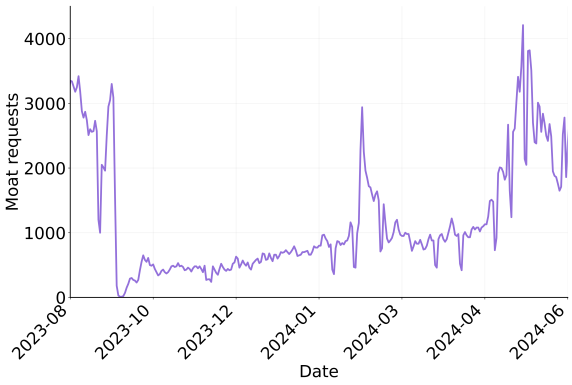


Figure 6: A drop and slow recovery of domain fronted Moat requests from Iran in September 2023.

request bridges from Tor's Telegram bot,[7] through email, or directly from the website https://bridges.torproject.org. The effectiveness of the channel, resistance to enumeration, and user experience varies and may depend on the censorship or support resources available in the user's region. Another method is to ship built-in bridge configurations with the client software. Both Tor Browser and Orbot use this method to provide anti-censorship defaults that are effective for many users. This distribution method, while simple, is best suited for PTs that function well with public bridge lines. It is also slow and costly to release a new software update if a censorship event requires a modification to the PT configuration.

Tor offers a circumvention settings[8] service to help guide users in selecting distribution methods and PTs that work well in their region. For PTs with public bridge lines, it also offers those bridges lines directly to users. However, this control plane service itself relies on a censorship-resistant channel. At the moment, both Orbot and Tor Browser use domain fronting provided by Moat for this channel, but several recent events concerning the configuration of this channel have led to losses in connectivity from which users struggled to recover. We show the impact of one such event using publicly available Moat usage metrics published by CollecTor[9] in Figure 6. In September of 2023, cdn.sstatic.net, the front domain used by both Orbot and Tor Browser for Moat at that time, switched its cloud hosting provider from Fastly to Cloudflare.[10] This immediately caused the channel to fail, and recovery required an updated app installation.

Push notifications offer an alternative cost-effective and highly censorship resistant means for delivering updates to anti-censorship configurations and settings in response to blocking events or configuration failures. They also provide a significant usability enhancement, enabling the immediate notification of users in event of a required settings update. In this work, we focus on their use for providing access to Tor's circumvention settings service and public bridge line updates, but they could be a powerful tool for the distribution of secret bridge lines as well. Secret bridge lines are more easily blocked, and the bridges themselves are more prone to churn as volunteer bridge operators join and leave the network. Allowing users to subscribe to updates would facilitate a speedy and effective recovery from bridge blockages and outages.

We are working with the Tor anti-censorship team and Orbot developers to deploy push notification support.

5.1.1 Implementation Details. We implemented push notifications as a censorship-resistant control plane channel for the distribution of updates to Tor's circumvention settings service. We have integrated support for this channel in Orbot [17], a mobile application that can function as a full-device Tor network VPN. The application is open source and maintained by the Guardian Project, allowing us to easily modify and present a prototype.[11] Since Orbot is an Android application, we use Google's Firebase Cloud Messaging (FCM) [33] as the push notification service for this work.

Orbot Integration Users can opt-in to the use of push notifications in Orbot's settings, shown in Figure 7a. Once the user has enabled support the Orbot client registers with FCM, the Push Notification Provider, to get a unique device token. The client then sends this token, along with an encryption key, to our push notification distributor using the existing control plane channel for the circumvention settings service. This is a one-time step—after registration, the user is effectively subscribed to updates and will be able to receive these updates via push notifications even if this control plane channel fails.

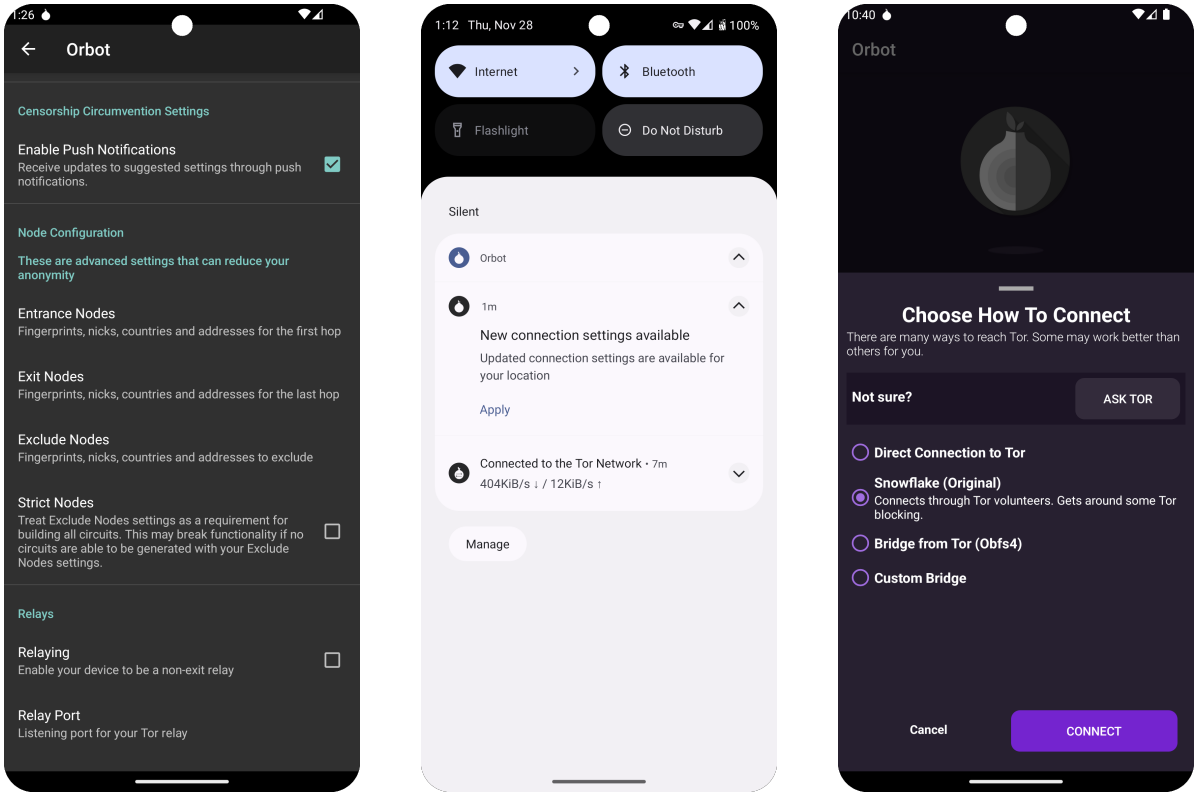
If the user has granted the notifications permission to Orbot, they will receive a notification when new circumvention settings are available, shown in Figure 7b, if the available circumvention settings for a user's location have changed. Users can tap this notification to apply the suggested settings. This will open up the settings configuration view, with the suggested settings displayed, shown in Figure 7c. The user can then connect to the Tor network using these settings, which will be saved in the application `Prefs` for future use.

From the user's perspective, the added push notification service should be transparent, but require informed consent. Our implementation requires users to opt-in to the use of push notifications but ensures that user interaction is not required after this step for the service to function. As Orbot already requests the permission to post notifications, users will not need to grant the app any additional permissions beyond its default settings.

Push Notification Settings Distributor The push notification settings distributor is responsible for managing user subscriptions and pushing updates to users via FCM, the push notification service. The distributor listens for user registrations and stores the provided FCM token and key in a database. The distributor does not store any additional identifying information about registered users. We implemented the distributor modularly in Go as an rdsys distributor,[12] so that it may be easily deployed along with the existing distributors, and easily adapted to distribute secret bridge lines later.

The distributor periodically fetches up-to-date circumvention settings in the form of a json file exported by the circumvention settings

[7] https://gitlab.torproject.org/tpo/anti-censorship/rdsys/-/blob/main/doc/telegram.md
[8] https://bridges.torproject.org/moat/circumvention/map
[9] https://metrics.torproject.org/collector.html#type-bridgedb-metrics
[10] https://archive.torproject.org/websites/lists.torproject.org/pipermail/anti-censorship-team/2023-September/000314.html
[11] The distribution server and Orbot integration are public and open source. We omit the links during the review process
[12] https://gitlab.torproject.org/tpo/anti-censorship/rdsys/-/blob/main/doc/distributors.md



(a) (b) (c)

Figure 7: Orbot Integration — (a) The user opts in to receive push notifications updates for circumvention settings. (b) When updates are available, the user will receive a notification that can be tapped to apply the new settings. (c) Settings are saved and can be applied from the connection configuration view.

service.[13] It then calculates a diff of the previous and updated settings. If any settings have changed, the distributor sends the updated settings in a push notification via FCM to the user.

All push notifications sent by the distributor to the user are signed and end-to-end encrypted. We ship the distributor's public key with the Orbot integration, to be used to check the authenticity of received circumvention settings and prevent malicious third parties, including the push notification service, from modifying provided bridge lines and configurations in an effort to influence a user's entry-point to the Tor network. Although these circumvention settings are already public, we encrypt the contents of our push notification messages to reduce the information available to third parties and to easily enable future use of this channel for the distribution of secret bridge lines.

5.1.2 Privacy Analysis. While third party services like push notifications offer highly censorship-resistant communication channels, they also come with privacy risks. This is not unique to push notifications—domain fronting through a centralized cloud provider also poses risks to user anonymity and privacy. In this section, we fully explore the privacy risks in our use of push notifications for sending Tor users updates to circumvention settings and bridge lines, and how we mitigate these risks in our design.

[13] https://bridges.torproject.org/moat/circumvention/map

Push notifications require a constant background connection between the user's device and the push notification service in order to function. When the service receives a message bound for a user's token, it can then immediately send that message to the user's device through this connection. This connection persists and reconnects as users change networks, giving the push notification service the ability to track users across different networks and monitor whether or not a user is online. This privacy risk is not specific to our use of push notifications, and this background connection will exist for all users with Google services enabled. Below, we focus on the additional privacy risks of using push notifications with Tor applications.

Preventing the leak of usage patterns One of our primary concerns is to not leak the Tor usage patterns of users to either the push notification service or our distributor. To this purpose, messages sent through push notifications are never triggered as a result of user behavior. They are only sent in response to censorship configuration updates that affect all users in a sufficiently large anonymity set equally. In our current implementation, updates are triggered by censorship events, changes to the implementation of circumvention tools, or service outages only, and can not be tied to the actions of individual users.

Preventing guard/bridge discovery Tor relays that act as entry points to the Tor network are sensitive as they often see direct connections from users. A guard discovery attack is any attack that allows an adversary to determine the entry point of a particular Tor client. This information can be used to target entry point relays for coercion or collect analytics data that can be used for de-anonymization.[14] For anti-censorship users of Tor, bridges serve as their entry points to the Tor network.

While our current use of push notifications distributes only public settings, additional care should be used if this channel is adapted to provide secret bridge lines to specific users. We already encrypt the contents of the push notification messages as a forward looking measure to prevent the push notification provider from seeing a user's bridge information. However, even with encryption, pushing immediate updates in response to bridge outages or blockages may also leak a user's previous bridge configuration to the push notification service via metadata. An adversarial service could note when a user receives an update and cross-reference that with public metrics on Tor bridge availability. This can be further mitigated by batching updates in a sufficiently large anonymity set of bridges and users.

Authentication of circumvention settings Not only do we wish to keep a user's bridge configuration private, we also aim to protect it from modification by a malicious third party. An adversary with the ability to send configuration settings to a user at will could successfully manipulate a user into creating a Tor circuit using an entry relay that they control. This would pose a significant threat to the user's anonymity. We mitigate this attack by cryptographically signing all messages from our push notification service. Our public key is shipped with the Android application and used to verify the authenticity of the settings.

Limiting collected information The only information collected by the tool developers is the FCM token needed to direct the push notification message to the subscribed user, and the user's encryption key. While the push notification service itself could ostensibly map these tokens to users, these tokens are distinct across different apps, and we do not collect or store any additional information that could be used to uniquely identify users. We specifically avoid collecting or logging user IP addresses, timing information about when the user registered, or even a user's country code. This last measure may be over-cautious, as the set of users for each country is sufficiently large, but while we roll this out, we have opted to send circumvention settings updates to all users and perform the country check locally on the user's device to further protect this information.

Again, additional care should be taken to protect users if this distributor is adapted for subscriptions to specific bridges. A database that stores a bridge line with a user's registration token in order to push updates if that bridge goes offline or becomes blocked is a potential target for an adversary who, knowing a user's registration token, wishes to perform a guard discovery attack, or get a list of user identities associated with a specific entry relay.

[14] https://spec.torproject.org/vanguards-spec/index.html#introduction-and-motivation

5.2 Tool CTZ Case Study

Now we provide the real-world experience of a popular circumvention tool CTZ,[15] which has millions of monthly active users across the globe. The organization behind CTZ is primarily focused on defeating censorship with over a decade of experience in censorship circumvention. CTZ operates in some of the most repressive regimes in the world, including China, Russia, and Iran.

Users interact with CTZ like a VPN: once the tool is installed and enabled, browser traffic is directed through the tool as necessary to circumvent censorship in the user's region. Behind the scenes, CTZ operates a global network of proxy servers. Each CTZ client is assigned a small number of proxy servers and periodically receives new assignments as individual servers are blocked.

5.2.1 The Control Plane and the Challenges. At times, a CTZ client can become disconnected from all of its assigned proxy servers. As a special case, all CTZ clients experience this state when starting up for the first time—a process referred to in this context as bootstrapping. This disconnected state can occur again if a client has been turned off for a sustained period of time and all of the clients' assigned proxy servers have been rotated. A client might also become disconnected if a censor blocks all of a client's assigned proxy servers. This is a common phenomenon for users living in highly censored countries, and the CTZ team has observed numerous instances when a censor blocks hundreds or even thousands of IP addresses at once, leaving a large share of users disconnected in an instant.

To ensure that the CTZ back-end is able to communicate with clients in disconnected states, CTZ relies on alternative communication mechanisms. These alternative channels are not suitable for proxying complete user traffic, due to bandwidth and cost limitations. However, these channels are generally more reliable and harder to recklessly block for the censor in comparison to the direct client-to-proxy communication. This set of alternative communication channels becomes what we call the control plane of CTZ.

Currently, the cornerstone of CTZ 's control plane is domain fronting. It was introduced in 2015 [28] as a method of sending traffic to a blocked domain (or the proxy server) via an unblocked domain. All the cleartext fields in the connection, point to the unblocked domain, while facilitating a mechanism to contact the blocked domain inside the encrypted communication. However, domain fronting requires support of different platforms with a high collateral to host such a service. But over time all major providers have discontinued support for fronting services with even the last few major providers recently joining the list [7, 14]. This leaves a void for channels that could be considered for the control plane. We corroborate this impact with data which clearly demonstrates high error rates with domain fronting requests in censored regions. Figure 8 depicts two such recent periods where domain fronting could not be reliably used in censored regions due to errors in connection. These anomalies occurred in regimes known for extreme Internet censorship and affected millions of CTZ users. CTZ has not determined whether these events reflected deliberate attacks on CTZ 's use of domain fronting. Regardless, during this time, the CTZ control plane was crippled.

Even outside of such anomalous periods, the error rate for domain fronting requests can be significant. CTZ 's user facing team

[15] The tool name is anonymized for submission.

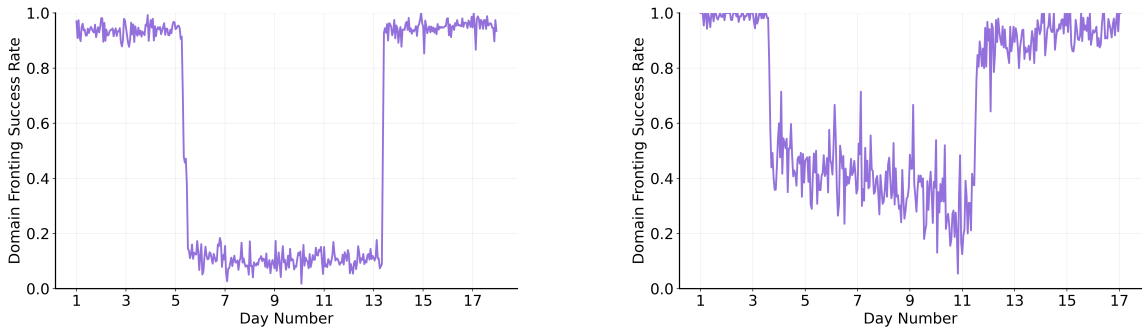


Figure 8: Two recent periods of elevated blocking in two different regions. The success rate of domain fronting significantly dropped during blocking events. The exact dates and regions of these events have been anonymized.

sees significant messages on a daily basis from new users who report a failure of the client to connect to any proxies. This failure to bootstrap is primarily due to a failure to connect to the domain fronting service. This phenomenon is so common that in some regions, the CTZ team has observed users posting instructions on how to bootstrap CTZ using alternative VPNs (this process allows the CTZ client to fetch its initial configuration through the alternative VPN). A senior member of CTZ 's customer success team reported this bootstrapping problem as the most important issue currently impacting CTZ 's user experience.

Fundamentally, domain fronting (and other such channels) is limited as a control plane transport because it is client-initiated. If CTZ needs to send messages to clients via domain fronting, the back end must queue these messages until the client connects. If the client is disconnected from its proxy servers and domain fronting requests are not functioning, these messages will never be picked up and displayed to the user. Other components of CTZ 's control plane (*e.g.*, telemetry services or one-time emailers) suffer from the same problem or only support one-way communication (client to server).

This limitation impacts CTZ users most during times of elevated blocking. It is during these times that CTZ clients are most likely to be completely disconnected from the proxy network. The CTZ team in these situations feels a strong need to be in touch with users, letting them know about network and software updates or simply that CTZ engineers are working to get them back online. During times of elevated blocking in a region, CTZ 's customer success team tends to be overrun with support requests (which they receive through multiple mediums, including Telegram, Twitter, GitHub, and even domain fronting). Figure 9 depicts spikes of such support tickets in a region during elevated blocking. The fact that there are thousands of users desperately inquiring and waiting for updates necessitates the need for some information channels from the control plane to the users. Such an ability to reliably communicate awareness of an issue to users would be instrumental in helping mitigate these spikes and ensure the smooth functioning of CTZ.

5.2.2 Utility of a Push Notification Transport. The addition of a push notification transport to CTZ 's control plane improves the ability for the CTZ back-end to serve new proxy configuration to disconnected clients. This helps keep CTZ clients connected to the

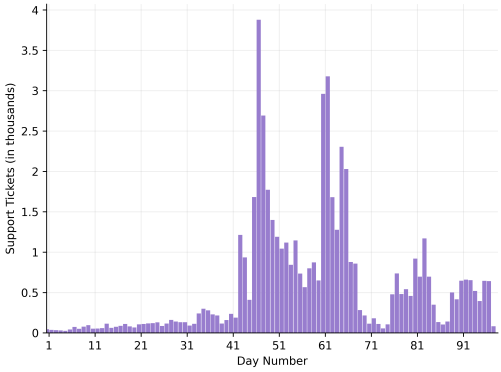


Figure 9: Support requests received by CTZ during elevated blocking in a censored region.

proxy network and in turn, helps keep CTZ users connected to the open Internet. The addition of a push notification transport also helps CTZ clients connect to the network when bootstrapping, improving the experience for new users. For users who experience high error rates with domain fronting, this improvement could be substantial.

A push notification transport also adds a new server-initiated communication channel to the CTZ control plane. This allows the CTZ team to send messages to users who are experiencing blocking, along with the flexibility of sending messages tailored to users experiencing specific issues. Server-initiated communication could also allow the CTZ back-end to distribute software updates—a critical tool in times of elevated censorship. Though mobile applications cannot overwrite their compiled software, new tools are being developed to allow circumvention tools to load new protocol implementations at runtime [25, 53]. Moreover, server-side control can also help manage and implement a robust load management and proxy rotation policy, which is not possible with existing transports.

One challenge in using a push-notification-based transport is that it is a one-way communication channel. While this is limiting, the server-to-client direction of this channel is unique compared with other channels in CTZ's control plane. Existing channels, such as email, instant messaging, and domain fronting, already provide

client-to-server communication. However, none of these channels allow communication initiated by the server and directed to the client. Thus, a reliable, unblocked, and indirect channel from server to client fills an essential gap in the context of CTZ's control plane.

Overall, CTZ team places high value for new control plane transports. The unique capabilities that a push-notifications transport brings in had catalyzed the CTZ team to integrate PN transport into their system. Right now, CTZ has already developed client and backend-server libraries and is in the final stages of review and approval for deployment.

6 DISCUSSION

6.1 Near real-time obfuscation modifying protocols

The evolving censorship strategies adopted by the censor have led to the development of various new circumvention technologies. One of the recent designs for circumvention aims to provide an adaptive capability to the circumvention client apps such that they can be updated to change their circumvention strategy on the fly. This design diverges from the traditional methods of building circumvention systems that aim to design a single technique and make it robust toward blocking. However, these techniques become unusable once an adversary finds ways to block the system. The new adaptive designs come in handy in these situations, as once a particular technique is blocked, the client app can be updated to another advanced strategy. In contrast, the traditional method requires recreating and resending a new client app. Two such adaptive approaches aim to facilitate modifying the client app without needing to download a new separate app. The first is Proteus [53], which provides a programmable way of changing the obfuscation methods of the client traffic. The change requires specifying the obfuscation in the language's syntax and sending only a few programmable lines to the client app. The second design is WATER [25], that utilizes WebAssembly to allow for modifications to the client app. Each modification requires sending a small-sized binary to the client that can be processed by the WebAssembly-based client app.

However, one fundamental challenge for such designs to succeed is having a channel to transfer the modification information to the client. While currently these designs assume some existing client-initiated or out-of-band channel to do this, the PN-based channel is the perfect solution for this requirement. With a PN channel, these designs can push updates (either small binaries or protocol specifications) directly to the client app in response to any advance of the censor methodology. We leave the integration of PN to such designs as part of future work.

6.2 Push Notification and Proxy Distribution

We discussed in detail the use of push notifications to distribute bridges and proxies for popular circumvention tools in Section 5.1.1. While we do not focus on proxy distribution strategies, or what proxies should be distributed to which user when, we posit that the server-initiated nature of push notifications as a control channel would augment several recent proposals for proxy distribution.

Lox [50] is a reputation-based proxy distribution system that uses blinded anonymous credentials to limit the rate at which censors discover new proxies and reward users who receive proxies that remain unblocked. While Lox does not store client information at the server, and Lox protocols are interactive and require a two-way channel between the client and server-side components, the table of encrypted proxies and reachability tokens must be distributed to users out-of-band and updated each day.

The Lox Authority (LA) groups available proxies into buckets and maintains a public list of encrypted buckets and their corresponding reachability credentials. Clients download this encrypted proxy list out-of-band and receive a bucket id and decryption key with their credential from the LA. When proxies go offline due to network churn our service outages, the LA swaps out the old proxy information with new, functioning proxies and updates the public encrypted bucket. Each day, the LA creates a new bridge reachability credential for each bucket, using the latest information on whether or not the proxies in that bucket have been blocked by a censor. These reachability credentials are used by clients in Lox's interactive protocols to prove knowledge of unblocked proxies.

It is critical for clients to fetch these new encrypted buckets and reachability credentials as soon as they become available. Proxies that have gone offline should updated quickly to prevent a user's entire bucket from becoming unreachable, and clients that attempt to use out-of-date reachability credentials will fail the interactive Lox protocols. Push notifications can be used to push these updates to users in near real-time, saving several back-and-forth connections to the Lox Authority, and reducing error rates and connection failures. Because these updates apply to all users of Lox equally, they will not break the anonymity provided by the system.

SpotProxy [40] is another recent proposal that uses Spot VMs to create a high-churn proxy pool that evades blocking with a constant influx of new proxy IP addresses. The relocator component of SpotProxy actively migrates clients to new proxies when client proxy assignments change due to induced churn or Spot VM reclamation. When a proxy assignment to a client has changed, the relocator must send the updated proxy connection information to the client over a control channel in the notification phase. Because this update is server-initiated, the original design requires the client to have an active connection to its current proxy. If client's connection is not migrated before the connection fails, the client needs to re-register with the system. Push notifications offer a potential solution for this failure case, allowing the relocator to push new proxy details to the client even if the active connection has failed.

6.3 Collusion Defense

A nation state adversary attempting to restrict PN transport can try to coerce the push notification providers to identify and block notifications. This would require the platform to comply with and block push notification for apps that are used for circumvention. However, a crucial aspect for this approach to succeed is to identify what app to block the push notifications for. It may be straightforward to download the circumvention client app and block the notifications corresponding to it. However, it will be very difficult to block the notifications for apps that are not used for circumvention but provide benign functionalities. For example, if the circumvention app could read push notifications of any benign app, the only option for the adversary would be to then block notification for all apps. While this approach sounds promising to resist collusion based blocking, it requires the circumvention app to have the capability to read the

notifications of all apps on the device. While its possible to do it on a rooted or jailbroken phone, such a requirement would not work for non-technical users and would hinder practicality.

To overcome this, we find that there are a class of services known as accessibility services that can facilitate reading notification of all apps without needing to root the phone. Accessibility services are used for facilitating access for people with disabilities and allow apps to monitor and interact with system events such as notification events. In order to use this service it needs to be declared as a permission in the `AndroidManifest.xml` file with a service component that extends `AccessibilityService` (it specifies the event types the app is allowed to monitor and ensures the app only accesses relevant data like notifications). The new XML file that defines this service component, specifying the exact event type, which in our case is the `TYPE_NOTIFICATION_STATE_CHANGED`. Further, one needs to define the logic for detecting and processing the notifications once a notification event has been generated. Once a notification is detected, one can log the package name (to identify the app sending the notification) and extract notification text content, enabling the app to process this information.

While this approach is beneficial for resisting collusion, it can put the user at some privacy risk as the app can read notification for all apps. We can minimize the privacy concern by applying filters to access only the relevant notifications. However, this approach should strictly be used in severe censorship environments when evidences of collusion are apparent. This is why we do not implement this feature by default but present it as an effective measure against a colluding adversary.

6.4 Platform Censorship

Push notification providers that consciously support censorship circumvention may face pressure from the censor, who can threaten to block or take legal action. The censor may demand the provider to disable push notifications for circumvention apps or reduce the bandwidth by lowering the rate limit, rendering push notification as a transport less useful. A broader adoption of push notifications for circumvention purposes is likely to motivate censors to impose stricter technical and policy controls over such communication channels. However, we note that despite the majority of Google services being blocked in China since 2014, push notifications powered by FCM remain accessible as of November 2024. This observation hints at the economic and societal repercussions that would arise from blocking such a service, possibly creating a backlash that outweighs the benefits of censorship.

7 RELATED WORK

Blocking-resistant communication channels have mostly been explored in the context of mimicry and tunneling-based circumvention systems. Mimicry-based systems such as SkypeMorph [42] and Censorspoofer [54] were aimed at disguising and mimicking censored content to look like standard applications' protocol traffic (such as Skype calls). However, such systems were shown to be easy for the adversary to detect and block due to the inherent difficulty in mimicking all the features of the underlying protocol [35]. Tunneling systems [2, 22, 24, 36, 37, 39, 48] are an improvement over such systems as they do not mimic protocol messages, rather they use the underlying protocol *as-is* to transfer covert content. This ensures that all the features of the underlying protocol (e.g., packet size) remain unaltered, making the job of the adversary much more difficult. Raceboat [52] formalizes the usage of tunneling systems as signaling channels with the help of a conceptual framework.

The channel that has been specifically used for conveying control information in highly censored regions over the past years has been domain fronting [28]. It has been integrated in Psiphon, Lantern, Tor pluggable transports such as Snowflake, and even in tools such as Massbrowser [43]. However, almost all major organizations have stopped support for domain fronting [7] with even the last few remaining ones set to join the list [14]. Another interesting one-way communication channel was proposed in Tapdance [55] and is currently being used in Conjure [30]. The channel steganographically hides information in the TLS payload.

Despite being highly efficient, both of these channels (and other mimicry and tunnelling-based channels) are client-initiated, with the TLS-based channel in Conjure completely bypassing server-to-client communication. However, the push-notification-based channel, as proposed in this work, is the first server-to-client channel that is implemented in popular circumvention tools. Note that push notification in the context of circumvention has been preliminarily discussed [56]. The previous work discussed and presented the feasibility of a full-fledged two-way circumvention channel. However, the one-way characteristic of push notifications makes any full-fledged circumvention tool design impractical, essentially requiring another asynchronous channel to support two-way communication.

8 CONCLUSION

Fueled by the rise of censorship events, users in censored regions increasingly rely on circumvention tools. While it seems impossible to build a system that the censor cannot block, we note that the sustainable way of fighting censorship is to continuously evolve circumvention technology. However, the core challenge in achieving such sustainability is for the tool maintainers to be able to continuously communicate protocol or proxy IP updates to end users.

Thus, in this paper, we explored the use of push notification services to maintain a stable channel of communication between circumvention tool servers and their clients. Our measurements suggest that adversaries are wary of censoring push notification services due to the potential for high collateral damage.

We demonstrate the utility of push notifications by studying various popular circumvention tools and show real-world cases where integrating push notifications is useful. We successfully integrated push notifications for use in Tor to automatically push bridge line updates and are in the advanced stages of integration and deployment in another popular circumvention tool. We believe a server-initiated channel in the form of push notification advances the capabilities of circumvention tools and can significantly help in the ongoing arms race with the censor.

REFERENCES

[1] 2015. Timeline of Tor censorship. https://www1.icsi.berkeley.edu/~sadia/tor_timeline.pdf.
[2] 2018. DNS tunnel over DNS over HTTPS (DoH) or DNS over TLS (DoT) resolvers. https://www.bamsoftware.com/software/dnstt/.
[3] 2018. Google confirms some of its own services are now getting blocked in Russia over the Telegram ban. https://techcrunch.com/2018/04/22/

google-confirms-some-of-its-own-services-are-now-getting-blocked-in-russia-over-the-telegram-ban/.
[4] 2021. Cyberoam firewall blocks meek by TLS signature. https://groups.google.com/forum/#!topic/traffic-obf/BpFSCVgi5rs/.
[5] 2022. Increase of Tor users in Russia. https://metrics.torproject.org/userstats-bridge-combined.html?start=2021-12-01&end=2022-03-10&country=ru.
[6] 2023. 15 Must-Know Web Push Notification Statistics. https://gravitec.net/blog/15-must-know-web-push-notification-statistics/.
[7] 2023. Amazon follows Google to block domain fronting. https://www.bleepingcomputer.com/news/cloud/amazon-follows-google-in-banning-domain-fronting/.
[8] 2023. The Great Push Notifications Benchmark 2024. https://batch.com/ressources/etudes/benchmark-notifications-push-crm-mobile.
[9] 2023. How the Russian Government Silences Wartime Dissent. https://www.nytimes.com/interactive/2023/12/29/world/europe/russia-ukraine-war-censorship.html.
[10] 2023. Increase of Tor users in Iran during Mahsa Amini protests. https://metrics.torproject.org/userstats-bridge-country.html?start=2022-08-01&end=2023-01-02&country=ir.
[11] 2023. Lifespan of a Push Notification message in Google FCM. https://firebase.google.com/docs/cloud-messaging/concept-options#ttl.
[12] 2023. Operating System Market Share Worldwide. https://gs.statcounter.com/os-market-share/.
[13] 2023. Push Notifications Service Market Size, Share, Growth, and Industry Analysis, By Type (Mobile Push, Web Push, In-App Push & Others), By Application (Education, Consumer, Government, Entertainment, News Information & Others), and Regional Insights and Forecast to 2032. https://www.businessresearchinsights.com/market-reports/push-notifications-service-market-116554.
[14] 2024. Fastly to block domain fronting in 2024. https://riskybiznews.substack.com/p/fastly-to-block-domain-fronting-in-2024.
[15] 2024. Getting bridges from Tor. https://tb-manual.torproject.org/bridges/.
[16] 2024. Obtaining bridges from Tor. https://tb-manual.torproject.org/circumvention/.
[17] 2024. Orbot: Proxy with Tor. https://guardianproject.info/apps/org.torproject.android/.
[18] Accessnowcensorship 2023. AccessNow Censorship Archives. https://www.accessnow.org/tag/censorship/.
[19] Alice, Bob, Carol, Jan Beznazwy, and Amir Houmansadr. 2020. How China Detects and Blocks Shadowsocks. In *ACM Internet Measurement Conference (IMC)*.
[20] Apple. 2023. User Notifications. https://developer.apple.com/documentation/usernotifications.
[21] Diogo Barradas, Nuno Santos, and Luís Rodrigues. 2018. Effective detection of multimedia protocol tunneling using machine learning. In *27th USENIX Security Symposium (USENIX Security 18)*.
[22] Diogo Barradas, Nuno Santos, Luís Rodrigues, and Vítor Nunes. 2020. Poking a hole in the wall: Efficient censorship-resistant Internet communications by parasitizing on WebRTC. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*.
[23] Cecylia Bocovich, Arlo Breault, David Fifield, Serene, and Xiaokang Wang. 2024. Snowflake, a censorship circumvention system using temporary WebRTC proxies. In *USENIX Security Symposium*. USENIX. https://www.usenix.org/system/files/sec24fall-prepub-1998-bocovich.pdf
[24] Chad Brubaker, Amir Houmansadr, and Vitaly Shmatikov. 2014. CloudTransport: Using Cloud Storage for Censorship-Resistant Networking.
[25] Erik Chi, Gaukas Wang, J Alex Halderman, Eric Wustrow, and Jack Wampler. 2023. Just add WATER: WebAssembly-based Circumvention Transports. *arXiv preprint arXiv:2312.00163* (2023).
[26] Roger Dingledine, Nick Mathewson, Paul F Syverson, et al. 2004. Tor: The second-generation onion router.. In *USENIX security symposium*.
[27] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. 2015. Examining how the great firewall discovers hidden circumvention servers. In *Proceedings of the 2015 Internet Measurement Conference*.
[28] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. 2015. Blocking-Resistant Communication through Domain Fronting. *Proceedings on Privacy Enhancing Technologies* (2015).
[29] freedomh 2020. Freedom House Report Internet Freedom Status. https://freedomhouse.org/explore-the-map?type=fotn&year=2020.
[30] Sergey Frolov, Jack Wampler, Sze Chuen Tan, J. Alex Halderman, Nikita Borisov, and Eric Wustrow. 2019. Conjure: Summoning Proxies from Unused Address Space. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*.
[31] Sergey Frolov, Jack Wampler, and Eric Wustrow. 2020. Detecting Probe-resistant Proxies.. In *NDSS*.
[32] Genevieve Gebhart and Tadayoshi Kohno. 2017. Internet censorship in Thailand: User practices and potential threats. In *2017 IEEE European symposium on security and privacy (EuroS&P)*. IEEE.
[33] Google. 2023. Firebase Cloud Messaging. https://firebase.google.com/docs/cloud-messaging.
[34] Yang Han, Dawei Xu, Jiaqi Gao, and Liehuang Zhu. 2022. Using Blockchains for Censorship-Resistant Bootstrapping in Anonymity Networks. In *Information and Communications Security*.
[35] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. 2013. The parrot is dead: Observing unobservable network communications. In *2013 IEEE Symposium on Security and Privacy*.
[36] Amir Houmansadr, Thomas J. Riedl, Nikita Borisov, and Andrew C. Singer. 2013. I want my voice to be heard: IP over Voice-over-IP for unobservable censorship circumvention. In *Network and Distributed System Security Symposium*.
[37] Amir Houmansadr, Wenxuan Zhou, Matthew Caesar, and Nikita Borisov. 2017. SWEET: Serving the Web by Exploiting Email Tunnels. *IEEE/ACM Transactions on Networking* (2017).
[38] Freedom House. 2023. Freedom on the net report 2023 by freedom house. https://freedomhouse.org/sites/default/files/2023-10/Freedom-on-the-net-2023-DigitalBooklet.pdf.
[39] Shengtuo Hu, Xiaobo Ma, Muhui Jiang, Xiapu Luo, and Man Ho Au. 2017. Autoflowleaker: Circumventing web censorship through automation services. In *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*.
[40] Patrick Tser Jern Kon, Sina Kamali, Jinyu Pei, Diogo Barradas, Ang Chen, Micah Sherr, and Moti Yung. 2024. SpotProxy: Rediscovering the Cloud for Censorship Circumvention. In *USENIX Security Symposium*. USENIX. https://www.cs-pk.com/sec24-spotproxy-final.pdf
[41] Microsoft. 2023. Windows Push Notification Services (WNS) overview. https://learn.microsoft.com/en-us/windows/apps/design/shell/tiles-and-notifications/windows-push-notification-services--wns--overview.
[42] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. 2012. Skypemorph: Protocol obfuscation for tor bridges. In *Proceedings of the 2012 ACM conference on Computer and communications security*.
[43] Milad Nasr, Hadi Zolfaghari, Amir Houmansadr, and Amirhossein Ghafari. 2020. MassBrowser: Unblocking the Censored Web for the Masses, by the Masses.. In *NDSS*.
[44] obfs4 2023. Learning more about the GFW's active probing system. https://blog.torproject.org/learning-more-about-gfws-active-probing-system.
[45] pushstats 2023. Push Notification Statistics (2023). https://www.businessofapps.com/marketplace/push-notifications/research/push-notifications-statistics/.
[46] Ram Sundara Raman, Prerana Shenoy, Katharina Kohls, and Roya Ensafi. 2020. Censored Planet: An Internet-wide, Longitudinal Censorship Observatory. In *Computer and Communications Security*. https://www.ramakrishnansr.com/assets/censoredplanet.pdf
[47] Reethika Ramesh, Ram Sundara Raman, Matthew Bernhard, Victor Ongkowijaya, Leonid Evdokimov, Anne Edmundson, Steven Sprecher, Muhammad Ikram, and Roya Ensafi. 2020. Decentralized control: A case study of russia. In *Network and Distributed Systems Security (NDSS) Symposium 2020*.
[48] Piyush Kumar Sharma, Devashish Gosain, and Sambuddho Chakravarty. 2021. Camoufler: Accessing The Censored Web By Utilizing Instant Messaging Channels. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*.
[49] The Tor Project. 2023. Moat | Tor Project | Support. https://support.torproject.org/glossary/moat/.
[50] Lindsey Tulloch and Ian Goldberg. 2023. Lox: Protecting the Social Graph in Bridge Distribution. *Proceedings on Privacy Enhancing Technologies* (2023).
[51] Benjamin VanderSloot, Allison McDonald, Will Scott, J Alex Halderman, and Roya Ensafi. 2018. Quack: Scalable Remote Measurement of {Application-Layer} Censorship. In *27th USENIX Security Symposium (USENIX Security 18)*. 187–202.
[52] Paul Vines, Samuel McKay, Jesse Jenter, and Suresh Krishnaswamy. 2024. Communication Breakdown: Modularizing Application Tunneling for Signaling Around Censorship. *Proceedings on Privacy Enhancing Technologies* (2024).
[53] Ryan Wails, Rob Jansen, Aaron Johnson, and Micah Sherr. 2023. Proteus: Programmable protocols for censorship circumvention. *Free and Open Communications on the Internet* (2023).
[54] Qiyan Wang, Xun Gong, Giang TK Nguyen, Amir Houmansadr, and Nikita Borisov. 2012. Censorspoofer: asymmetric communication using ip spoofing for censorship-resistant web browsing. In *Proceedings of the 2012 ACM conference on Computer and communications security*.
[55] Eric Wustrow, Colleen M Swanson, and J Alex Halderman. 2014. TapDance: End-to-Middle Anticensorship without Flow Blocking. In *23rd USENIX Security Symposium (USENIX Security 14)*.
[56] Diwen Xue and Roya Ensafi. 2023. The Use of Push Notification in Censorship Circumvention. *Free and Open Communications on the Internet* (2023).
[57] Tarun Kumar Yadav, Akshat Sinha, Devashish Gosain, Piyush Kumar Sharma, and Sambuddho Chakravarty. 2018. Where the light gets in: Analyzing web censorship mechanisms in india. In *Proceedings of the Internet Measurement Conference 2018*.

FCM endpoints used in Hyperquack measurement
mtalk.google.com
mtalk4.google.com
mtalk-staging.google.com
mtalk-dev.google.com
alt1-mtalk.google.com
alt2-mtalk.google.com
alt3-mtalk.google.com
alt4-mtalk.google.com
alt5-mtalk.google.com
alt6-mtalk.google.com
alt7-mtalk.google.com
alt8-mtalk.google.com

Table 2: FCM Endpoints used in Hyperquack measurement. The set of endpoints were collected from the FCM documentation.

A APPENDIX

The Table 2 lists the FCM push notification service URLs that were checked for reachability in various countries (including the most censored ones) across the globe for a duration of seven months.