Towards a Scalable Censorship-Resistant Overlay Network based on WebRTC Covert Channels

Diogo Barradas, Nuno Santos
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
{diogo.barradas,nuno.m.santos}@tecnico.ulisboa.pt

Abstract

In many regions of the world, nation-states enforce Internet censorship policies that prevent unrestricted access to information and services by their citizens. Over the years, many censorship circumvention tools have been proposed which, however, require either the deployment of a dedicated infrastructure within trusted ISPs, or are vulnerable to state-of-the-art traffic analysis techniques. To fill this gap, we propose to build a practical censorship-circumvention service that exhibits strong resistance against traffic analysis attacks. By relying on a recent proposal for creating covert channels through WebRTC streams, we discuss the design of a distributed system named Censorship-Resistant Overlay Network (CRON). CRON aims at offering to the users located in censored regions a set of services that allow them to locate proxies positioned in the free Internet region, and set up secure covert tunnels for accessing arbitrary sites on the Internet. We present the key challenges and explore the solutions in making CRON robust against state-level attacks.

CCS Concepts: • **Security and privacy → Network security**; • **Social and professional topics → Censorship**.

Keywords: Censorship circumvention; Social networks; Traffic analysis; Trust; WebRTC

ACM Reference Format:
Diogo Barradas, Nuno Santos. 2020. Towards a Scalable Censorship-Resistant Overlay Network based on WebRTC Covert Channels. In *1st International Workshop on Distributed Infrastructure for Common Good (DICG'20), December 7–11, 2020, Delft,Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3428662.3428788

1 Introduction

Many states worldwide implement strict censorship policies that prevent their citizens from freely accessing information on the Internet [21]. In countries such as China, Russia, or Iran, this level of control can be enforced by the use of network-level censorship techniques such as URL keyword filtering [10], the blocking of Internet destinations [20] or by thwarting selected networking protocols, e.g., Tor [12]. To overcome such barriers and give Internet users free access to blocked content, it is important to devise censorship-resistant communication tools that not only can effectively circumvent state-of-the-art Internet content blocking techniques but also prevent a state-level adversary from detecting the usage of such tools and put potential users under high risk of reprisal.

However, despite the numerous Internet censorship circumvention systems proposed over the years [17], many systems have struggled to satisfy both these conditions for two main reasons. On the one hand, an existing class of systems requires the deployment of dedicated proxy / routing infrastructures by third-party ISPs or other trusted organizations which constitutes a significant hurdle to the widespread adoption of these tools [22]. On the other hand, many existing systems have been found to be vulnerable to DPI [16] and sophisticated machine learning (ML)-based traffic analysis attacks [3]. Such systems rely on some form of covert channel to stealthily transmit sensitive data through an apparently innocuous carrier medium, e.g., the encrypted media streams of videoconferencing applications such as Skype [2, 18].

This paper presents a preliminary design of Censorship-Resistant Overlay Network (CRON), a distributed system that aims at allowing common Internet users to bypass network-level censorship while i) precluding the need to depend on a dedicated ISP infrastructure, and ii) providing strong resistance against traffic analysis attacks. CRON's central idea is to leverage an emerging ecosystem of WebRTC-enabled streaming applications as a carrier infrastructure. WebRTC [23] is a W3C standardization initiative for protocols and APIs that enables secure real-time communication between browsers. Currently supported by all major browsers, WebRTC is used by many popular video conferencing services for making real-time video calls, e.g., Whereby (`https://whereby.com`). Our idea is then to use network covert channels established over encrypted WebRTC streams, generated during typical video calls, to tunnel arbitrary payload traffic to the open Internet.

To achieve this goal, CRON will leverage a recently proposed system named Protozoa [4] which allows the creation of secure point-to-point covert channels over WebRTC streams. Using Protozoa, a user located in a censored region will be

DICG'20, December 7–11, 2020, Delft,Netherlands

ACM ISBN 978-1-4503-8197-0/20/12.
https://doi.org/10.1145/3428662.3428788

able to call a second (trusted) user placed in the free Internet region – a proxy – and exchange covert IP traffic with an arbitrary Internet destination while the encrypted video call takes place. Protozoa is able to deliver high performance and provide strong resistance to traffic analysis. However, the scalability of this system is currently impaired by the absence of mechanisms for finding trusted peers to engage in circumvention. Specifically, an isolated user that does not have a direct acquaintance outside a censored region will be unable to leverage Protozoa for accessing blocked Internet content. In addition, Protozoa cannot deal with a range of state-level attacks, such as the deployment of Sybil (fake) users, the usage of WebRTC streaming applications controlled by an adversary, or long-term user profiling attempts which may tip off an adversary about unusual WebRTC connection patterns.

CRON introduces a new system design which, in addition to overcoming these limitations, lays the groundwork for building a whole range of distributed censorship-resistant applications. First, CRON will allow isolated users to reach out to remote proxies through a multi-hop chain of trusted nodes through the establishment of end-to-end covert channels (named *circuits*) for accessing arbitrary Internet destinations. In this paper, we characterize the main threats that can be launched by a state-level adversary, and propose defenses to be developed in the future. Second, CRON will provide a common API to allow the development of extensions and third-party applications that may benefit from the underlying covert channel circuitry, for instance for the delivery of delay-tolerant content (e.g., in a CDN-like fashion) or the deployment of a distributed censorship-resistant filesystem.

2 Background on WebRTC Covert Tunnels

The general operation of Protozoa [4] can be grasped with the help of Figure 1 which shows two Internet users. One of them (the *client*) is located in a censored region where the access to certain content or services (e.g., Youtube) is blacklisted by a state-level censor who can inspect and control all the network communications within its perimeter of influence. The client intends to overcome these restrictions and access blocked content without the censor's awareness. This will be achieved with the help of a trusted user (e.g., a family member) located in the free Internet region who will act as a *proxy*.

To this end, Protozoa creates a high-performance covert channel (≈1.4Mbps) between the client and the proxy such that the former is able to transparently tunnel through all the IP traffic generated between local networked applications (e.g., a web browser) and remote Internet destinations (e.g., Youtube). To create such a covert tunnel, the two users must establish a video call using a WebRTC-enabled streaming website, such as Whereby (`https://whereby.com`). As it is common in such services, one of the users must first create a chatroom, obtain a corresponding URL identifier, and share that URL with the other user via some out-of-band medium

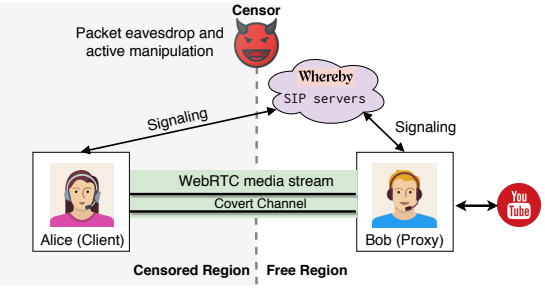

Figure 1. WebRTC covert tunnel over a Whereby call.

(e.g., SMS or email). Upon receiving that URL, the invited user can then join the chatroom and a video call is initiated. At this point, the WebRTC stack implementation of each of the users' browsers engages with the Whereby servers into the execution of WebRTC-specific signalling protocols that guarantee the authentication of both communicating peers and the integrity and confidentiality of the ensuing peer-to-peer encrypted video transmission between them.

To create a covert channel between client and proxy, Protozoa instruments the browser's WebRTC stack and replaces the encoded frames pertaining to the carrier video with information of the covert payload. This replacement is performed such that the size of the encrypted RTP (SRTP) multimedia packets sent to the network retain their original size and timing properties, making Protozoa flows hardly distinguishable from unmodified WebRTC streams using existing ML-based traffic classifiers [3, 4]. Moreover, since all unencrypted packet fields remain intact, Protozoa tunnels cannot be detected through DPI. By acting at the WebRTC layer, Protozoa can leverage any WebRTC-powered application to transmit covert data without additional supporting infrastructure. To block Protozoa traffic, a censor could ultimately block all WebRTC traffic within the censored region. However, this is highly discouraged as it would bring about extensive collateral damage due to ever-greater dependence on video streaming applications.

Despite its considerable advances, Protozoa is only the first step towards obtaining a scalable platform for enabling widespread censorship-resistant communication. In fact, Protozoa assumes a limited threat model which may hamper the scalability of the system to a vast range of users and does not defend against other sophisticated state-level attacks beyond the ML-based analysis of individual flows. First, presupposing that a user knows some contact outside the censored region may be unrealistic for many users, and Protozoa does not facilitate the discovery of trusted peers. Second, Protozoa's covert channels are prone to be compromised in cases where an adversary controls the carrier WebRTC application and is able to decrypt and inspect media traffic. Third, an adversary may eavesdrop and profile users during large periods of time and later pinpoint WebRTC interactions that deviate from this profile. Next, we present a system design aimed at overcoming these limitations and offering additional services.

3 The CRON System Model

In the interest of enabling a vast amount of users located within censored regions to access arbitrary Internet content, our goal is then to build a Censorship-Resistant Overlay Network (CRON). As shown in Figure 2, we conceptualize CRON as a distributed system of Internet nodes that allows the creation of (possibly) multi-hop covert channels – named *covert circuits* – enabling clients located within a censored region to access the free Internet through proxies deployed outside the censored region. Nodes can exchange covert data within small trusted social groups by leveraging the videoconferencing services of WebRTC-enabled applications. CRON will provide the necessary support for helping clients to discover available proxies based on a chain of trust, set up covert circuits, and sustain the transmission of covert data for different workloads. Next, we present a system model decoupling all aspects involving the formation of social groups from network-level connectivity and data transmission issues.

3.1 Chatrooms and Social Circles

We envision the CRON overlay network to be organically formed on top of a communication abstraction commonly offered by WebRTC-enabled streaming services, namely the notion of *chatroom*. A chatroom allows a group of users to stream real-time video from their local camera devices to one another. A chatroom is explicitly created on a given web streaming service by a user who can then invite other users to join the chatroom by sharing a unique URL generated by the streaming service. This URL serves both as a chatroom identifier and as an access control token. All users that accept the invitation become chatroom members and have the ability to send / receive video feeds to / from other members.

Typically, users share invitations within their circle of contacts or relationships (personal, professional, or others). Considered globally, this network of relationships forms a *social graph*, where each node represents a user and the edges the relationships between users. It expresses the real-world network of social contacts that can be partially or fully supported by different systems, e.g., email, mobile apps such as Whatsapp, OSNs, etc. An edge exists between u_1 and u_2 if u_1 has the contact of u_2 on *some* particular system (e.g., email). We use the term *social circle* of a given user (u_1) to identify all the direct contacts known to the user, i.e., all nodes u_2, that have a unidirectional edge from u_1. Social circles will provide the foundations for building trust amongst CRON nodes in the establishment of covert circuits (see in Section 4.3).

3.2 Covert Circuits and Data Transmission

This section describes the generation of communication channels in CRON, which leverages the chatroom functionality presented above to build *covert circuits*. Each CRON user will locally run a piece of software that turns its local computer into a *node* of the CRON overlay network (Figure 2).

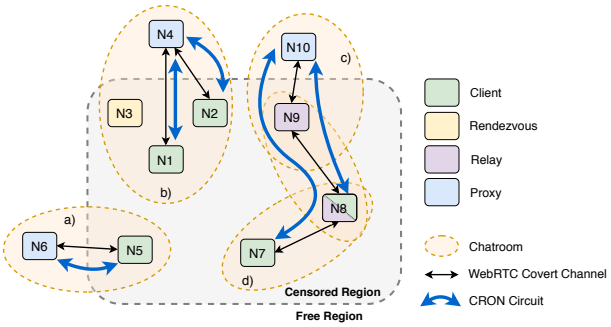

Figure 2. CRON's system model. Nodes participating in the same chatroom are assumed to be part of the same social circle.

If a client and a proxy are members of the same chatroom, they both can exchange covert content through the WebRTC streams operated by the chatroom's web streaming application. In this case, we say that a *direct covert circuit* can be established between the client and the proxy (Figure 2a).

If there is no potential proxy that a client can choose from within its social circle (e.g., someone located outside the censored region), a covert tunnel can only be established with the help of one (or multiple) *mediator node*(s). A mediator node can simply play the role of *rendezvous*, i.e., one that is part of the social circles of client and proxy alike, and can put them in contact by inviting them both to a common chatroom. Once they all join that chatroom, a direct covert circuit can be established between client and proxy (Figure 2b).

Alternatively, a mediator node can play the role of a *relay*. It bridges the communication between client and proxy by joining two different chatrooms – one shared with the client, and another with the proxy – and by relaying covert traffic across them (Figure 2c). Client and proxy will then be connected by a *1-hop covert circuit*. In *N-hop covert circuits*, client and proxy are bridged by N relays, chained together by a sequence of *segments* which consist of pair-wise WebRTC connections between client-relay, relay-relay, and relay-proxy. Each segment reflects a real-time media streaming transmission held between two parties in an independent chatroom which, in addition, may have been moderated by a specific rendezvous node. In a direct covert circuit, N=0, and there is a single segment connecting client and proxy.

We envision two models of covert data transmission. If all N relays of a covert circuit are simultaneously available and the corresponding N+1 segments are operational, then it is possible to synchronously tunnel *real-time* traffic between client and proxy. Otherwise, if at least one segment of the circuit is disconnected and cannot transmit covert data, the transmission must be *staged*, i.e., the relays must temporarily store pending data requests and responses on a local cache, and defer the data delivery until the connectivity is reestablished (Figure 2d). Thus, while real-time mode allows for the transmission of interactive traffic, the staged mode is suited only for delay-tolerant workloads, e.g., CDN applications, distributed file systems, or key-value stores.

4 Thwarting State-Level Attacks

In this section, we discuss important attacks that can be launched by a state-level adversary to detect covert traffic and identify the intervening parties, despite the fact that Protozoa point-to-point covert channels are secure. We describe such attacks by increasing order of mitigation difficulty and discuss our ideas for making CRON robust against them.

4.1 Analysis of WebRTC Connection Patterns

In addition to analyzing pairwise WebRTC streams, an adversary may compute alternative statistics that enable it to profile the regular traffic patterns of users over a long period of time and pinpoint potential CRON users that display abnormal WebRTC connection patterns. Such long-term statistical attacks were already applied in other domains, for instance for detecting users' actions in anonymity networks [9].

We identify three types of WebRTC statistics (Sx) through user profiling to pinpoint abnormal patterns in the generic usage of WebRTC, regardless of how legitimate the traffic characteristics of a single flow look like: (S1) existence of simultaneous video calls, (S2) existence of uncommon call parties, and (S3) existence of uncommon call times, frequency, and duration. As for S1, nodes connected simultaneously to multiple chatrooms (i.e., relays) are prone to be easily spotted since users do not typically conduct more than one video-call simultaneously. Regarding S2, nodes that suddenly start placing a large amount of WebRTC video calls to nodes which have so far seen little connection to may be seen as abnormal to an adversary. Lastly, S3 may reveal atypical WebRTC video calls when compared with a user's past behaviour, e.g., during times of the day where they do not usually make video calls.

To cope with user profiling threats, we propose two alternative models of functioning for CRON: *passive*, and *active*.

Passive mode: This is the most risk-averse strategy, where CRON provisions covert circuits exclusively without modifying any of the statistics S1, S2, or S3. To achieve this, it will monitor in background the usual video calls made by a user, and identify windows of opportunity for the establishment of a covert circuit. Such a window occurs if, in the course of an ongoing video call, there is at least another participant in that same chatroom running a CRON node. If such a participant runs a relay or a proxy node, there is the possibility for the local CRON node to run as a client and connect with the former via a direct covert circuit. The local user can then be given the opportunity to tunnel real-time payload traffic while the video call takes place. Similar occasions can also be leveraged for fetching delay-tolerant traffic from an immediately connected relay or proxy. Hence, CRON's passive mode allows users to stealthily access sensitive content (with the cooperation of a trusted party outside the censored region) while engaging in their routinely professional or personal virtual encounters.

Active mode: In addition to exploiting the passive mode's connectivity chances, a local node will have some room for making new or longer WebRTC video calls with the purpose of increasing the opportunities to exchange covert traffic. Since users' social interactions are rather dynamic [5], the S1 and S2 statistical patterns of a typical user are characterized not by a set of fixed-value features but by a range of plausible values. In CRON's active mode, we want to take advantage of this variability so as to increase the connectivity opportunities of a node while keeping the pattern deviations in S1 and S2 below some acceptable threshold. This can be achieved, for instance, by placing calls to different rendezvous contacts or introducing bounded variability in call times, frequency, and duration. Our approach is marginally inspired by existing obfuscation techniques based in the addition of noise to thwart the generation of accurate user profiles [7]. Since the establishment of multiple simultaneous WebRTC video calls is highly atypical in "normal" user profiles, in CRON, the establishment of N-hop covert circuits ($N \geq 1$) will serve the purpose of transmission of delay-tolerant traffic only.

4.2 Control of Carrier WebRTC Services

Internet users of certain censored regions may generally be constrained to using videoconferencing services controlled by the state. In such cases, a state-level adversary is able to mount two new attacks. First, by controlling the WebRTC signalling phase, it can hijack the identity of legitimate CRON nodes and launch MITM attacks. As such, it can detect if any of the interlocutors request or deliver covert data to their counterpart. This detection would be straightforward since CRON's vanilla covert data encoding technique (see Section 2) fully replaces the video payload with an apparently random covert data signal that results in a scrambled video image at the receiver's endpoint. Second, by controlling the data transmission, an adversary is able to decrypt SRTP packets' payload and decode the enclosed video frames, e.g., by leveraging WebRTC gateways to record, inspect, or re-encode media on-the-fly [1]. So here, by replaying the video, it is possible to obtain clear evidence of video manipulation.

To counter such attacks, we can devise an additional type of covert circuits named *stego* circuits. The main difference between stego circuits and *regular* (i.e., vanilla) ones is the incorporation of two security mechanisms. First, instead of replacing the encoded video frames with covert data, stego circuits will *embed* the covert data into the encoded video frames in such a way as to maintain the visual characteristics of the video feed transmitted by each endpoint. To this end, we can employ video steganography algorithms that operate at the compressed data domain [26]. Second, to further prevent MITM attacks, the steganographic decoding algorithm can be protected with a secret key. By securing this key with standard public-key encryption and exchanging the public keys out-of-band, CRON nodes can be properly authenticated when setting up stego circuits. Assessing the performance penalties of stego circuits will require further experiments.

4.3 Deployment of Sybil Nodes

It is expected that, in a real-world setting, a state-level adversary will likely attempt to infiltrate state-controlled agents into the network who can offer fake proxying or relaying services, or issue fake client requests in an attempt to track down legitimate CRON nodes. To make matters worse, the adversary may actively coerce honest users of the CRON overlay to act on the adversary's behalf. Thus, the CRON system must provide defence mechanisms against Sybil nodes.

While multiple systems have tackled the problem of trust and reputation in P2P networks [15], CRON's usage scenario demands special consideration. In fact, existing automated Sybil detection approaches are generally only able to wield probabilistic guarantees on the possibility of contacting / timely evicting of Sybil nodes from the network. In our context, however, minimizing the amount of node-specific information propagated across the network is highly commendable to reduce the exposure of legitimate nodes [8].

For these reasons, to decrease the chances of interaction between honest CRON users and Sybil nodes, we substitute an automatic reputation scoring system with a discretionary system where the ultimate choice to connect to a particular node is left to CRON users. In a nutshell, CRON's trust establishment system is centred around each user (u) and it is organized into two *rings of trust*. The inner and most critical ring of trust corresponds to *1st degree trustees*. These consist of the users that belong to u's social circle (i.e., the nodes in the social graph that u is immediately connected to). For each of these users, u can essentially tell the CRON system whether a particular user can be deemed to be trusted when acting in each of several possible roles: *client*, *rendezvous*, *relay*, or *proxy*. CRON will ensure that any given covert circuit will only be created if all involved nodes are mutually trusted. An additional outer ring corresponds to *2nd degree trustees*, i.e., "the friends of a friend". This allows u to delegate to individual 1st degree trustees the possibility that some of its "friends" can collaborate in the creation of N-hop circuits with u. This two-ringed trust system constitutes our working hypothesis which we believe strikes a good balance between security, and complexity, and flexibility. Nevertheless, our trust system requires further validation, e.g., through the study of existing trust models and usability assessments.

5 Envisioned CRON Architecture

To build CRON in such a way as to incorporate all the ideas presented above, we propose a preliminary architecture depicted in Figure 3. A CRON node consists of a server built on top a browser runtime engine featuring a modified WebRTC implementation stack. Through a *user interface* installed as an extension of the browser runtime, the local user is able to configure the system, e.g., set it up to run in different roles (i.e., client, proxy, etc.), assign a level of trust to other users, and manage covert circuit sessions.

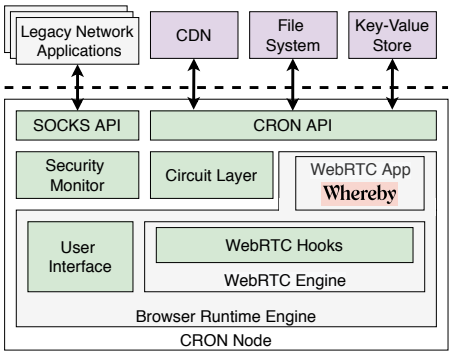

Figure 3. CRON software stack.

The covert channel services are internally structured in a layered fashion. From bottom-up, a set of *WebRTC hooks* intercepts the WebRTC streams generated by a guest client-side web streaming application, and creates point-to-point covert channels accordingly. This basic functionality is leveraged by the *circuit layer* which sits on top of the browser runtime engine. This layer deals with all major challenges involving the establishment of end-to-end N-hop covert circuits, the mitigation of user profiling attacks (i.e., passive and active mode support), and the management of regular and stego covert circuits. The circuit layer exposes two external interfaces: a SOCKS proxy interface for tunnelling local IP traffic, and an API for building custom CRON applications, such as CDNs, distributed file systems, or key-value stores.

The *security monitor* performs all major security checks upon the establishment of CRON circuits. For instance, it checks that all the nodes that participate in the circuit are trusted by the user. It also enforces some safety conditions, in particular, that proxies are not located in the censored region. To this end, we plan to explore recent techniques which generate location proofs for proxy servers [25]. Additionally, when the local node runs as a proxy or as a relay, the security monitor provides support for whitelisting policies so as to block all potential clients' requests to access illegal content that would make proxy operators liable to prosecution in their own countries (e.g., accessing to child pornography).

As for the next steps, we aim to implement a prototype of CRON as a middleware component for major web browsers (e.g., Firefox), while adhering to the existing WebRTC standards [23]. This component will expose an API that can be used for building generic distributed applications on top of the CRON censorship-resistant overlay network. For evaluating our prototype with regard to long-term user profiling (Section 4.1), the traffic-analysis resistance of circuits (Section 4.2), and the resistance to Sybil attacks (Section 4.3), we plan to collect and analyse network data produced within our university's campus network. To assess whether these properties can indeed be helpful in pinpointing endpoints engaged in circumvention will be a significant contribution of our future work. We intend to release the produced datasets to the community after proper anonymization.

6 Related Work

We can find in the literature a large body of work with the goal of evading state-level censors [17]. In contrast to CRON, refraction networking requires the deployment of tailored routers in the network paths between clients and overt destinations [22]; these routers, managed by cooperating ISPs, redirect client traffic to blocked destinations.

A different class of techniques that does not require a dedicated infrastructure holds in the general idea of obfuscating covert traffic. Randomization tools like Obfsproxy [11] transform covert traffic into random bytes to evade protocol blacklists, while protocol imitation systems aim at mimicking protocols allowed across a censor's borders [18]. Unfortunately, imitation systems are prone to multiple network attacks [16].

Other approaches tunnel covert data through a protocol's application layer in order to better cope with the intricacies of the carrier protocol [2, 14]. Cache browsing allows the access to censored content cached in CDNs' edge servers [27]. MassBrowser [19] leverages cache browsing to split covert traffic amongst volunteer proxies. However, many of these systems have been found to be vulnerable to traffic analysis [3, 24] and ineffective in the presence of protocol whitelisting [11, 19].

Lastly, works complementary to ours redirect unobfuscated covert traffic through short-lived proxies [13], or invalidate the state of connections tracked by censors' firewalls [6].

7 Conclusions

This paper presented CRON, a distributed system of nodes interlinked by WebRTC video streaming channels. These nodes will be able to tunnel covert traffic through ongoing WebRTC streams. Some nodes are located within the censored region (clients) and others in the free region (proxies). Acting globally and in a coordinated fashion, the nodes will assist each other in providing uncensored access to Internet resources.

Acknowledgments: We thank the anonymous reviewers for their insightful comments. This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) via the SFRH/BD/136967/2018 grant, and the UIDB/50021/2020 projects.

References

[1] Alessandro Amirante, Tobia Castaldi, Lorenzo Miniero, and Simon Pietro Romano. 2015. Performance analysis of the Janus WebRTC gateway. In *Proc. of AWes*.

[2] Diogo Barradas, Nuno Santos, and Luís Rodrigues. 2017. DeltaShaper: Enabling Unobservable Censorship-resistant TCP Tunneling over Videoconferencing Streams. In *Proc. of PETS*.

[3] Diogo Barradas, Nuno Santos, and Luís Rodrigues. 2018. Effective Detection of Multimedia Protocol Tunneling using Machine Learning. In *Proc. of USENIX Security*.

[4] 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 *Proc. of CCS*.

[5] Fabrício Benevenuto, Tiago Rodrigues, Meeyoung Cha, and Virgílio Almeida. 2009. Characterizing user behavior in online social networks. In *Proc. of SIGCOMM*.

[6] Kevin Bock, George Hughey, Xiao Qiang, and Dave Levin. 2019. Geneva: Evolving Censorship Evasion Strategies. In *Proc. of CCS*.

[7] Richard Chow and Philippe Golle. 2009. Faking contextual data for fun, profit, and privacy. In *Proc. of WPES*.

[8] Ian Clarke, Oskar Sandberg, Matthew Toseland, and Vilhelm Verendel. 2010. Private communication through a network of trusted connections: The dark freenet. *Network*.

[9] George Danezis and Andrei Serjantov. 2004. Statistical disclosure or intersection attacks on anonymity systems. In *International Workshop on Information Hiding*. Springer.

[10] A. Darer, O. Farnan, and J. Wright. 2017. FilteredWeb: A framework for the automated search-based discovery of blocked URLs. In *Proc of. TMA*.

[11] Roger Dingledine. 2012. Obfsproxy: the next step in the censorship arms race. https://blog.torproject.org/blog/obfsproxy-next-step-censorship-arms-race. Accessed: 2020-09-21.

[12] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. 2015. Examining How the Great Firewall Discovers Hidden Circumvention Servers. In *Proc. of IMC*.

[13] David Fifield, Nate Hardison, Jonathan Ellithorpe, Emily Stark, Dan Boneh, Roger Dingledine, and Phil Porras. 2012. Evading Censorship with Browser-based Proxies. In *Proc. of PETS*.

[14] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. 2015. Blocking-resistant Communication through Domain Fronting. In *Proc. of PETS*.

[15] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. 2009. A survey of attack and defense techniques for reputation systems. *ACM CSUR*.

[16] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. 2013. The Parrot Is Dead: Observing Unobservable Network Communications. In *Proc. of S&P*.

[17] Sheharbano Khattak, Tariq Elahi, Laurent Simon, Colleen M Swanson, Steven J Murdoch, and Ian Goldberg. 2016. SoK: Making Sense of Censorship Resistance Systems. In *Proc. of PETS*.

[18] Hooman Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. 2012. SkypeMorph: Protocol Obfuscation for Tor Bridges. In *Proc. of CCS*.

[19] Milad Nasr, Hadi Zolfaghari, and Amir Houmansadr. 2020. MassBrowser: Unblocking the Censored Web for the Masses, by the Masses. In *Proc. of NDSS*.

[20] Arian Akhavan Niaki, Shinyoung Cho, Zachary Weinberg, Nguyen Phong Hoang, Abbas Razaghpanah, Nicolas Christin, and Phillipa Gill. 2020. ICLab: A Global, Longitudinal Internet Censorship Measurement Platform. In *Proc. of S&P*.

[21] Rafal Rohozinski Ronald Deibert, John Palfrey and Jonathan Zittrain (Eds.). 2010. *Access Controlled: The Shaping of Power, Rights, and Rule in Cyberspace*. MIT Press.

[22] Benjamin VanderSloot, Sergey Frolov, Jack Wampler, Sze Chuen Tan, Irv Simpson, Michalis Kallitsis, J Alex Halderman, Nikita Borisov, and Eric Wustrow. 2020. Running Refraction Networking for Real. *Proc. of PETS* (2020).

[23] W3C. 2020. WebRTC 1.0. https://www.w3.org/TR/webrtc/. Accessed: 2020-09-21.

[24] Liang Wang, Kevin P. Dyer, Aditya Akella, Thomas Ristenpart, and Thomas Shrimpton. 2015. Seeing Through Network-Protocol Obfuscation. In *Proc. of CCS*.

[25] Zachary Weinberg, Shinyoung Cho, Nicolas Christin, Vyas Sekar, and Phillipa Gill. 2018. How to catch when proxies lie: Verifying the physical locations of network proxies with active geolocation. In *Proc. of IMC*.

[26] Pei Xie, Hong Zhang, Weike You, Xianfeng Zhao, Jianchang Yu, and Yi Ma. 2019. Adaptive VP8 Steganography Based on Deblocking Filtering. In *Proc. of IH&MMSec*.

[27] Hadi Zolfaghari and Amir Houmansadr. 2016. Practical censorship evasion leveraging content delivery networks. In *Proc. of CCS*.