On the unobservability of multimedia-based covert channels for Internet censorship circumvention

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

***Abstract*—Several tools support the establishment of covert channels in the Internet by encoding data in the application layer of data exchange protocols. Depending on the techniques used to perform this embedding, and on the amount of information to embed, the procedure may generate network flows that differ, in subtle ways, from the regular flows that do not carry covert channels; these differences may render covert channels prone to detection. Hence, the quality of the tools used to establish covert channels can be assessed by the measure in which they are able evade detection, i.e., how difficult it is to detect their covert channels using traffic analysis techniques. For instance, the effective detection of covert channels may put the life of journalists at risk when they rely on these tools to exercise freedom of speech within countries ruled by oppressive regimes. Here, we deliver an overview of our recent research in the unobservability of multimedia-based network covert channels. In particular, we dwell on the extent to which covert channels can evade screening in face of adversaries with access to traffic analysis capabilities. In this context, we start by describing our past attempts at creating a traffic analysis-resistant covert channel. Then, we detail our efforts in evaluating the effectiveness of state-of-the-art machine learning techniques in the detection of contemporary tools which generate covert channels, and discuss novel techniques to scale such detection capabilities. Lastly, we discuss the design of an improved scheme for the creation of multimedia-based covert channels.**

I. Introduction

The primary goal of network covert channels is to conceal the existence of selected information flows between any two communicating parties from an adversary capable of monitoring traffic flows [1]. Due to the wide variety and increasing bandwidth of existing application layer protocols, these have become a prime target for the deployment of covert channels. Lately, systems able to generate network covert channels leverage the widespread use of encryption for developing data hiding mechanisms. A state-of-the-art approach for the design of such systems, named *protocol tunneling*, embeds concealed data directly into the application layer of multiple encrypted protocols [2]–[4]. In short, protocol tunneling systems take advantage of two fundamental properties for hindering an adversary's efforts aimed at unveiling the presence covert channels: a) they prevent an adversary from inspecting the contents of transmissions in plaintext; b) they force an adversary to distinguish between a protocol's legitimate and covert executions through the analysis of traffic patterns alone.

Albeit adversaries are prevented from directly inspecting the content of encrypted transmissions, they may still be able to detect subtle differences in the packet traces of a protocol when it is used for carrying covert data. Such differences can be uncovered using strictly passive methods (e.g., by observing the length or inter-arrival delay of transmitted network packets) or by analyzing the protocol's behavior in response to active network manipulations (e.g., the loss or reorder of packets) [5]. Thus, an important property that all protocol tunneling systems strive to achieve is *unobservability*. A covert channel is deemed unobservable if an adversary that is capable of scanning any number of streams is unable to distinguish those that carry a covert channel from those that do not [5]. In practice, systems that provide a high degree of unobservability can prevent an adversary from flagging a large fraction of covert flows unless they risk to erroneously flag a large amount of regular traffic as covert flows.

Regardless of its benefits, the existence of unobservable covert channels can be faced as a double edged-sword. On the one hand, such channels may be the only way for citizens living within countries ruled by totalitarian states to communicate their ideas freely. On the other hand, there is the possibility that unobservable covert channels can be leveraged by criminals to transmit data while evading the network monitoring capabilities of law-enforcement. If covert channels which are deemed unobservable can actually be detected by an adversary (e.g., a censor), it is likely that citizens trusting the unobservability properties of these channels will end up being prosecuted. Conversely, if covert channels can be designed in such a way that their accurate detection is intractable, it is possible that the actions of criminals will remain unanswered. Studying and evaluating the limits of the unobservability of covert channels is a subject of important practical implications which has not been thoroughly addressed in the past.

Here, we make an overview of our research in the design and analysis of unobservable covert channels for the Internet. We start by describing the limitations of early tools that use multimedia channels to deploy network covert channels. Then, we introduce the design of DeltaShaper [6], a covert channel tool we have developed in an attempt to tackle limitations of previous systems. Our experience in developing DeltaShaper made us revisit the methodology used to assess the resistance of state-of-the-art covert channels against traffic analysis: we report a summary of our findings published in [7], that shows that past evaluation methods were flawed. Then, we briefly summarize our current research work, namely on techniques to detect covert channels efficiently and our effort to build a novel network covert channel tool which does not suffer from

the design limitations of previous systems. Lastly, we discuss some directions for future work.

II. Unobservability and Censorship Resistance

Over the last years, a number of authors have developed censorship-resistant systems which aim to provide access to blocked information over the Internet [8]. A common approach to achieve this goal is to stealthily *tunnel* covert data over multimedia protocols that are unlikely to be blocked by a censor [4], [9], such as Skype. The advent of such systems fostered our interested in performing a closer analysis of their security properties and to study the emerging issues concerning their practical deployment.

A. Limitations of early protocol tunneling systems

The first system to explore the idea of tunneling covert data through the application data of multimedia protocols was FreeWave [4]. The system allowed users to modulate covert data into acoustic signals of VoIP connections over Skype. Since Skype's traffic is encrypted, a state-level censor that controls the network is prevented from inspecting the content of packets and look for covert data. Aside from this advantage, one of FreeWave's significant strengths was that of enabling the transmission of covert data as full-duplex TCP/IP streams.

FreeWave lacks unobservability. Unfortunately, FreeWave was found to be vulnerable to traffic analysis techniques [5]. In particular, the modified Skype streams produced by FreeWave exhibit patterns that can be detected by a censor that places a tap on the network (e.g., changes in packet size distributions when compared to the transmission of actual speech data). By comparing such patterns against those of regular Skype calls, a censor can identify suspicious Skype streams with high probability and proceed with dropping down such connections, identifying and prosecuting endpoints based on their IPs, or carry out other censorship measures. FreeWave was designed without defense mechanisms that can thwart such attacks. Given that most electronic communication over the Internet can be controlled today by governments and/or by a few corporations [10], the lack of such mechanisms renders FreeWave connections *observable*.

Tunneling approaches improve unobservability. To mitigate traffic analysis attacks, later censorship-resistant systems attempted to ensure that multimedia traffic embedding covert data did not display individuating patterns that could enable a censor to distinguish it from regular traffic. Notably, Facet [11] allows clients to watch arbitrary videos by replacing the audio and video feeds of Skype videocalls. For approximating the traffic patterns of regular videocalls, Facet re-samples the audio frequency and overlays the desired video in a fraction of each frame while the remaining frame area is filled up by a video resembling a legitimate videocall. CovertCast [12] is another relevant system that uses video streaming as message carrier. CovertCast scrapes and modulates the content of web pages into colored matrix images which are distributed via live-streaming platforms such as YouTube. Colored matrix images are parameterized by a cell size (adjacent pixels with a given color), the number of bits encoded in each cell (represented with a color), and the colored matrix framerate.

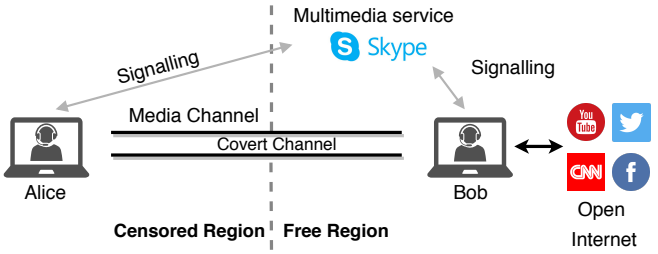


Figure 1: Overview of DeltaShaper communication model.

Facet and CovertCast lack coverage. The main limitation of both Facet and CovertCast, however, is that they do not support the transmission of covert TCP/IP streams. In these systems, the format of covert messages are restricted to certain data types, namely videos (in Facet) or web content (in CovertCast). Moreover, both systems are designed to enable the unidirectional consumption of content. Operations as simple as sending an email are not possible under these systems. Thus, the lack of support for covert TCP/IP streams can considerably impair the *coverage* of such systems, i.e. the content, services and protocols they can access in the Internet.

B. Combining and improving protocol tunneling capabilities

Our first attempt to devise a system that could provide both unobservability and a higher coverage than early works resulted in the design, implementation, and evaluation of DeltaShaper [6], a new censorship resistance system that suports TCP/IP tunneling over videoconferencing Skype streams.

DeltaShaper's communication model overview. A high level overview of the system operation is depicted in Figure 1. In DeltaShaper, the covert TCP/IP packets are encoded and embedded into the video stream transmitted by Skype between the communication endpoints. Similarly to FreeWave, DeltaShaper is meant for scenarios where a user linked to a censoring ISP wishes to establish a covert TCP/IP connection with a remote computer located outside the censoring region. After installing both Skype and DeltaShaper clients, the user can set up a Skype videoconferencing call to a proxy computer and open a DeltaShaper TCP tunnel over that call. The proxy must be configured to run Skype and DeltaShaper clients, be located outside the censoring region, and be maintained by some trusted third party (e.g., a family member or friend). Once the tunnel has been established, the user can run unmodified networked applications such as a mail client, a web browser, or a file transfer agent to access remote services.

Design and evaluation. Our system tunnels TCP/IP traffic by modulating it into colored matrices which are transmitted through a bi-directional Skype videocall. A colored matrix is overlayed in a fraction of the call screen, on top of a typical chat video running in the background. This overlay is

parameterized according to the observed network conditions to increase the unobservability properties of the covert channel.

We have implemented a prototype of DeltaShaper which offers a data-link interface to enable the support of applications running over TCP/IP. Our results show that it was possible to achieve a throughput of 2.56 Kbps with limited impact on the traffic characteristics of the generated streams (vs. legitimate ones). This covert channel allows the execution of TCP/IP applications that are able to tolerate low throughput / high latency links, such as FTP, SMTP, Web clients or chat clients.

III. The security of multimedia covert channels

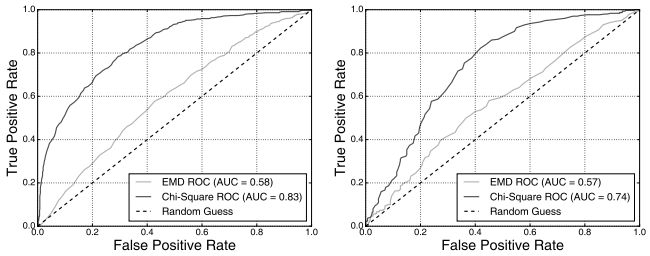
Our experience in the evaluation of DeltaShaper made us realize that the unobservability measurement of protocol tunneling systems (including our own assessment over DeltaShaper) was being performed in an *ad hoc* fashion. It was unclear whether the approaches used to assess unobservability were the best, in particular, when taking into account the recent advances in Machine Learning (ML) techniques.

A. Extensive study on protocol tunneling unobservability

Based on the insight above, we conducted an experimental study over the unobservability properties of the three state-of-the-art protocol tunneling systems which create covert channels in multimedia streaming protocols [7]: Facet [11], CovertCast [12], and DeltaShaper [6]. To the best of our knowledge, this was the first extensive experimental study comparing the performance of supervised, semi-supervised, and unsupervised ML techniques on the detection of multimedia protocol tunneling systems. The outcome of this work led to the development of a novel framework for the assessment of unobservability.

Experimental setup and methodology. In our study, we analyzed Facet's unobservability when scaling covert videos by a factor of 50%, 25% and 12.5% of the frame area (a steganography factor *s* = 50%, 25%, 12.5%, respectively). We also analyzed the unobservability of two DeltaShaper encoding configurations which respect the tuple (payload frame area, cell size, number of bits, framerate). These are comprised by the $\langle 320 \times 240, 8 \times 8, 6, 1\rangle$ and $\langle 160 \times 120, 4 \times 4, 6, 1\rangle$ tuples. Lastly, we used the single default configuration of CovertCast for conducting the experiments on this system.

We started by evaluating the unobservability of each system against the similarity-based classifiers proposed in the literature (χ^2 [11], KL [12], and EMD [6]). Then, we evaluated the unobservability of the same systems by resorting to three different decision tree-based algorithms: a simple C4.5 decision tree, and two prominent decision tree ensemble methods (Random Forest and XGBoost). Akin to the feature set used in similarity-based classifiers, we used the quantization of the frequency distribution of packet lengths as features. In our experiments, we quantize the value of packet lengths in bins of 5 bytes. Additionally, and differently from previous classification approaches, we exploited the relevance of particular ranges of the feature space by feeding these features to decision tree-based classifiers.



(a) Facet. (b) DeltaShaper.

Figure 2: ROC curve for the χ^2 and EMD classifiers when identifying Facet *s*=50% and DeltaShaper $\langle 320\times240, 8\times8, 6, 1\rangle$.

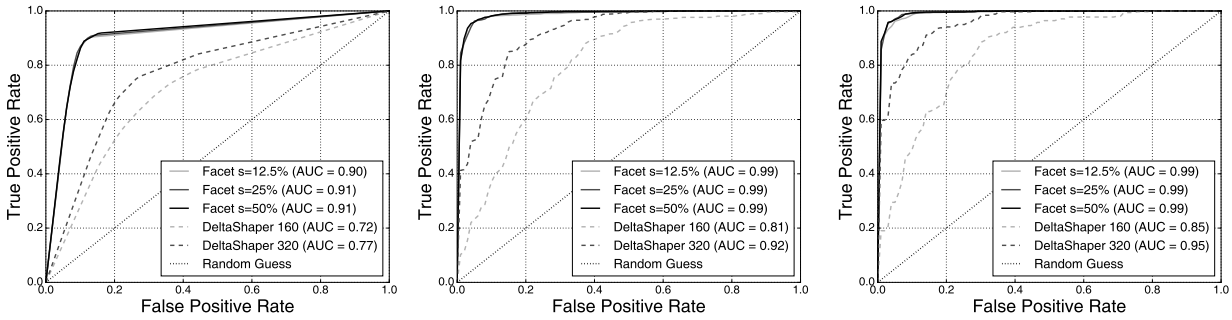
Results and collected insights. The outcomes of our experiments reveal the following interesting findings.

First, we find that CovertCast can be detected with few false positives (FP) when resorting to existing similarity-based classifiers. We conjecture two explanations that may justify the differences beween our results and those published in the original CovertCast paper. Firstly, our results may stem from the use of a dataset which is one order of magnitude larger than the one used for CovertCast evaluation. This increased dataset may more accurately represent the patterns generated by legitimate YouTube streams' traffic and reveal CovertCast activity. Secondly, implementation changes in YouTube may have impacted the unobservability properties provided by hardcoded data modulation parameters, which may in turn be no longer adequate to ensure unobservability.

Second, we conclude that the χ^2 classifier outperforms all other classifiers of its kind in the task of detecting covert channels. Figure 2 depicts the ROC curves for the χ^2 and EMD classifiers when detecting Facet *s*=50% and DeltaShaper $\langle 320 \times 240, 8 \times 8, 6, 1\rangle$ traffic. Albeit we can observe that χ^2 outperforms EMD for this kind of assessment, we also observe that the χ^2 classifier fails to detect a large amount of covert flows while sustaining a low FP rate (e.g. to block 90% of all Facet *s*=50% traffic, the χ^2 classifier erroneously tags 45% of legitimate connections as covert traffic). We omit a ROC curve for the KL classifier as it is not adjustable by an internal threshold. However, its accuracy was found to be inferior to that of a simpler version of χ^2 with no adjustable threshold.

Third, our results suggest that decision tree-based classifiers largely defy the previous unobservability claims of existing multimedia protocol tunneling systems. This finding is supported by the results in Figure 3, which show the performance of different decision-tree based classifiers when detecting Facet and DeltaShaper. For instance, 90% of all Facet *s*=50% traffic can be detected with just 2% FP rate. Furthermore, we found that the accurate detection of different systems is closely tied to disparate ranges of quantized packet lengths.

Lastly, our analysis also suggests that the existence of labeled samples is a requirement for the successful detection of multimedia protocol tunneling covert channels. We evaluated each system resorting to One-Class SVMs and Autoencoder neural networks, two state-of-the-art ML techniques able to perform one-class classification. Our results indicate that One-



(a) Decision Tree. (b) Random Forest. (c) XGBoost.

Figure 3: ROC curves of the decision tree classifiers when identifying Facet and DeltaShaper.

Class SVMs perform poorly on identifying the covert channels produced by these systems, while autoencoders show promising results for covert traffic detection. We have also evaluated the unobservability of the above systems resorting to Isolation Forest, an unsupervised learning algorithm. The outcome of this experiment reveals that an adversary has no advantage in using Isolation Forest for the detection of covert traffic.

B. Lessons Learned

The key lessons learnt from our study can be summarized as follows: a) the unobservability claims of existing multimedia protocol tunneling tools heavily rely on the classifier employed in their evaluation and past similarity-based classifiers were found to provide optimistic unobservability guarantees; b) decision tree-based classifiers can accurately detect existing multimedia protocol tunneling systems; c) different methods for embedding covert data in multimedia streams affect disparate ranges of the considered feature space.

IV. Scaling Covert Channel Detection Machinery

As we have seen in the previous section, the frequency distribution of packet sizes / inter-packet timing can be used to identify flows carrying covert channels. Unfortunately, collecting information regarding the packet size frequency distributions of live flows can be quite challenging on high-speed multi-gigabit networks without imposing considerable overheads in storage, communication, and computation. For instance, port mirroring can be used to duplicate packet headers and offload their analysis to dedicated hardware but requires a large amount of available bandwidth. Alternatively, the use of middleboxes for traffic monitoring encompasses a high infrastructure cost and complex management, and may introduce additional bottlenecks [13].

A. Measuring packet distributions at the network core

Towards tackling this pressing challenge, we devised FlowLens, a system that enables the generation of compact yet expressive representations of flows' packet size / inter-packet timing distributions at *line speed* to perform ML-based traffic analysis. By leveraging the capabilities of modern programmable switches, which can be programmed with domain-specific languages like P4 [14], FlowLens achieves considerable savings in storage and communication. In FlowLens, as packets are processed by the switch, a space-saving data structure is storing measurement information at line speed without the need to forward or duplicate packets. This information consists in a compressed representation of the packet distributions for each flow traversing the switch. This representation, named a *flow marker*, aims to provide enough information to accurately classify flows of interest. Regularly, for any given sampling period, a centralized FlowLens controller retrieves the flow markers from the switch(es) and feeds that information to be processed by an ML-engine.

Note that other space saving algorithms, such as *sketches* have also been widely used for performing flow measurement tasks [15]. The granularity of information collected by existing sketches is however insufficient for our problem domain.

Challenges of a constrained programming environment. A key challenge in designing network measurement frameworks that leverage the capabilities of programmable switches is that one is forced to use data structures that are memory efficient and algorithms that use a limited set of operations, that can be executed by the switch at line speed. Because high-end switches operate at very high speed, they incorporate expensive SRAM memories which tends to limit the amount of fast memory made available to programs running in these devices (a few tens of megabytes) [16]. Moreover, in order to guarantee packet processing at line speed, switches have but a narrow time window for packet processing, which means that algorithms have to be very efficient and cannot perform complex operations, e.g., multiplications or floating point operations are considered to be expensive operations [17].

B. Generating compact representations of flows

Key to the operation of FlowLens is the design of a suitable methodology to derive flow markers that are small, can be computed in the switch, but still provide enough information to perform accurate traffic classification.

Operators. We compute flow markers using two operations that can be implemented efficiently, namely *quantization* and *truncation*. Quantization allows us to count aggregated packet sizes arranged into bins of 2^K bytes, where K is a configurable parameter. The higher K the smaller is the memory allocated per flow, hence the higher is the compression rate. Truncation allows us to record only a subset of pre-defined bins. Higher

Bins and Memory	Quantization Factor (K)							
`len(1 bin) = 2 Bytes`	1	4	8	16	32	64	128	256
Number of Bins	1500	375	188	94	47	24	12	6
Memory per Flow(B)	3000	750	376	188	94	48	24	12

Table I: Flow marker size for different quantization factors.

truncation rate means that lesser bin registers need to be allocated, thereby achieving higher compression.

Data structures. As part of the design of FlowLens we developed a dedicated data-structure that stores flow markers on the programmable switches, and that is tailored for our quantization and truncation operations. This data-structure has been designed considering the strict constraints imposed by programmable switches and enables the computation of flow markers at line speed. One important challenge involved in generating flow markers is that finding the right quantization and truncation parameters is an application-dependent process that needs to be performed in a precursory training stage in order to determine how much compression can be achieved without significant degradation of classification accuracy.

Profiling. FlowLens implements the necessary mechanisms to automate the choice of the quantization and truncation parameters of the compact flow marker for a classification task. For this purpose, it includes a profiling tool that finds values for these parameters that match a concrete high-level goal defined by the user. For instance, the profiler can find the marker that maximizes some user-defined trade-off between space-efficiency and accuracy, or the flow marker that can maximize the accuracy given some space constraint.

C. Experimental results

Next, we present the results of a series of micro-benchmarks that offer insights on the operation of our system when used to detect covert channels generated by Facet and DeltaShaper.

Quantization can be used to detect covert channels with up to 92% accuracy using 188-byte flow markers. For performing the detection of covert channels we leverage supervised gradient boosting ML [7]. Figure 4 shows how the absolute values obtained for the accuracy, FPR, and FNR of the classifier vary when identifying Facet and DeltaShaper covert channels for different quantization factors (K). For instance, for quantization factor K=16 FlowLens can correctly identify Facet and DeltaShaper flows with less than 5% and 1% decrease in accuracy, respectively. Table I shows that, for K=16, a flow marker can be represented in 94 bins instead of a full distribution composed of 1500 bins, amounting to an order of magnitude memory savings. While a single flow marker is then represented using 188B instead of 3000B, DeltaShaper classification scores are maintained with respect to those obtained when using full information (see Figure 4).

Truncation allows for the accurate detection of covert channels using a flow marker of just 20 bytes. We elaborated a tailored truncation approach which leverages the ability of the XGBoost classifier to output the importance of features used in classification. The training phase of the classifier

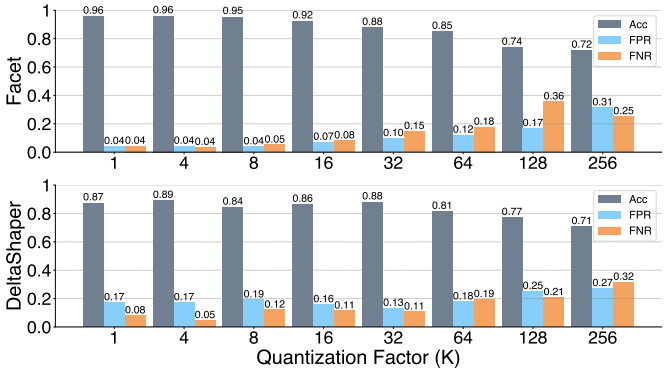


Figure 4: Accuracy, FPR, and FNR for covert channel detection when using quantized packet size distributions.

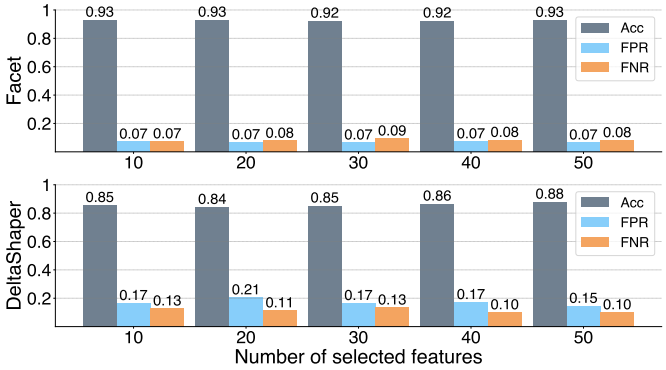


Figure 5: Accuracy, FPR, and FRN for covert channel detection when considering a different number of features for a quantization factor K=16.

involves building a model for different factors of quantization K, and to extract the top-N most important features identified for any given K. Later, FlowLens is instructed to only gather the features in a selected top-N range when deployed. Figure 5 depicts the results obtained when performing a quantization with K=16 and truncating the results obtained to the top-N features. We see that the accuracy, FPR, and FNR rate of the classifier are practically identical when using the top-10 and top-50 features to classify flows (e.g., a difference of only 1% in FNR for Facet flows), and very similar to the results obtained when using full information (Refer to K=16 in Figure 4). Thus, the use of truncation can not only maintain a high quality in covert channel classification, but further reduce the flow marker footprint from 188B (Quantization K=16) to just 20B (Quantization K=16 plus Truncation top-N=10).

V. Multimedia Covert Channels: Next Steps

We now briefly describe our on-going work with Protozoa, a novel tool to implement covert channels on multimedia streams that circumvents the limitations enumerated above.

A. Design

The main insight of our system is that of instrumenting the video pipeline of open-source multimedia frameworks to replace the contents of encoded video frames by covert data.

To preserve the correct functioning of the multimedia protocol, Protozoa injects pre-recorded encoded frames at the receiver endpoint so as to avoid video decoding errors and the trigger of congestion control mechanisms. Differently from existing tools, instead of embedding the covert data in the raw video input signal, we propose to embed the covert data at an intermediate stage within the video encoding pipeline *between* the lossy compression of the original video and the encryption of the compressed frames. Specifically, the idea is to replace the bits that result from compressing legitimate video frames with bits of covert data payload. The resulting frames will be encrypted and transmitted over the network. The receiver can then decrypt the frames and extract the covert data payload.

This scheme brings several advantages. First, the room for increasing the covert channel throughput can grow considerably. This is because we can use up all bits of the compressed video frames, thereby increasing the capacity of the channel. Secondly, because the covert payload bits are replaced in the same number as the number of bits in the original compressed video frame, the resulting video stream will preserve the same traffic characteristics, thereby undermining the effectiveness of modern traffic analysis techniques [7]. Moreover, since the replaced bits are encrypted, a network eavesdropper will not be able to read plain-text covert data.

B. Prototype evaluation

We developed and evaluated a prototype of our system over WebRTC, an open-source framework that allows for the establishment of media-rich communications between browsers. Our prototype operates on top of a modified WebRTC video pipeline of Chromium browser's native code.

A summary of the results obtained for our extensive evaluation of Protozoa is depicted in Figure 6. In this experiment, we have emulated the network conditions of a cross-continental link (with a round-trip-time of 50ms) between the endpoints involved in the communication over the `whereby.com` WebRTC service. The figure reveals that an adversary attempting to distinguish Protozoa from legitimate streams, while using the state-of-the-art XGBoost classifier (see Section III), does not obtain a significantly better chance than random guessing (AUC=0.59). We can also observe that, for the established conditions, Protozoa achieves an average throughput of 1400Kbps.

Additionally, we have found both the unobservability and performance characteristics of Protozoa to remain similar when testing Protozoa over multiple WebRTC-based applications, such as `appr.tc` or `coderpad.io`. This suggests that Protozoa can be transparently used by a multitude of WebRTC applications which adds to the censorship-resistant capabilities of Protozoa shall a censor ban the usage of a particular service.

VI. Future Work

One interesting direction for future work is tied to the major limitation exhibited by the vast range of research that analyzes the unobservability of covert channels in the Internet and that lacks the availability of real-world data. Since real traffic traces are seldomly disclosed by ISPs, the media streams used for evaluating covert channels are synthesized in laboratory. This reduces the amount of variability in terms of users network conditions and hardware, which may deem unrepresentative of a real world deployment of covert channel tools.

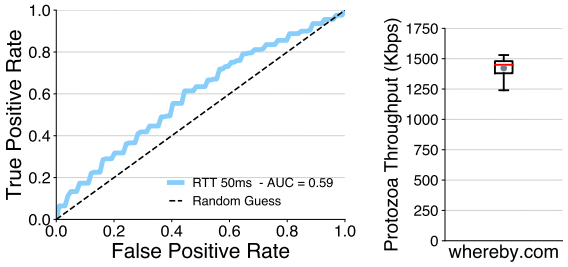


Figure 6: Results obtained for the baseline setting.

References

[1] S. Zander, G. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 44–57, 2007.

[2] C. Brubaker, A. Houmansadr, and V. Shmatikov, "Cloudtransport: Using cloud storage for censorship-resistant networking," in *PETS*, 2014, vol. 8555, pp. 1–20.

[3] B. Hahn, R. Nithyanand, P. Gill, and R. Johnson, "Games without frontiers: Investigating video games as a covert channel," in *arXiv:1503.05904 [cs.CR]*, 2015.

[4] A. Houmansadr, T. J. Riedl, N. Borisov, and A. C. Singer, "I want my voice to be heard: IP over Voice-over-IP for unobservable censorship circumvention." in *NDSS*, San Diego, CA, USA, 2013.

[5] J. Geddes, M. Schuchard, and N. Hopper, "Cover your acks: Pitfalls of covert channel censorship circumvention," in *CCS*, Berlin, Germany, 2013, pp. 361–372.

[6] D. Barradas, N. Santos, and L. Rodrigues, "Deltashaper: Enabling unobservable censorship-resistant tcp tunneling over videoconferencing streams," in *PETS*, vol. 2017(4), Minneapolis, MN, USA, 2017, pp. 5–22.

[7] ——, "Effective detection of multimedia protocol tunneling using machine learning," in *USENIX Security*, Baltimore, MD, USA, 2018.

[8] S. Khattak, T. Elahi, L. Simon, C. M. Swanson, S. J. Murdoch, and I. Goldberg, "Sok: Making sense of censorship resistance systems," in *PETS*, vol. 2016, no. 4, Darmstadt, Germany, 2016, pp. 37–61.

[9] D. Fifield, "Threat modeling and circumvention of internet censorship," Ph.D. dissertation, UCalifornia, Berkeley, 2017.

[10] Freedom House, "Freedom on the net 2018 - the rise of digital authoritarianism," https://freedomhouse.org/report/freedom-net/freedom-net-2018/rise-digital-authoritarianism.

[11] S. Li, M. Schliep, and N. Hopper, "Facet: Streaming over videoconferencing for censorship circumvention," in *Workshop on Privacy in the Electronic Society*, Scottsdale, AZ, USA, 2014, pp. 163–172.

[12] R. McPherson, A. Houmansadr, and V. Shmatikov, "CovertCast: Using live streaming to evade internet censorship," in *PETS*, vol. 2016(3), Darmstadt, Germany, 2016, pp. 212–225.

[13] C. Lan, J. Sherry, R. A. Popa, S. Ratnasamy, and Z. Liu, "Embark: Securely outsourcing middleboxes to the cloud," in *NSDI*, 2016, pp. 255–273.

[14] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *Computer Communication Review*, vol. 44, no. 3, pp. 87–95, July 2014.

[15] Q. Huang, P. P. Lee, and Y. Bao, "Sketchlearn: relieving user burdens in approximate measurement with automated statistical inference," in *SIGCOMM*, 2018, pp. 576–590.

[16] R. Miao, H. Zeng, C. Kim, J. Lee, and M. Yu, "Silkroad: Making stateful layer-4 load balancing fast and cheap using switching asics," in *SIGCOMM*, 2017, pp. 15–28.

[17] N. Sharma, A. Kaufmann, T. Anderson, A. Krishnamurthy, J. Nelson, and S. Peter, "Evaluating the power of flexible packet processing for network resource allocation," in *NSDI*, 2017, pp. 67–82.