Extended Abstract: Shaperd: Easily Adoptable Real-Time Traffic Shaper for Fully Encrypted Protocols

Sarah Wilson*
University of Waterloo
s5wilson@uwaterloo.ca

Stella Tian*
University of Waterloo
jy7tian@uwaterloo.ca

Sina Kamali
University of Waterloo
sinakamali@uwaterloo.ca

Abstract

Fully encrypted protocol-based tools (FEPs) are tools commonly used to circumvent censorship in restrictive regions, valued for their performance and security. However, in recent years, censors have been able to block them using an array of attacks based on passive traffic analysis and active probing. We propose Shaperd, an easily adoptable and real-time traffic shaper designed specifically to aid FEPs become more resilient to detection. Shaperd operates directly on packet contents in real time, using a novel constraint system to allow its users to generate traffic flows with any desired features. Our preliminary results reveal Shaperd introduces minimal overhead to the underlying system's throughput.

1 Introduction

Over time, more countries are resorting to internet censorship in times of political unrest, protests, or elections [2, 7]. This restriction on the information flow results in the censor's complete control over the public's perception of ongoing events, which could strongly impact the outcome of the situation.

To thwart internet censorship attempts, users have to resort to censorship circumvention (CC) systems. Many CC systems have been proposed over the years, with various degrees of practicality, effectiveness, and performance [10, 16]. These tools range from solutions such as decoy routing [3], which are potentially effective and resilient, but are hard to deploy, to data embedding solutions such as multimedia protocol tunneling [6, 9, 11], which are considered secure, but lack the required performance for daily activities.

One of the most used and widely adopted CC systems are tools based on fully encrypted protocols (FEPs) [5, 8]. FEPs function by encrypting the entirety of the traffic payload, making the traffic appear as random bytes. This approach allows FEPs to effectively hide traffic metadata, such as the underlying protocol and data length, which enables them to avoid censoring tactics that rely on protocol or domain blocklisting [15] as the censor cannot recognize them. Doing so effectively makes the data sent by FEPs "look random" [4]. There are many examples of FEPs, with varying security guarantees, including Shadowsocks [12] and VMess-based [18] systems.

Even though FEPs seem robust, they are easily recognizable when used without additional security measures, as "looking like nothing" is a standout feature. Recently, censors have been using a variety of methods ranging from statistical analysis to active probing to detect and block commercial FEPs such as Shadowsocks. Alice et al. [1] demonstrated how Shadowsocks can be detected using the packet length and entropy. Later on, Wu et al. [22] highlighted the extent of the detectability of FEPs, and were able to shed light on specific rules upon which FEPs are detected. They reveal that packets are blocked based on ad-hoc rules such as the number of printable ASCII characters, or the fact that the entropy of normal encrypted TLS packets is considerably lower than FEPs. These rules are subject to frequent changes, so the rigid design of older FEPs is unable to adapt to the updated detection rules. Additionally, Fenske et al. [5] deliver an extensive study on available FEPs and their features, highlighting their shortcomings, and revealing how packet lengths and protocol inconsistencies can be a main source of detection. They focus on fixing packet lengths using some primitive packet length shaping but leave the how to shape the traffic or shaping packet timings for future work. Although none of the aforementioned work discusses the effects of packet timings on the detection of FEPs, Wails et al. [21] showcase the general importance of packet timings in detecting CC tools.

Fortunately, a solution for the challenges outlined above exists, a FEP-specific traffic shaper. Traffic shapers exist in other domains (e.g., IoT) to add a layer of privacy to the data in transit [23]. A FEP-specific traffic shaper could in theory shape data so it evades rule-based detection, helps normalize packet lengths, and control packet timings to thwart all previously mentioned attacks. Some prior work such as Proteus [20] discusses how traffic shaping could help improve their tool's privacy guarantees, and some such as Fenske et al. [5] implement their own traffic shaper, but lack packet timing shaping and are not designed for integration with other solutions. Nevertheless, to the best of our knowledge, no prior stand-alone FEP traffic shaper exists that can address our concerns.

In this paper, we introduce Shaperd, a real-time traffic-shaping tool that can easily be integrated with any existing FEP. Shaperd provides a novel *constraint* system, enabling its users to define the features of the generated traffic, shaping both the packets lengths and timings of a given traffic flow. Shaperd aims to be an independent traffic shaper that is compatible with all prior FEPs, thus addressing unobservability challenges highlighted by Wu et al. [22]. The generality of Shaperd's constraint system enables users to generate traffic flows with any desired features, such as particular values for a packet's entropy, to circumvent detection systems. Moreover, this allows integration with previous FEPs detected using such methods while incurring minimal performance overheads.

Our main contributions are as follows: a) Shaperd, a traffic-shaping system that can easily be adopted by previous CC systems, allowing them to overcome detection techniques, and b) a novel constraint system that enables users to describe how they want to shape their traffic, resulting in a flexible traffic flow.

*Authors contributed equally to this work.

Free and Open Communications on the Internet 2025(2), 37–39

2 Technical Details

In this section, we introduce Shaperd and its inner workings. Shaperd is designed to be easily integrated with any existing or future FEP, such as Shadowsocks [8]. Shaperd relies on its constraint system to allow users to adapt to the ever-changing detection systems used by the adversaries, such as the ad-hoc rules used by the censor analyzed by Wu et al. [22]. Shaperd uses two simple but effective methods to shape a packet's content, which was designed with constraint agnosticism in mind.

Workflow. The overall workflow of Shaperd is as follows. Alice, a Shaperd user, has a set of constraints outlining her desired traffic flow. This constraint set can either be created by her, received from trusted sources (e.g., how bridge information is distributed [17]), or a mix of both. Then, Alice starts a Shaperd instance with the constraint set and routes her FEP through Shaperd. Data received by Shaperd is stored in a traffic buffer prior to being shaped according to the defined constraints. As a part of this process, the input traffic goes through two separate systems before being sent out: the shaper and timer threads.

The shaper thread constantly monitors the traffic buffer, directly inspecting the contents of the buffered packets to generate new valid packets as soon as the contents either exceed the maximum capacity of a valid packet or a certain predefined period. Packets are created using two main shaping methods: a) length reduction, where a packet's content length gets reduced until the constraints are satisfied, and b) content padding, during which potential packets are padded with incremental byte values until a satisfying packet is found. We are using naive padding, leaving the exploration of more efficient heuristics for future versions (see a more detailed discussion in Appendix A). Length reduction, although simple, has proven to be more effective and performative in our preliminary testing, so in our current version, content padding is used as a fallback when length reduction fails. Finally, the shaper sends all created packets to the timer thread.

The timer thread is in charge of deciding when packets are sent out according to the preset timing constraints. These constraints can denote the flow throughput, the minimum and maximum delay between any two packets, or any other information regarding packet timing. We note that the time thread is currently a work in progress. Our goal is to support real-time shaping of packet timing by enforcing user-specified constraints on delay and throughput. We aim to finalize and test this feature in the near future. We discuss the timer and its state more thoroughly in §3.

The constraint framework. We present the constraint framework, a simple but effective system for defining content or timing-specific conditions on packets in traffic flows. A content constraint consists of four main components: a) the constraint function, a function that calculates the metric to be assessed, b) a value in the unit of the constraint function, which will be compared against, c) a mode, that sets the comparison operator (e.g., equal, lower than, or greater than), and d) a target, which denotes which packets this constraint should affect. We can support any constraint on a value expressible by a well-defined, deterministic function, however, constraints defined by non-computable functions, cannot be handled. A practical example of valid constraint could be one of the proposed rules described by Wu et al. [22], that there needs to be over 50% of ASCII printable characters in a given packet. For this particular example, our constraint function would be one that calculates the percentage of ASCII printable characters in a given packet, the value would be 50%, the mode would be equal to or greater than, and the target could be all packets. Timing constraints, on the other hand, are still a work in progress. We envision that users will be able to define constraints such as minimum and maximum delay between consecutive packets, fixed inter-packet intervals, or jitter bounds. These constraints will guide the timer thread in determining when to release packets from the queue, enabling realistic and tunable traffic shaping behavior.

Implementation. We have implemented a proof-of-concept of Shaperd [13] and tested it using a mix of unit and real-time client-server tests. We developed our prototype using Go with ∼1000 lines of code and implemented a client that generates random bytes to act as the traffic generator in testing. In our tests, we mainly used a simple entropy constraint on messages to evaluate Shaperd's overhead. Our preliminary results reveal a minimal overhead of 4.1% over the input throughput. This overhead scales with the number of additional constraints on the packets. Our evaluation revealed that the first constraint added an overhead of 5.1% and the second added 5.5% of overhead. We note that these values are subject to change, based on how rigorous the chosen constraints are.

Usage scenarios. We believe Shaperd can prove useful by not only helping widely deployed tools such as Shadowsocks and VMess-based systems achieve a new strong layer of privacy but also the development of new FEPs by shifting the concerns from traffic shaping to the development of other features and improvements.

3 Discussion and Future Work

As suggested by §2, Shaperd's performance has room for improvement. There are several features that we are actively working on.

Constraint agnosticism. Shaperd's constraint agnostic design is both a strong point of our work and a major performance drawback. To accommodate this, we decided to keep our constraint agnostic design, while adding an optional field named "type" to our constraints. The type field can be used by the shaper to better understand the constraint and satisfy it more efficiently.

Supported protocols. The current version of Shaperd only supports shaping TCP packets. We plan to add support for more protocols in the future to allow a wider array of tools to be at the disposal of our users and to allow more flexible traffic morphing strategies for evading modern detection techniques.

Evaluation. Our current evaluation methodology uses a random byte generator as the client. We aim to test Shaperd in combination with state-of-the-art academic tools such as Proteus [20], or widely adopted open-source tools such as Shadowsocks and V2Ray. Moreover, we plan to use sophisticated traffic fingerprinting techniques [14, 22] to evaluate the effectiveness of our shaper.

To conclude, we introduce Shaperd, a CC-specific traffic shaper designed for seamless integration with existing FEPs. By leveraging a novel constraint system, our approach allows tool designers to offload the complexity of traffic shaping. While Shaperd is still a work in progress, we are committed to advancing its development and capabilities.

References

[1] Alice, Bob, Carol, Jan Beznazwy, and Amir Houmansadr. 2020. How china detects and blocks shadowsocks. In *Proceedings of the ACM Internet Measurement Conference*. 111–124.

[2] Simurgh Aryan, Homa Aryan, and J Alex Halderman. 2013. Internet censorship in Iran: A first look. In *3rd USENIX Workshop on Free and Open Communications on the Internet (FOCI 13)*.

[3] Cecylia Bocovich and Ian Goldberg. 2016. Slitheen: Perfectly imitated decoy routing through traffic replacement. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 1702–1714.

[4] Lucas Dixon, Thomas Ristenpart, and Thomas Shrimpton. 2016. Network traffic obfuscation and automated internet censorship. *IEEE Security & Privacy* 14, 6 (2016), 43–53.

[5] Ellis Fenske and Aaron Johnson. 2024. Bytes to schlep? Use a FEP: Hiding protocol metadata with fully encrypted protocols. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 1982–1996.

[6] Gabriel Figueira, Diogo Barradas, and Nuno Santos. 2022. Stegozoa: Enhancing webrtc covert channels with video steganography for internet censorship circumvention. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. 1154–1167.

[7] Nguyen Phong Hoang, Arian Akhavan Niaki, Jakub Dalek, Jeffrey Knockel, Pellaeon Lin, Bill Marczak, Masashi Crete-Nishihata, Phillipa Gill, and Michalis Polychronakis. 2021. How Great is the Great Firewall? Measuring China's {DNS} Censorship. In *30th USENIX Security Symposium (USENIX Security 21)*. 3381–3398.

[8] Qingbing Ji, Zhihong Rao, Man Chen, and Jie Luo. 2022. Security analysis of shadowsocks (r) protocol. *Security and Communication Networks* 2022, 1 (2022), 4862571.

[9] Watson Jia, Joseph Eichenhofer, Liang Wang, and Prateek Mittal. 2023. Voiceover: Censorship-Circumventing Protocol Tunnels with Generative Modeling. *Free and Open Communications on the Internet* (2023).

[10] Sheharbano Khattak, Mohammad Tariq Elahi, Laurent Simon, Colleen M Swanson, Steven J Murdoch, and Ian Goldberg. 2016. SOK: Making sense of censorship resistance systems. *Water Treatment Technology* 2016, 4 (2016), 37–61.

[11] 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*. 97–108.

[12] Shadowsocks org. 2025. Shadowsocks. https://shadowsocks.org.

[13] Shaperd. 2025. Shaperd. https://zenodo.org/records/15259131.

[14] Meng Shen, Kexin Ji, Zhenbo Gao, Qi Li, Liehuang Zhu, and Ke Xu. 2023. Subverting website fingerprinting defenses with robust traffic representation. In *32nd USENIX Security Symposium (USENIX Security 23)*. 607–624.

[15] Jonathan Spring and Leigh Metcalf. 2015. Domain Blocklist Ecosystem - A Case Study. Carnegie Mellon University, Software Engineering Institute's Insights (blog). https://insights.sei.cmu.edu/blog/domain-blocklist-ecosystem-a-case-study/ Accessed: 2025-Apr-19.

[16] Michael Carl Tschantz, Sadia Afroz, Vern Paxson, et al. 2016. Sok: Towards grounding censorship circumvention in empiricism. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 914–933.

[17] Lindsey Tulloch and Ian Goldberg. 2023. Lox: Protecting the Social Graph in Bridge Distribution. *Proceedings on Privacy Enhancing Technologies* 1 (2023), 494–509.

[18] Project V. 2025. VMess | Project V. https://www.v2ray.com/en/.

[19] Vecna. 2024. Troll Patrol: Detecting Blocked Tor Bridges. https://dspacemainprd01.lib.uwaterloo.ca/server/api/core/bitstreams/2825c095-3fbe-4b71-a8a7-2ae6b7e7eea8/content.

[20] Ryan Wails, Rob Jansen, Aaron Johnson, and Micah Sherr. 2023. Proteus: Programmable protocols for censorship circumvention. *Free and Open Communications on the Internet* (2023).

[21] Ryan Wails, George Arnold Sullivan, Micah Sherr, and Rob Jansen. 2024. On precisely detecting censorship circumvention in real-world networks. In *Network and Distributed System Security*.

[22] Mingshi Wu, Jackson Sippe, Danesh Sivakumar, Jack Burg, Peter Anderson, Xiaokang Wang, Kevin Bock, Amir Houmansadr, Dave Levin, and Eric Wustrow. 2023. How the Great Firewall of China detects and blocks fully encrypted traffic. In *32nd USENIX Security Symposium (USENIX Security 23)*. 2653–2670.

[23] Sijie Xiong, Anand D Sarwate, and Narayan B Mandayam. 2022. Network traffic shaping for enhancing privacy in iot systems. *IEEE/ACM Transactions on Networking* 30, 3 (2022), 1162–1177.

A Additional Discussion

Padding algorithm. We plan to use more prominent padding solutions, as most of the current performance loss stems from our naive padding system which tries to find proper padding bytes iteratively, which scales exponentially the more padding bytes a flow requires. This solution helps Shaperd's flexibility but is inefficient. We plan to use a heuristic-based padding system that selects padding bytes more efficiently in the future.

Timing and blocking redirection. To better support adaptive circumvention, we are exploring how Shaperd can detect and adapt to blocking. For detection, we can use tools such as Troll Patrol [19], which tackle various methods to detect endpoint blockages. Then, to adapt to blocking, Shaperd can adopt alternate constraint sets to alter packet patterns, thereby circumventing the block.