DeResistor: Toward Detection-Resistant Probing for Evasion of Internet Censorship

Abderrahmen Amich[*], Birhanu Eshete[*], Vinod Yegneswaran[§], and Nguyen Phong Hoang[†]

[*]*University of Michigan, Dearborn,* [§]*SRI International,* [†]*University of Chicago*

Abstract

The arms race between Internet freedom advocates and censors has catalyzed the emergence of sophisticated blocking techniques and directed significant research emphasis toward the development of automated censorship measurement and evasion tools based on packet manipulation. However, we observe that the probing process of censorship middleboxes using state-of-the-art evasion tools can be easily fingerprinted by censors, necessitating detection-resilient probing techniques.

We validate our hypothesis by developing a real-time detection approach that utilizes Machine Learning (ML) to detect flow-level packet-manipulation and an algorithm for IP-level detection based on Threshold Random Walk (TRW). We then take the first steps toward detection-resilient censorship evasion by presenting DeResistor[1], a system that facilitates detection-resilient probing for packet-manipulation-based censorship-evasion. DeResistor aims to defuse detection logic employed by censors by performing detection-guided pausing of censorship evasion attempts and interleaving them with normal user-driven network activity.

We evaluate our techniques by leveraging *Geneva*, a state-of-the-art evasion strategy generator, and validate them against 11 simulated censors supplied by Geneva, while also testing them against real-world censors (i.e., China's Great Firewall (GFW), India and Kazakhstan). From an adversarial perspective, our proposed real-time detection method can quickly detect clients that attempt to probe censorship middleboxes with manipulated packets *after inspecting only two probing flows*. From a defense perspective, DeResistor is effective at shielding Geneva training from detection while enabling it to narrow the search space to produce less detectable traffic. Importantly, censorship evasion strategies generated using DeResistor can attain a high success rate from different vantage points against the GFW (up to 98%) and 100% in India and Kazakhstan. Finally, we discuss detection countermeasures and extensibility of our approach to other censor-probing-based tools.

[1]Could also be thought of as a system that delays and distributes censorship probes similar to how a resistor regulates the flow of current.

1 Introduction

An increasing number of nation states are resorting to widespread and draconian censorship of network traffic to suppress access to various forms of information (e.g., offensive content, national-security threats, or politically uncomfortable literature). A popular and particularly egregious form of censorship is blocking of forbidden keywords/domain names (e.g., China [67], Pakistan [46], India [30]).

Through measurement studies and engagements with censorship middleboxes, researchers have observed that the state-of-the-practice of such keyword-based censorship involves the use of TCP Reset (RST) packets [40] or misguiding clients (e.g., via DNS spoofing [7]). One way to confuse censors and disarm them of their ability to tear down connections is to perform evasive client-side *packet manipulations*. Early efforts in censorship evasion involved manually crafting evasion strategies [42, 63]. However, these countermeasures would be easily thwarted by censors if widely deployed or integrated into circumvention tools such as pluggable transports [4, 44].

With the continuing cat-and-mouse game between anti-censorship researchers and censoring regimes, the quest for reliable censorship circumvention solutions has given rise to sophisticated and automated approaches such as Alembic [45], SymTCP [64], and Geneva [13]. These approaches, while having different strategies, are all designed to manipulate IP packets at the transport layer with a goal to confuse censorship middleboxes, thus enabling users to connect to censored destinations.

Although these tools have been demonstrated to craft successful evasion strategies against real-world censors, we observe that during their censor-probing phase, they produce abnormal network patterns that substantially differ from normal flows. Such potential distinguishability of packet-manipulated flows from ordinary ones leads to a practical concern: if these tools are ubiquitously deployed, censors will leverage this vulnerability to detect them. The implication of such detection depends on what the censor chooses to do but could involve: (*i*) early blocking of clients that are running evasion tools to

prevent them from reaching a successful evasion strategy; (*ii*) misleading clients into converging on wrong or ineffective strategies; or (*iii*) even referral to law enforcement [3].

Our paper focuses on Geneva [13], a genetic algorithm-based censorship evasion tool that *automates* the generation of evasion strategies against a target censor. To investigate the feasibility of detecting censorship evasion systems, we extend our preliminary study [6] in which we systematically investigate the behavior of Geneva on behalf of an adversary (e.g., a censoring regime) that aims to detect Geneva flows and block them. We mimic an intelligent censor by proposing a real-time detection approach that accurately detects all clients running Geneva after fewer than three observations (§4). We do so by leveraging Machine Learning (ML) techniques for flow-level detection (§4.1) and the *Threshold Random Walk (TRW)* algorithm [39] for IP-level detection using sequential hypothesis testing (§4.2). Our ML detectors detect Geneva flows with $\approx 99\%$ accuracy. To make a final detection decision on Geneva clients, we extend the classification system by leveraging TRW —a hypothesis testing framework first developed for port-scan detection. Our results reveal that Geneva clients are easily detectable using ML detectors that enable TRW to make fast and accurate rejection of Geneva sources, typically after approximately two flow-level observations. This is achieved with a low false positive rate on diverse types of publicly available traffic packet capture datasets (e.g., normal traffic, network forensics, and malware infection) [6]. These findings suggest that probing-based censorship evasion systems like Geneva need to be detection-resilient in the face of dynamic and resourceful adversaries (e.g., state-sponsored filtering middleboxes).

In light of our finding that evasion tools like Geneva are susceptible to adversarial detection, we explore the development of a detection-resilient probing and circumvention strategy generation system to effectively cope with the detect-block arms race. We proceed by first framing the censorship problem as a two-player game, where a client in a censored regime learns to improve its evasive packet manipulation strategy generator through the feedback (e.g., censoring, detection) returned from its opponent (i.e., the censor). Grounded in this framing, we propose DeResistor, a system extension that enables censorship evasion systems like Geneva to generate detection-resilient packet-manipulation strategies to evade censors. More precisely, we consider a censor that not only performs censorship on the client's connections to a forbidden server, but also includes a detection module that captures any packet manipulation attempt. Using the censor's feedback, the client automatically learns better strategies by optimizing a two-objective fitness function to find a trade-off between a strategy's *effectiveness* and its *detection-resilience*. Furthermore, to avoid early rejection (i.e., IP blocking), we enhance the client with a second module that offers a guided pausing of the strategy generator training (probing phase) and switching to normal network traffic to confuse the censor's detector. This measure is crucial to delay the convergence of the TRW algorithm into a detection decision until the censor probing phase discovers a working evasion strategy.

We first evaluate DeResistor via *in-situ* experiments against 11 mock censors from the Geneva paper [13] and confirm its detection-resilience across multiple runs. Our results suggest that while a client running Geneva is detected and IP-blocked after only two packet-manipulation attempts, DeResistor succeeds in avoiding IP-blocking until it finishes its training and generates effective strategies. We then field-test DeResistor's censorship evasion effectiveness against China's GFW, India and Kazakhstan. Specifically, we run DeResistor using Geneva as a strategy generator from multiple vantage points in Mainland China, one vantage point in India and one vantage point in Kazakhstan and evaluate its detection-resilience against the ML and TRW-based detection approaches proposed in §4. We show that DeResistor can make Geneva's training phase more resistant to detection while maintaining a high censorship evasion success rate. To that end, this paper makes the following contributions:

- We propose the first real-time detection approach of clients probing a censor with manipulated packets for censorship evasion.
- We develop and evaluate DeResistor, an approach that protects the censor-probing phase of censorship evasion tools from being detected.
- We show that DeResistor can guide Geneva into learning effective evasion strategies against real-world censors with less detectable features.
- We discuss censor-side countermeasures and DeResistor's adaptability to different classes of probing-based censorship evasion tools.

To foster future research, we have made our code publicly available at: https://github.com/um-dsp/DeResistor.

2 Background and Related Work

We first review some background information and related work on censorship measurement and evasion to facilitate the explanation of our approaches in §4 and §5, as well as the motivation behind the development of DeResistor.

2.1 Censorship Measurement

Early studies on Internet censorship focused on a small number of countries with stringent information control policies, notably China [41, 43, 50, 66, 67] and Iran [9].

To better understand GFW's operating mechanism, Xu et al. [67] used probes with limited time-to-live (TTL) values in TCP header to determine locations from which RST packets were injected. Crandall et al. [18] and Park et al. [50] studied the effectiveness of keyword-based filtering in China. Khattak et al. [40] studied limitations in the TCP and HTTP inspections performed by GFW.

Over the last decade, with the prevalence of authoritarian

governments making use of different network interference technologies for censorship purposes, the literature has witnessed more in-depth studies in different regions of the world, including India [58, 69], Pakistan [5, 46], Russia [54, 68], Syria [15], Thailand [27], and Turkmenistan [49]. This new wave of censorship around the world has also led to the development of global censorship monitoring platforms such as OONI [26], CensoredPlanet [52], and ICLab [47]. Despite the difference in scale of these measurement studies, most of them are designed to discover network interference techniques widely used by state-sponsored censors via DNS tampering [7, 8, 36, 51, 55], TCP packet injection [34, 61, 65], and IP-based blocking [35, 37] (e.g., via null routing [60]).

2.2 Censorship Evasion

Prior research on bypassing censorship middleboxes have broadly adopted one of two approaches. The first approach involves tunneling censored traffic, e.g., via domain fronting [25], VPNs [48], or anonymity network relays such as Tor [23] or I2P [35]. The second approach, which is also the focus of our study, relies on confusing filtering middleboxes with crafted packets [13, 42, 45, 49, 63, 64, 66] or ignoring packets injected by censors [24, 36].

Some earlier works, including INTANG [63], Lib•erate [42], and brdgrd [66], relied on semi-automated efforts to discover packet-manipulation strategies through characterization and reverse engineering of a targeted censor's blocking mechanism. Nevertheless, censors do adapt and may change their blocking behaviors over time [8, 36, 49]. As a result, more sophisticated methods have been introduced with new capabilities to automate the process of strategy generation through Genetic algorithms (Geneva [13]) or symbolic execution (e.g., SymTCP [64], Alembic [45]).

Of these tools, Geneva distinguishes itself with demonstrated capabilities for not only re-deriving strategies described in prior manual efforts but also generating new (previously unknown) evasion strategies against multiple censorship regimes, including China [14], India [10], Iran [11], Kazakhstan [33], and Turkmenistan [49]. We therefore choose Geneva as a foundational building block for DeResistor.

2.3 Geneva as a Motivation

Geneva implements a genetic algorithm that automatically derives packet-manipulation-based evasion strategies against a censor. Geneva's strategies stem from four basic manipulation primitives: drop, tamper headers, duplicate, and fragment packets.

Strategies in Geneva comprise a set of (trigger, action tree) pairs. Packets that match a given trigger (for instance, all TCP packets with the ACK flag set) are modified using the corresponding sequence of actions in an action tree. Triggers represent TCP/IP fields in a packet header that, when matched, cause packet manipulation actions to be applied. Actions are the aforementioned permitted packet manipulations.

Geneva automatically derives censorship evasion strategies through evolution of a series of generations. Each generation comprises multiple individuals. The evolution is achieved by random mutation that can occur at the level of actions, triggers or entire individuals. Geneva also performs crossover between a pair of individuals in the same generation. At the final step of a generation, it runs a selection tournament. Some individuals are drawn at random (with replacement) from the population; the highest-fitness individual among them is added to the offspring pool. This process repeats until the offspring pool is the same size as the population pool. The offspring pool then becomes the population for the next generation. During its training, Geneva evaluates fitness by running each strategy directly against the censor, resulting in multiple probing attempts.

Due to the design choice of Geneva during the training phase in which a censor is triggered repetitively, we are interested in investigating whether a censor can detect this probing traffic to hinder the automated process of discovering evasion strategies. In §4 we show that Geneva-generated probing traffic can be easily and quickly detected. This pitfall is the primary motivation for the creation of DeResistor (§5). We do not introduce DeResistor as another strategy generation tool to evade censorship, but we build it to offer other evasion tools the protection from being detected by making their repetitive censor-probing phase more resilient to detection.

3 Problem Formulation

As the arms-race between censored users and state-sponsored censors continues, our goal is to build a detection-resilient probing and evasion system that provides users with real-time protection against censor-side detection. Anti-censorship technology deployers and censorship middleboxes can be considered as two contending agents engaging in a two-player game. The instance of the game is that the censored user, whose intention is to evade censorship, generates a strategy to bypass the censor's filtering middlebox. The user strives to maximize their odds of bypassing the censor's detection and/or blocking, whereas the censor aims to minimize the chance that the user could evade it via its evasive maneuvers. We explore two designs to model the game, *generative adversarial models* and *genetic algorithms*.

3.1 Generative Adversarial Models

Generative Adversarial Networks (GANs) [29]) offer a natural formalism of the aforementioned two-player game as a Minimax game [57]. In particular, an Internet user can be modeled as a Generator G and the censor as a Discriminator D. In such a setup, while D learns to discern legitimate connections from the censored ones (e.g., connection to a censored server), G learns more advanced packet manipulation strategies that can overcome D's censorship techniques.

At a high-level, the GAN setup seems suitable to our problem. Yet upon deeper examination, we find that such a setup

is subject to *data embedding-related constraints*, especially if we aim to evade real-world censors. Suppose an Internet user inside a censorship regime attempts to connect to a censored server. As per the GAN setup, G can be trained to take as an input a normal network flow (i.e., without packet manipulations) and generates a manipulated counterpart flow that is tested against D. In this case, a flow has to be embedded to a vector representation x. Next, G performs manipulations to produce $\tilde{x}$. Previous work proposed multiple byte embedding methods (e.g., AttackGAN [17], PacketCGAN [62], PAC-GAN [16], and Flow-WGAN [31]). Unfortunately, these byte embedding methods are not lossless. Hence, the embedding function is neither invertible nor differentiable. Consequently, mapping back $\tilde{x}$ to construct a unique network flow amenable to being tested against the censor is practically challenging. Moreover, even if it is possible to map $\tilde{x}$ to multiple estimations of packets, there is no guarantee that the constructed manipulated packets, if found, preserve the same functionality/goal of the desired traffic. In light of these practical limitations of the GAN setup, next we explore genetic algorithm-based modeling of detection-resilient censorship evasion.

3.2 Genetic Algorithms

We shift our focus to another generative modeling technique that is suitable to the problem at hand and avoids the limitations of the GAN alternative. In particular, we model the generator G by a genetic algorithm that aims to fool the censor (playing the role of the discriminator D). The censor's response is (1) whether or not the generator's connection attempt is to be *censored* and (2) whether a probing/measurement tool is *detected*. The training of the generator is governed by the maximization of a fitness function that considers the feedback of the censor.

Fortunately, Geneva's genetic algorithm [13], explained in §2.3, is a natural fit to play the Generator role. By leveraging Geneva, we generate packet manipulation strategies as a set of (trigger, action tree) pairs. For instance, if a packet matches a given trigger (e.g., SYN flag), it is modified using the corresponding sequence of actions in an action tree. As a result, we avoid the data-embedding problem of the GAN-based formulation. The key challenge here is, if the censor detects and blocks connections from the generative model (e.g., Geneva), then how can we train the generator to make seamless packet manipulation? In §4, we show that automated censorship evasion tools like Geneva can be detected easily by the censor in real-time. In §5, we then propose a detection-resilient design that operates on top of existing strategy generators. Given Geneva's suitability to our problem and its reported effectiveness, in the remainder of this paper, we use Geneva's genetic algorithm to illustrate strategy generation to better explain our detection-resilient probing and censorship evasion approach. We also discuss in §7 DeResistor's adaptability to other censor-probing based strategy generation tools (e.g., SymTCP [64], INTANG [63], Lib•erate [42] and others).

4 Real-Time Censorship Evasion Detection

To demonstrate how probing traffic of automated evasion tools can be easily detected, we introduce a two-step approach to detect Geneva clients on the censor side with high confidence.

4.1 Flow-Level ML-Based Detection

By running Geneva against the censor, middlebox operators can collect Geneva traces and train a ML model f_{Censor} that distinguishes Geneva traffic from normal traffic. Figure 1 shows feature analysis of Geneva flows. From the density plots, we notice that Geneva TCP packets have several corrupt data-offset fields and tend to have smaller size compared to normal traffic. Furthermore, Geneva may tamper with other TCP header fields like checksum or TTL, as part of its probing design to locate filtering middleboxes. We also notice that overlapping TCP segments are more likely to occur in Geneva traffic due to the tampering of packet payloads. Using these distinctive features, a fairly simple ML model (e.g., Decision Trees, Random Forests) is able to accurately distinguish Geneva flows from normal flows. Figure 2 shows that all four models (Decision Trees (DT), Random Forests (RF), Logistic Regression (LR) and Support Vector Machines (SVM)) can detect almost all Geneva flows in the test set, with negligible false positives (AUC > 0.99).

However, we notice that some Geneva flows might resemble normal flows, which could confuse f_{Censor} and result in an incorrect decision based on observing only one flow. Consequently, a one-flow detection algorithm is ineffective to be deployed for real-time blocking. To address this limitation, we develop a sequential hypothesis-testing approach based on the popular TRW algorithm [39] used in port-scan detection. TRW enables a censor to make more confident multi-flow decisions (block or pending) at the IP level.

4.2 IP-Level TRW-Based Detection

For a given source IP address **(IP)**, let Y_i be a random variable that represents the outcome of the i^{th} flow started by **(IP)**. In our case, the outcome of an observation is the prediction made by f_{Censor}. Formally, $Y_i = f_{Censor}(x_i)$. If f_{Censor} classifies x_i as Geneva flow, $Y_i = 0$. Otherwise, $Y_i = 1$. As outcomes $Y_1, Y_2, ...,$ are observed, we wish to determine, with high probability of correctness, whether **(IP)** is a Geneva client. Additionally, we would like to make the decision as quickly as possible before Geneva successfully evades the censor. Specifically, we consider two hypotheses $\mathcal{H}_0$ and $\mathcal{H}_1$, where $\mathcal{H}_0$ is the hypothesis that source **(IP)** is running Geneva and $\mathcal{H}_1$ is the hypothesis that it is a benign source. The TRW algorithm assumes that conditional on the hypothesis $\mathcal{H}_j$, the random variables $Y_i|\mathcal{H}_j, i = 1, 2, ...$ are independent. Then, we express the distribution of the Bernoulli random variable Y_i as:

$$\begin{aligned} Pr[Y_i = 0|\mathcal{H}_0] = \theta_0, \quad Pr[Y_i = 1|\mathcal{H}_0] = 1 - \theta_0 \\ Pr[Y_i = 0|\mathcal{H}_1] = \theta_1, \quad Pr[Y_i = 1|\mathcal{H}_1] = 1 - \theta_1 \end{aligned} \tag{1}$$

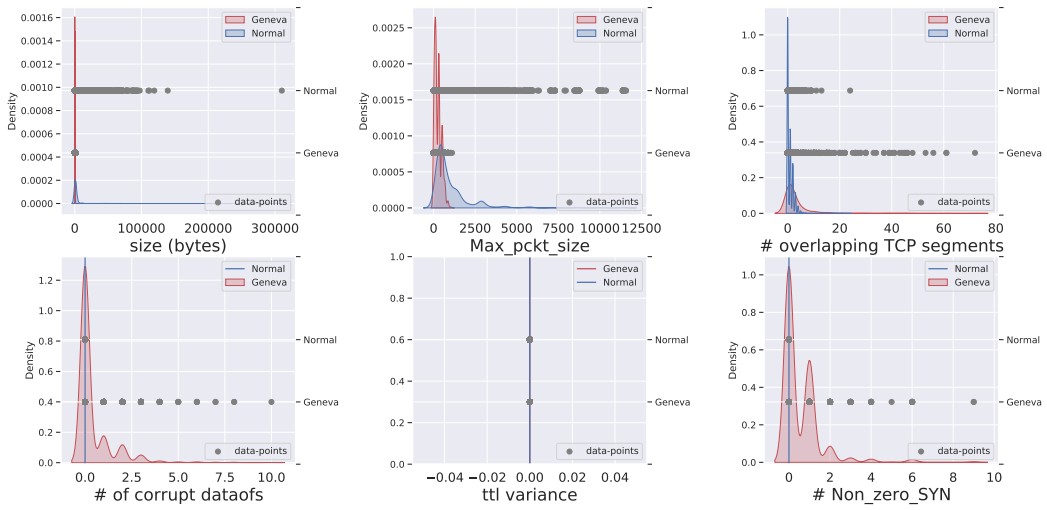

Figure 1: Geneva training feature analysis. Scatter plots show data-points of Geneva vs. normal traffic. Curves show data distribution density of each traffic type.

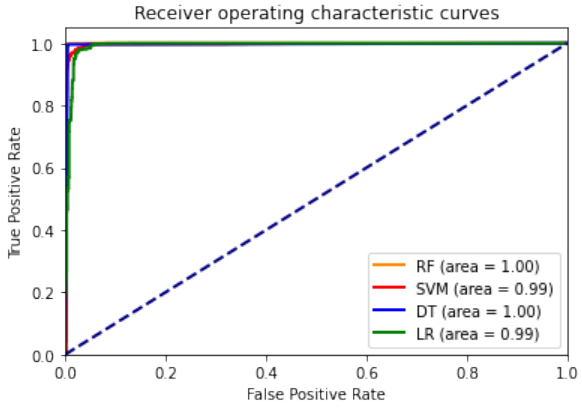

Figure 2: ML model performance for flow-level detection.

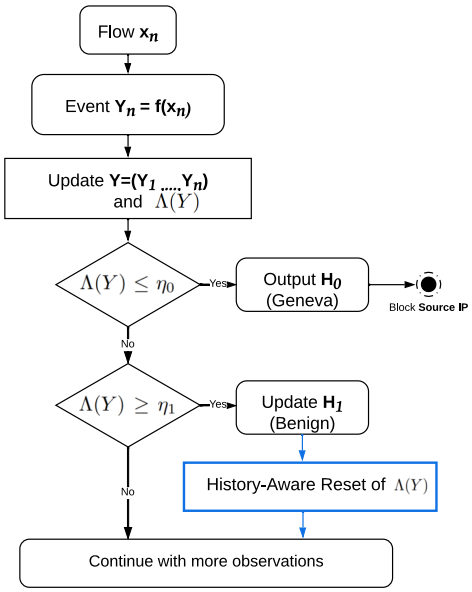

Figure 3: Flow diagram of TRW for real-time Geneva detection. The blue box represents an extension to the original TRW algorithm.

Since f_{Censor} is trained to make mostly correct predictions, if **(IP)** makes a Geneva attempt x_i, then $Y_i = f_{Censor}(x_i) = 0$ is the most likely observation. Consequently, $\theta_0 > \theta_1$. With respect to Equation 1, θ_0 and θ_1 can be set as: $\theta_0 = \frac{TP}{TP+FN}$ and $\theta_1 = \frac{FP}{TN+FP}$, where TP and FP denote respectively the number of true positives and false positives that f_{Censor} makes on a test set. The algorithm makes a correct detection when it selects $\mathcal{H}_0$ while $\mathcal{H}_0$ is in fact true, and it makes a false positive when it selects $\mathcal{H}_0$ while $\mathcal{H}_1$ is the correct hypothesis. To ensure that TRW converges to a decision with high precision, we use the detection probability P_D and the false positive probability P_F to specify performance conditions with respect to user-selected values α and β. We desire that $P_F \leq \alpha$ and $P_D \geq \beta$, where typical values are $\alpha = 0.01$ and $\beta = 0.99$.

As shown in Figure 3, given a new observation Y_n, we compute the likelihood ratio Λ of the n events observed so far as $Y = (Y_1, ..., Y_n)$, where,

$$\Lambda(Y) = \frac{Pr[Y|\mathcal{H}_1]}{Pr[Y|\mathcal{H}_0]} = \prod_{i=1}^{n} \frac{Pr[Y_i|\mathcal{H}_1]}{Pr[Y_i|\mathcal{H}_0]} \quad (2)$$

After n events, if $\Lambda(Y) \leq \eta_0$, then TRW stops observing flows coming from **(IP)** and decides to block it. If $\Lambda(Y) \geq \eta_1$ then TRW converges to the decision that **(IP)** is currently benign. In this case, we add a new step to the original TRW (i.e., blue box in Figure 3). More precisely, although it is currently benign, we perform a history-aware reset of the ratio to keep observing for possible Geneva attempts. For instance, if **(IP)** had previous history of Geneva detection

(i.e., $hist > 0$), then $\Lambda(Y)$ is reset to a value that considers the number of previous detections ($hist$) and the number of previous benign observations ($n - hist$). Formally, $\Lambda(Y) \leftarrow \frac{hist}{n}.\eta_0 + (1 - \frac{hist}{n}).\eta_1$, otherwise (if $hist = 0$) we reset it to its initial value and continue observing future flows of **(IP)** to detect potential future Geneva traffic. Finally, if $\eta_0 < \Lambda(Y) < \eta_1$, then a decision cannot be made yet. Thus, we wait for the next observation and update $\Lambda(Y)$. The thresholds η_1 and η_0 should be chosen such that the false alarm and detection probability conditions are satisfied. As explained in the TRW paper [39], we can choose $\eta_1 = \frac{\beta}{\alpha}$ and $\eta_0 = \frac{1-\beta}{1-\alpha}$.

Using the proposed ML-TRW-based detection approach we were able to detect Geneva probes (training) after it tests only 2 strategies of the first generation. This result is consistent across all simulated censors and China's Great Firewall. Considering that Geneva can derive previous manipulation strategies proposed in other censor probing-based tools (e.g. INTANG, liberate) [13], we believe that our detection approach is adaptable to be performed against them. In particular, the ML model used for flow-level detection can be trained on the network traces of any other probing-based tool.

5 DeResistor System Design

We now expand on the problem formulation described in §3 and describe the components of both players engaged in the two-opponent arms-race. As illustrated in Figure 4, Player 1 is an Internet user in a censored regime that aims to generate censorship evasion strategies by running a strategy generator. Player 2 is composed of a real-word censor and a ML model $f_{\text{DeResistor}}$ that we locally train to detect Player 1 probes. $f_{\text{DeResistor}}$ is intended to expose Player 1 to real-time detection threat to enable more detection-resilient strategy generation through adversarial training. It can be trained similarly to what was illustrated earlier in §4. We continue to adopt Geneva's genetic algorithm to play the role of strategy generation within our design. We design DeResistor with the following goals/requirements in mind:

• **Detection-Resilience**: DeResistor has to remain stealthy against a censor-side evasion detection.

• **Balance Strategy Fitness and Detection-Resilience**: DeResistor has to find an optimal trade-off between evasion strategy fitness and vulnerability to censor-side detection.

• **Leverage Background Traffic on Demand**: DeResistor has to avoid early detection and blocking by utilizing normal background traffic as a cover.

5.1 Two-Objective Fitness Function

DeResistor improves the genetic training using a two-objective fitness function to encourage detection-resilience (Figure 4). Formally, we define it as:

$$fitness(s) = a \cdot G(s) - b \cdot P(s), \tag{3}$$

where s denotes a manipulation strategy. The first term of the fitness function $G(s)$ computes the evasion strategy's effectiveness based on the censor feedback from running s, while the second term $P(s)$ computes the probability that s is detectable, returned by $f_{\text{DeResistor}}$. The fitness function is composed of two conflicting objectives, hence, the negative sign to the second term since we aim to maximize the strategy effectiveness while minimizing detectability. DeResistor aims to find an optimum trade-off between both objectives guided by two parameters a and b, where $a + b = 1$. Each parameter represents the weight of each objective in the fitness function, according to the user preferences. For instance, if a user is not interested in hiding its packet manipulation attempts (e.g., if the detection threat is minor), the parameters can be tuned accordingly (i.e., $a > b$). On the other hand, using a very large value of b compared to a is not recommended as it may not lead to any working strategies. Given a set of strategies $\{s_1, ..., s_n\}$, DeResistor runs each one against the censor to compute their fitness $G(s_i)$, it allows only some of the fittest to survive, and mutates or crosses over the surviving ones to generate new individuals for the next generation. In the setting where the genetic algorithm is Geneva, we keep the same genetic building blocks (i.e., $s = (trigger, actiontree)$), the same mutation and crossover functions, and the same strategy selection approach as defined in the original work.

5.2 Normal Background Traffic

Due to the two-objective fitness function, DeResistor can eventually learn to produce detection-resilient strategies. However, early in the training (e.g., generation 0), the generated strategies are expected to be very similar to Geneva's, which makes them likely to be detected by the censor-side detector f_{Censor}. Furthermore, our IP-level detection approach (i.e., ML+TRW) is able to block a source IP running Geneva, with high confidence, as soon as it evaluates 2 strategies against the censor, including the canary (i.e., empty) strategy, which is not enough to produce fitter future generations. In order to avoid early IP-blocking by the TRW due to early detection, DeResistor takes advantage of the seamless background benign traffic naturally produced by the client while surfing the Internet to connect to uncensored URLs (e.g., using a browser). Particularly, if s is detected (i.e., $P(s) > 0.5$), then DeResistor pauses the censor probing (in this case Geneva training) and switches to normal traffic (Figure 4). Until exactly J normal network flows have been produced, the genetic algorithm remains on standby. Otherwise, if s was not detected, then DeResistor allows Geneva to continue its genetic evolution normally (i.e., without pausing). We recall that the TRW ratio $\Lambda(Y)$ gets closer to the threshold η_1 if a benign flow x_n is observed, while it gets closer to η_0 (i.e., closer to block the IP source) if a Geneva flow is observed. Thus, a number J of benign observations received from the same IP address would increase $\Lambda(Y)$ and make it farther from η_0, which would delay detection and give more room to attempt other strategies. We note that, for automation and experimental purposes, DeResistor can be configured to automatically

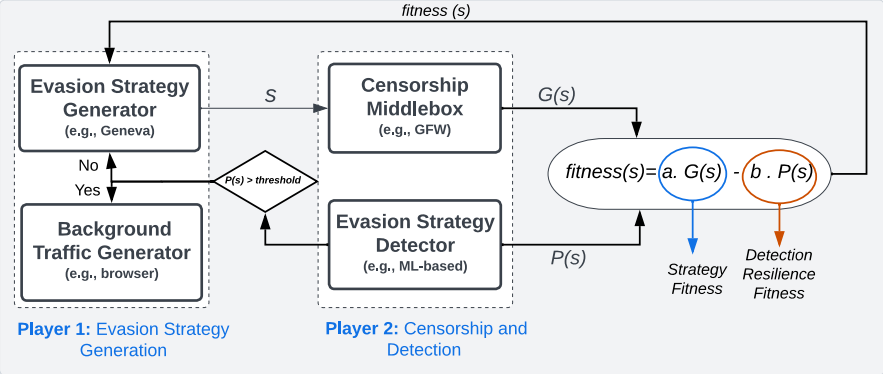

Figure 4: DeResistor System Overview. s: a manipulation strategy, $P(s)$: Probability that s is an evasion strategy.

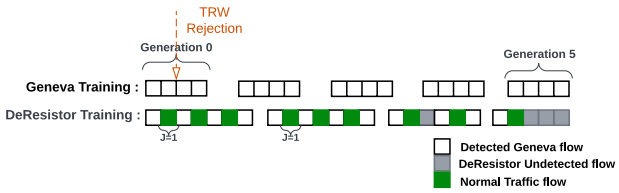

Figure 5: An illustration of Geneva genetic evolution traces when trained with DeResistor design vs. standalone training.

generate normal traffic on-demand through automated GET requests (e.g., `baidu.com` for GFW experiments) or using traffic generation tools, instead of relying on real traffic. For instance, for *in-situ* experiments against mock censors we leverage Harpoon [59], a flow-level traffic generator. In particular, when the censor-probing stops after a flow-level detection, Harpoon generates normal-like traffic between the client and a local uncensored server. While automating the background normal traffic is convenient to accelerate experiments, it is not recommended to use against a real-world censor since tools like harpoon can be tracked and detected.

In Figure 5, we illustrate an example that shows how guided pausing and blending with normal traffic in DeResistor traces can help avoid TRW detection. In particular, DeResistor runs the first strategy against the censor, that is likely to be detected (white boxes); if that is the case, then it pauses the probing and it returns after a jump (green box) of J normal flows is produced ($J = 1$ in this case). Next, it continues to alternate between detectable Geneva flows and normal flows until it reaches future generations, where the optimization algorithm gets closer to converging to a trade-off between the two objectives. This is shown using gray boxes for Generations 4 and 5 where DeResistor produces undetectable strategies.

Does the switch to normal traffic affect DeResistor training? As shown in Figure 4, Geneva and the background normal traffic modules are two separate processes. Consequently, no fitness calculation is performed during the jump J of normal traffic. As a result, it has no impact on the training process of Geneva. Furthermore, the normal traffic is produced through normal network activity performed by the user when Geneva is paused.

How to set the jump size J? Intuitively, if J is high, more normal traffic will be observed from the source IP running DeResistor, which would further confuse the ML-based evasion strategy detector. However, sending multiple consecutive benign flows might trigger a ratio reset in the TRW algorithm. We recall that, if $\Lambda(Y)$ surpasses the threshold η_1, then an automatic history-aware reset is performed (Figure 3, $\Lambda(Y) \leftarrow \frac{hist}{n}.\eta_0 + (1 - \frac{hist}{n}).\eta_1$). As DeResistor deployers, it is in our interest to avoid triggering these resets. Given that the reset considers the history of flow-level detection, if *hist* is very high, then we might risk a ratio reset that is much closer to η_0 than to η_1. Consequently, switching back to Geneva's evasion attempts might further decrease the ratio to make it less than η_0, which will cause an IP-level detection, hence, IP blocking. In conclusion, our operational observations suggest starting with $J = 1$ and then progressively consider increasing it based on detection results. In §7, we further discuss the impact of J parameter tuning when the TRW is set to perform more aggressive detection.

6 System Evaluation

Our evaluation of DeResistor is guided by these questions:
RQ1: Is DeResistor effective at making Geneva's probing traffic more resilient to detection?
RQ2: Are strategies generated by DeResistor effective at evading real-world censors?
RQ3: How effective is our approach in mitigating the detectability of packet manipulation features?

6.1 Experimental Setup

For a fair comparison, we run DeResistor and Geneva in the same experimental environment. To the extent possible, we make our setup as close as possible to what was proposed in the Geneva paper [13]. For our real-world evaluation, we thus specifically focus on GFW's HTTP censorship, India and Kazakhstan. We run our experiments from three vantage points in China (Qingdao, Beijing, and Shanghai), one vantage point in India (Bangalore) and one vantage point in Kazakhstan (Oral) to evaluate DeResistor effectiveness in

generating successful evasion strategies compared to Geneva. In each vantage point we run Geneva and DeResistor three times independently and collect the fittest strategies. We run each training session with a starting population pool of 500 individuals, capped at 20 generations. Our experiments showed that the adopted parameter-tuning is sufficient to successfully train both tools against all the studied censors. Additionally, we evaluate their detection-resilience against the real-time detection system proposed in §4.

Deployment Alternatives: We envision three usage models for DeResistor: (1) *One-shot Strategy Generation* where each client uses DeResistor to generate a set of strategies and use them to access censored sites. When all strategies are exhausted due to censor adaptation, users re-run DeResistor to generate new strategies. (2) *Strategy-as-a-Service* where DeResistor probes are performed by distributed nodes owned by a centralized service. Users then query this service on demand to obtain new strategies. This has the disadvantage that access to the service itself could be blocked. (3) *P2P Strategy Generation* where DeResistor nodes that are in-country (on one side of the firewall) coordinate to explore, generate, and share working Geneva strategies. In our experiments, we use (1) where multiple effective strategies are generated in the censor-probing phase that are then used to connect to multiple censored websites.

***In-Situ* Validation:** Like Geneva validation experiments adopted in [13], we first perform a Dockerized evaluation for DeResistor to train against 11 mock censors proposed by Geneva authors. We run each strategy in an isolated environment with four containers (a client, a mock censor, a forbidden server, and a legitimate server). The legitimate server container is added to be engaged in a simulation of background normal traffic created by Harpoon [59] (defined in §5) to be used by DeResistor when the training is triggered to pause. Harpoon is set to continuously send TCP packets from the client container to the legitimate server container using the same source IP engaged in the training-time evaluation of the generated strategies against the censor. Despite the reality that the mock censors are a bit outdated, we found them still relevant to compare DeResistor results with Geneva, especially for the detection resilience experiments. Furthermore, they are designed to mimic some specific aspects of nation-state censor behavior (e.g., China, India) [13].

Real-World Evaluation: Against the arguably most advanced Internet censorship apparatus, the Great Firewall (GFW) of China, we perform a post-training evaluation on the collected strategies by testing each strategy 30 times in each vantage point (Table 2). The GFW injects RST packets if a censored word is included in the URL of an HTTP GET request. Additionally, it performs residual censorship where it forbids new connections between client's IP address and the website's IP:port pair for approximately 90 seconds [63]. To avoid residual censorship, we use a different destination port after each test of the same censored website. Some websites are also censored by DNS (e.g., `google.com`, `wikipedia.org`). In particular, the GFW performs DNS poisoning to point clients that attempt to connect to these websites to a fake IP [36], which makes the intended destinations unreachable. One way to circumvent DNS poisoning is to first figure out the correct IP of the destination website, and then point the domain to its correct hosting IP (e.g., editing `/etc/host` in Linux). After sidestepping DNS poisoning, we are able to run Geneva and DeResistor to connect to websites like `wikipedia.org` and human rights watch (`hrw.org`). Websites like `google.com` that are additionally censored by null-routing in China are excluded from our experiments. Particularly, even if we figure out the correct hosting IP addresses of sites such as `google.com`, GFW drops any connection attempts to them. Therefore, packet-manipulation-based circumvention tools like Geneva cannot try evasion strategies since no response is received to the First SYN packet. Additionally, we run the same experiments in India using censored URLs like `bannedthought.net, xnxx.com, vidwatch.me` and Kazakhstan (e.g., `youporn.com`).

Detection Resilience: Our experiments also focus on evaluating the detection-resilience of DeResistor. For each run against GFW, India, Kazakhstan or the mock censors we also run the proposed real-time Geneva detection system (i.e., ML+TRW with history-aware reset) in parallel to monitor the entire training process and block the client if detected running censor-probing with 99% confidence. Particularly, we use $\alpha = 0.99$ as detection confidence and $\beta = 0.01$ as allowed detection error. We balance between the strategy success and the detection resilience objectives by running DeResistor with $a = 0.5$, $b = 0.5$. As for the jump-size we start with $J = 1$. An IP-level detection-resilience is observed if DeResistor finishes its training without getting IP-blocked by the TRW-based detector. A flow-level detection-resilience is observed if the detection rate returned by f_{Censor} decreases as DeResistor reaches fitter strategies with respect to the adopted two-objective fitness function in Equation 3.

6.2 Detection-Resilience Results

A strategy evaluation against the censor is tracked by a network flow between the client and the server. We recall that, our detection approach uses a ML model that performs a preliminary flow-level detection (f_{Censor}). Each flow detection is marked as another observation for the TRW algorithm to make a more confident IP-level detection. Figure 6 shows generations (x-axis) vs. flow-level detection rate trend (y-axis) during Geneva and DeResistor training against GFW. Furthermore, in Table 1, we report the IP-level detection results recorded by the TRW on Geneva and DeResistor when trained against different censors.

Flow-Level Detection Resilience: Figure 6 confirms that DeResistor traces are way less detectable compared to Geneva. In particular, as DeResistor training advances, the detection rate continues to drop until it reaches 45.06% after 5 genera-

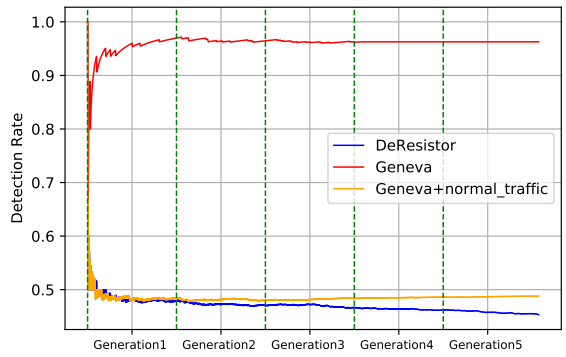

Figure 6: Flow-level detection rate evolution during Geneva and DeResistor training against China's GFW. We consider the 5 first generations.

tions while it stays very high (96.27%) during Geneva training. The immediate drop observed at the beginning of generation 1 (i.e., $1 \rightarrow 0.5$) is caused by the guided pausing of the genetic algorithm training when a flow is detected that permits the client to engage with a number J of normal benign flows. The same drop is observed when we run Geneva with guided pauses for normal traffic injection (without multi-objective optimization). However, due to the proposed two-objective fitness function, the observed decrease in the detection rate from generation 0 to generation 5 shows that DeResistor is learning to generate less detectable strategies as it advances to higher generations, compared to "Geneva+normal traffic".

To further explain our findings, we investigate the features of DeResistor traces compared to Geneva traces and normal traffic in Figure 7. Overall, we observe that the feature values density of DeResistor (green curve) are closer to Normal traffic (blue curve) compared to Geneva (red curve). Furthermore, looking into the data-points of each traffic type (gray scatter plots), it seems that DeResistor traces exhibit less overlapping TCP segments, less corrupt data-offset fields, and less corrupt SYN packets compared to Geneva. We conclude that DeResistor is able to tone down detectable features exhibited by Geneva, which leads to lower flow-level detection rate (answers **RQ3**). However, we acknowledge that DeResistor traces are still different from normal traces, which is natural considering that its main objective is still to manipulate packets and evade censorship. It is noteworthy that DeResistor can be even less detectable if the user chooses to give advantage to the detection-resilience objective at the expense of the strategy success objective (e.g., a = 0.3, b = 0.7). Inline with the GFW results, Table 1 (3^{rd} column) shows that DeResistor was able to reduce the flow-level detection rate (99.5% $\rightarrow$ 34.93% and 49.22%) respectively against India and Kazakhstan. Additionally, we observe similar decrease (99.4% $\rightarrow \approx$ 32%) when trained against the 11 mock censors.

IP-Level Detection Resilience: As reported before, using the proposed detection approach we were able to detect with high confidence a source IP address running Geneva

Censors	**IP-Level Detection** (Geneva→DeResistor)	**Flow-Level Detection** (Geneva→DeResistor)	**Jump Size** J
China's GFW	Detected after 2 flows → **Undetected**	96.27% → **45.06%**	1
India	Detected after 2 flows → **Undetected**	99.50% → **34.93%**	1
Kazakhstan	Detected after 2 flows → **Undetected**	99.50% → **49.22%**	1
Censor 1-4,7,9	Detected after 2 flows → **Undetected**	99.4% → **32.46%**	1
Censor 5,10.	Detected after 2 flows → **Undetected**	99.4% → **31.21%**	1
Censor 6	Detected after 2 flows → **Undetected**	99.4% → **34.05**	1
Censor 8	Detected after 2 flows → **Undetected**	99.4%→ **30.93%**	1
Censor 11	Detected after 2 flows → **Undetected**	99.4%→ **29.91%**	1

Table 1: Geneva vs. DeResistor detection results using history-aware TRW. Details about mock censors can be found in [13].

after the TRW receives only 2 observations (i.e., 2 Geneva flows/probes). According to results reported in Table 1, we observe that, against all the studied censors, DeResistor was able to complete its training without being rejected by TRW (2^{nd} column), which answers **RQ1**. A jump size $J = 1$ was sufficient to reach these results (4^{th} column). Particularly, as we illustrated in Figure 5, after each flow-level detection of a strategy test, the injected normal flow restores the likelihood ratio $\Lambda(Y)$ of the TRW to its initial value. This pattern encapsulates the $\Lambda(Y)$ between η_0 and η_1, which makes it longer for the TRW to converge to a decision. Setting the jump size to $J = 1$ is not only sufficient to avoid detection, but also recommended to avoid triggering TRW resets. We recall that, our implementation of TRW algorithm is powered by a history-aware reset of the likelihood ratio in case the TRW is converging to a decision that the source IP is benign. Thus, it is in favor of the client to avoid pushing the TRW to make a reset that considers all previous detection of packet manipulations. More precisely, if we use higher jump size $J > 1$, DeResistor would pause Geneva training until the client has engaged in normal traffic of $J > 1$ flows, which might cause an undesired reset that might lead to IP-level detection. Yet, we note that our observations reveal that clients running DeResistor are safe to use even higher values of J.

6.3 Censorship Evasion Results

To explore whether Geneva can still generate evasion strategies with high success rate when trained within the DeResistor design, we report in Table 2 the success rate of representative strategies generated by Geneva and DeResistor. Furthermore, we manually analyze composed strategies that have multiple actions. In particular, we removed individual actions and verified whether the strategy is no longer successful. Additionally, we ensure that a strategy success (i.e., offers access to a censored website) is not by accident or due to a malfunction of the censor-side. Hence, for each highly successful strategy, we manually examine the resulting network traces, to verify

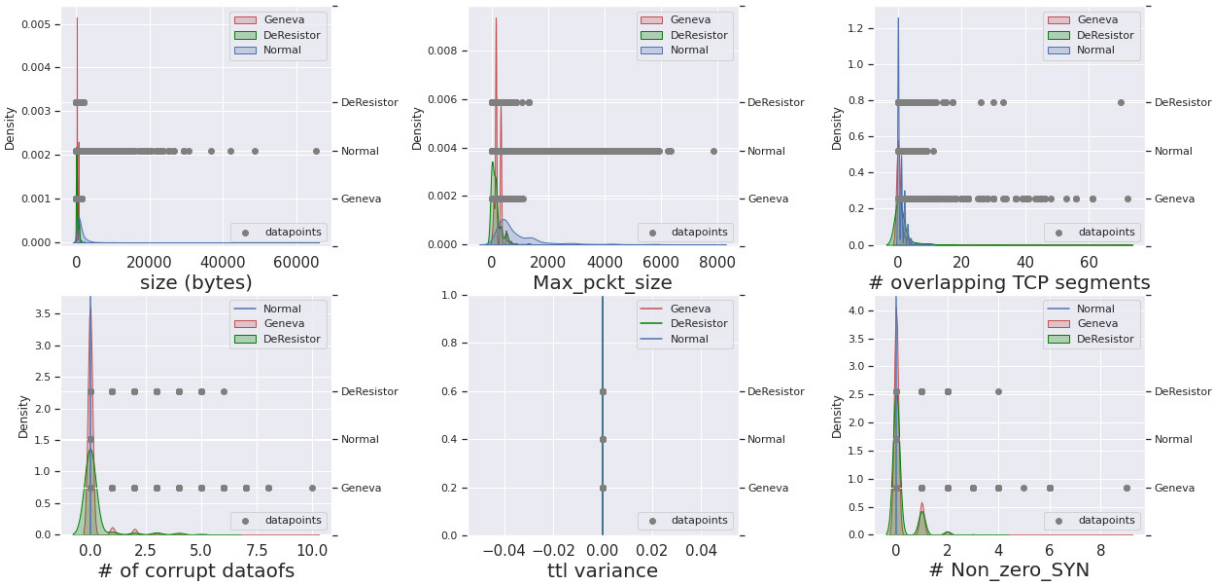

Figure 7: DeResistor training feature analysis. The scatter plot displays data points for Geneva, DeResistor, and normal traffic. The curves show the data distribution density of each traffic type.

Description	**Strategy Genetic Code**	**Generated By**		**China**				**India**	**Kazakhstan**
		Geneva	DeResistor	Qingdao	Shanghai	Beijing	Overall	Bangalore	Oral
Corrupt timestamp and fake HTTP Request	[TCP:dataofs:8]-tamper{TCP:options-timestamp:corrupt} (tamper{TCP:load:replace:__HTTP_REQUEST__},)-|		✓	95.83%	100%	100%	98.61%	0%	0%
Fake HTTP request.	[TCP:dport:80]-tamper {TCP:load:replace:__HTTP_REQUEST__}-|	✓	✓	100%	100%	84.21%	94.74%	0%	0%
TCB Desync. with invalid dataofs and corrupt checksum.	[TCP:flags:PA]-duplicate (tamper{TCP:dataofs:replace:10} (tamper{TCP:chksum:corrupt},),)-|	✓		90%	100%	86.67%	92.23%	0%	100%
Fake DNS request and incorrect Ack.	[TCP:dport:80:4]-tamper{TCP:load:replace:__DNS_REQUEST__} (tamper{TCP:ack:replace:1621145327},)-|	✓		50%	75%	47.61%	57.54%	0%	100%
Invalid Payload and corrupt options.	[TCP:reserved:0]-tamper{TCP:load:corrupt} (tamper{TCP:options-sack:corrupt},)-|	✓		38.09%	52.38%	77.78%	56.08%	0%	63.33%
Invalid Payload and corrupt options.	[TCP:reserved:0]-tamper{TCP:load:corrupt}(tamper{TCP:options-altchksum:replace:131},)-|	✓	✓	36.36%	56%	50%	47.45%	0%	63.33%
Invalid Payload and corrupt urgptr.	[TCP:options-mss:]-tamper{TCP:load:corrupt} (tamper{TCP:urgptr:corrupt},)-|		✓	52.17%	48%	42.11%	47.43%	45%	56.67%
Invalid Payload.	[TCP:flags:PA]-tamper{TCP:load:corrupt}-|	✓	✓	52.17%	39.1%	47.8%	46.36%	0%	66.67%
Invalid Payload and incorrect reserved.	[TCP:reserved:0]-tamper{TCP:load:corrupt} (tamper{TCP:reserved:replace:7},)-|	✓		52.63%	41.67%	41.66%	45.32%	30%	63.33%
Incorrect dataofs and segmentation.	[TCP:reserved:0:3]-tamper{TCP:dataofs:replace:10}(fragment{tcp:-1:False},)-|	✓		80%	31.03%	24.13%	45.05%	0%	100%
Segmentation	[TCP:flags:PA]-fragment{ip:20:False:6}-|		✓	0%	0%	0%	0%	100%	0%
TCB Desync. with stutter Req.	[TCP:flags:PA]-duplicate(tamper{IP:len:replace:64},)-|	✓	✓	0%	0%	0%	0%	100%	0%
Segmentation and corrupt dataofs	[TCP:options-wscale::3]-fragment{ip:-1:True:5} (fragment{ip:4:True:43}, tamper{TCP:dataofs:replace:14})-|		✓	0%	0%	0%	0%	100%	100%

Table 2: A summary of top strategies ([Trigger](actions)) generated by Geneva and DeResistor and their post-training success rates. We run every strategy 30 times from each vantage point (VP). Strategies in rows 1-10 are produced in China's VPs, rows 11-12 are produced in India and the final row is produced in Kazakhstan.

that every action of the strategy is actually performed on triggered packets. We emphasize that these manual efforts are performed only for the sake of conveying accurate results in the paper and it is not a main component of DeResistor.

Strategies and Success Rates: Table 2 shows that DeResistor is able to automatically produce strategies with high success rate in China (i.e., up to 98.61% in 1^{st} row), India (100% in rows 11-12) and Kazakhstan (100% in final row) (addresses

RQ2). Although DeResistor's search space is narrowed to balance strategy effectiveness with detection-resilience, we observe that the genetic algorithm is still successful in finding multiple optimum solutions that satisfy the trade-off. Hence, in addition to its detection resilience, DeResistor does not sacrifice the success rate. However, we note that DeResistor takes longer to find strategies with high fitness values. Since Geneva does not punish detectability, it can start producing fit strategies within the 4^{th} generation, while DeResistor starts to effectively find the evasion/detection-resilience trade-off after at least 8 generations. Additionally, by design, Geneva should be able to produce a higher number of fit strategies given the difference in the fitness complexity.

Difference Between Geneva and DeResistor Strategies: We examine DeResistor's fit strategies to understand why they are more resilient to detection. We find that most of them do not include actions like tampering with dataofs, TCP segments, or tampering with the TTL. Furthermore, unlike Geneva, DeResistor strategies do not utilize invalid flags (e.g., FRAPUEN, FRAPUEN) for connection's state tear-down (Table 2), which confirms our previous observations in Figure 7. As explained before, DeResistor exhibits less detectable features than Geneva. These findings validate our intuition that DeResistor enables Geneva to learn from the detection feedback of $f_{\text{DeResistor}}$ to produce more detection-resilient strategies.

Success Rate Across Different Vantage Points: Over 30 runs, Table 2 shows that a strategy success rate against GFW can slightly differ if performed from different vantage points from within China. In some cases, we observe that an effective strategy in one vantage point might not be effective in another one. For instance, row 10 in Table 2 includes a strategy that is highly effective in Qingdao (80%) but has a poor evasion performance in Shanghai (31.03%) and Beijing (24.13%). This is also observable in row 5 (77.78% in Beijing and 38.09% in Qingdao). Although prior works have shown that GFW is managed in a centralized manner [20, 28, 36, 56, 71], it is a distributed system consisting of many filtering middleboxes located across different autonomous systems. The difference in successful evasion rates therefore could have been caused by several reasons, including different filtering middleboxes being under/over-loaded [70], or additional filtering policies applied by regional ISPs [18, 67]. Our findings underscore the importance of training and validating the effectiveness of censorship evasion strategies across different network locations to obtain a non-biased overall view of their performance. From Table 2, we confirm that strategies that work in China do not necessarily work in India and vice versa. Some strategies that are generated in China can also evade censorship in Kazakhstan (rows 3-10).

7 Discussion

We next discuss how adaptive censors may attempt to detect DeResistor-generated strategies. We also further explain how our work can be adapted to numerous Geneva-like systems.

7.1 Adaptive Censors and Countermeasures

Training the censor-side model on DeResistor traces: In §4, we described a real-time detection approach that includes training ML models on Geneva traces. One way for the censor to turn around DeResistor detection-resilience is to train f_{Censor} on DeResistor traces instead, after excluding possible normal flows injected by DeResistor. As a result, f_{Censor} might be able to detect DeResistor flows and guide the TRW algorithm to a confident IP-level detection.

On the user's side, DeResistor can also be trained with a local ML detector $f_{\text{DeResistor}}$ that is retrained on the previous cycle of DeResistor traces. Hence, this will further improve the flow-level detection resilience of newly generated strategies. To explore to what extent DeResistor can stand against an improved censor-side ML detector, we perform multiple cycles of the game DeResistor vs. Censor. Results are displayed in Table 3. In the first cycle of the game (1), the censor trains a flow-level ML detector f_{Censor} on Geneva traces to be used for TRW's IP-level detection and DeResistor is trained using a local flow-level detector $f_{\text{DeResistor}}$ also trained on Geneva traces. As we revealed in §6, DeResistor wins cycle (1). In (2), the censor updates its f_{Censor} by extending its training data with DeResistor traces in addition to Geneva's. We assume that no changes are made on DeResistor training. As a result, f_{Censor} becomes more accurate than $f_{\text{DeResistor}}$ which leads to a less accurate detection feedback for DeResistor. Thus, the censor wins cycle (2) (DeResistor is detected "✓"). In (3), DeResistor allows normal background traffic after *every* censor-probing flow *even if it is not detected by* $f_{DeResistor}$. This is intended to mitigate the impact of the flow-level detection gap between $f_{\text{DeResistor}}$ and f_{Censor} observed in (2) by introducing periodic pauses independently from $f_{\text{DeResistor}}$ feedback. Furthermore, $f_{\text{DeResistor}}$ can also be updated to detect previous DeResistor traces and better simulate f_{Censor} for more accurate detection-resilience feedback. The third row in Table 3 shows that DeResistor regains the upper hand and avoids detection on the censor-side. Finally, in (4), we assume that the censor performs a new update on its model f_{Censor} while DeResistor keeps the configuration of (3). We observe that DeResistor is still able to avoid IP-level detection in (4). Two explanations stand out: first, this time around, f_{Censor} and $f_{\text{DeResistor}}$ have comparable flow-level detection rates ($38.97\% \approx 39.45\%$), despite the additional update of f_{Censor}. Second, the regular pauses after every censor-probing flow introduced in (3) mitigates the impact of any possible discrepancy between $f_{\text{DeResistor}}$ detection feedback and the real flow-level detection recorded by f_{Censor}. We note that our findings show that the introduced updates on DeResistor in (3) do not sacrifice its ability to generate highly fit strategies for censorship evasion against GFW.

Filtering out the background normal traffic: We assume

	Flow-Level Detection ($J=1$)		IP-Level Detection (TRW)		
	$f_{\text{DeResistor}}$	f_{Censor}	$J=1$	$J=2$	$J=3$
①	45.01%	45.06%	✗	✗	✗
②	45.01%	48.84%	✓	✓	✓
③	38.95%	39.05%	✗	✗	✗
④	38.97%	39.45%	✗	✗	✗

			$\tilde{J}_{max} < J_{max}$	$\tilde{J}_{max} > J_{max}$	$\tilde{J}_{max} = J_{max}$
⑤	38.97%	39.45%	0/3	0/3	0/3

	LSTM IP-Level Detection Accuracy				
	$J < L$	$J = L = 10$)	$J = 15$ ($J > L$)	$J = 25$ ($J > L$)	$J >> L$
⑥	99.63%	71.24%	37.89%	19.93%	$acc \rightarrow 0\%$

Table 3: Future cycles of the DeResistor vs Censor game. ✓ denotes *detected*, ✗ denotes *undetected*. **1** and **3** are DeResistor's moves, while **2** and **4** are censor's moves. In **5**, *(*/3)* denotes the number of detections within 3 runs. In **6**, L=10.

that the censor deploys a TRW that tries to filter out the background normal traffic to enable an IP-level detection of DeResistor. However, TRW is not designed to ignore all normal traffic. Otherwise, it will be biased against normal users. Particularly, all its flow-level observations $Y = \{y_1, .., y_n\}$ defined in 4.2 would only include flows that are detected as malicious. Hence, on the IP-level, the TRW would block innocent users based on potential erroneous detection of normal flows. Instead, the censor can attempt to approximate the jump size ($\tilde{J} \approx J$), based on previous traces of DeResistor. In DeResistor's side, we can make J more difficult to estimate by randomly changing its value after each censor-probing flow ($J \in \{1, .., J_{max}\}$). For a fair game, we assume that the censor also actively changes its approximation ($\tilde{J} \in \{1, .., \tilde{J}_{max}\}$), where $\tilde{J}_{max}$ can be the maximum jump recorded by the previous cycles of DeResistor. In experiment ⑤, We consider three cases: when the censor's approximation $\tilde{J}_{max}$ is lower than the actual value J_{max}, when it is higher and when they are equal. To reduce the impact of the added randomness on results, we perform each experiment 3 times. Table 3 shows that for all runs DeResistor was able to complete its training across multiple runs across the different experiments.

DeResistor vs. IP-level ML Detector: Apart from TRW, we explore whether DeResistor can confuse another IP-level detection approach. To that end, we train an LSTM [38] model to map a sequence of flow-level observations with length L to an IP-level label. LSTM (§B) is trained on different sequences of DeResistor traces that include a random number of jumps $J < L$ (more about how to fix L is provided in §B). Each sequence contains flow-level detection probabilities returned by f_{Censor} (i.e., $Y = \{P_1, .., P_n\}$). Similarly to the TRW, the goal is to reach a confident IP-level decision that a user is truly probing the censor. In experiment ⑥ (Table 3), our findings show that LSTM can detect Geneva and DeResistor sequences with 99.63% accuracy. The accuracy drops to 37.89% when we use $J > L$ and it continues to decrease closer to 0% if DeResistor further allows more background traffic between Geneva attempts. To improve the accuracy, the censor can try to increase L to make it higher than J again. However, this will always delay the IP-level decision until exactly L flow-level decisions are observed even if all of them are coming from Geneva (unlike TRW that automatically converges after a minimum of Geneva flows). Additionally, we noticed that a sequence with $L-1$ normal flows and only 1 flow of Geneva is always labeled by LSTM as malicious. This might result in blocking innocent users based on 1 potentially mistaken flow-level detection. To summarize, although LSTM is more capable to detect DeResistor, TRW is more suitable to reach a high confidence decision of IP-level blocking, since it does not rely on a fixed length of observations (i.e., L for LSTM). It automatically stops observing flow-level detection as soon as a high confidence decision is made.

Aggressive Detection: Another possible countermeasure is to tune the TRW algorithm to make more aggressive IP-level detection. This could be done either by decreasing the TRW's detection confidence (e.g., $\alpha = 0.8$, instead of $\alpha = 0.99$) or by performing detection-oriented ratio resets instead of history-aware resets (refer to Figure 3). We recall that the TRW automatically resets the likelihood ratio $\Lambda(Y)$ if a source IP is found to be benign (i.e., $\Lambda(Y) > \eta_1$). If we reset $\Lambda(Y)$ to a value that is very close to η_0 (e.g., $\Lambda(Y) \leftarrow \eta_0 + 0.1$), it will make the TRW biased toward making less confident decisions. The natural side-effect here is that aggressive resets are triggered after observing a sequence of normal network flows. Consequently, this countermeasure will affect many innocuous clients (i.e., IP addresses).

Misleading Detection Feedback: It is known that Geneva might fall short if the censor detects its probing phase. In particular, the censor could poison Geneva training by making strategies appear to (not) work [13]. For DeResistor the detection feedback is received from the local model $f_{\text{DeResistor}}$ ($\neq f_{Censor}$). Hence, while the censor can mislead censorship feedback it cannot do the same for detection feedback.

7.2 Extending DeResistor Beyond Geneva

Playing the role of a strategy generation module, Geneva is used as a foundational building block for DeResistor throughout this paper.

Our success in adopting a genetic algorithm like Geneva to the whole training process of DeResistor shows its potential adaptability to other similar evasion tools.

Genetic Algorithms: A genetic algorithm designed to generate evasion strategies by probing the censor is ideally suited for deployment within DeResistor to enhance detection resilience. Geneva is equipped with a server-side deployment [12], allowing censored clients to connect to a Geneva-supported server without having to discover client-side strategy. Instead, Geneva can be trained on the censored server using an external client (in a censored regime) that regularly attempts connections to the server during the training phase. As a probing-based method, the censor can implement our

ML+TRW approach to detect abnormal network behaviors from a censored server that is running server-side packet manipulations. Even when running on the server-side, DeResistor is still fit to protect Geneva training from detection. In particular, the two-objective fitness function can guide the training into punishing detection. Furthermore, DeResistor can perform guided pausing on Geneva training at the server-side. As a result, no server-side strategy is tested against the censor and the client connection attempts will be censored until J number of flows (normally censored flows) is performed.

Geneva has also been extended to generate application-layer evasion strategies [33]. Despite training on a different network layer, such strategies also involve repetitively probing a censor, thus being susceptible to detection. DeResistor therefore is also suitable to protect the training phase of such strategies regardless of what network layer is being targeted.

Symbolic Execution: SymTCP [64] is another packet-manipulation tool that automatically discovers censorship evasion strategies, which also involve repetitively probing a censor to check evasion effectiveness. In a similar fashion, SymTCP probing can be protected when being integrated with DeResistor. In particular, the symbolic execution can be guided to explore execution paths that are detection-resilient using the output of $f_{\text{DeResistor}}$ trained to detect SymTCP probes. The feedback of $f_{\text{DeResistor}}$ can either be embedded to the whole automatic process, or it might be useful for manual selections of the best input packet sequences. Additionally, the background traffic module can serve the same purpose of avoiding early detection by performing detection-guided pauses over the probing phase.

Guided Fuzz-Testing Tools: In addition to learning-based tools, other approaches (e.g., CenTrace [53], INTANG [63], and Lib•erate [42]) rely on fuzzing (brute force or guided fuzzing) to perform censorship measurement or reveal defects of middleboxes. These tools are also vulnerable to detection as the ML+TRW approach proposed in §4 can be used to detect network traces from these tools instead of Geneva. DeResistor can aid in the probing phase of these tools by making them more resilient to detection. In particular, the background normal traffic module can be leveraged for guided pausing of the probing phase while $f_{\text{DeResistor}}$ feedback can be used to thwart highly detectable probing strategies.

8 Ethical Considerations

Internet censorship, especially when it comes to state-sponsored network interference, is often motivated by political reasons [19, 21, 22, 32, 36]. Studies in this sensitive domain thus have to be performed in a responsible manner.

Human Subject Risk: Our work does not involve human subjects. Hence, there was no need to receive Institutional Review Board (IRB) approval. For both the mock censors and real-world experiments, we conducted all training and post-training evaluations from vantage points under our control. We made sure that our probing interactions with censors (mock or real-world) have no security and/or privacy impact on other users. We do so by running all experiments with devices isolated from real users/services and using special credentials that are not associated with any real individuals. The nodes under our control were operated in data centers at different locations inside China, India and Kazakhstan but not residential networks owned by any users. This setup is a common practice within the censorship measurement community. Our experiments therefore do not impose any risk on real users since we did not recruit or involve any human users.

On the surface, releasing DeResistor seems favorable for censors. However, the overall consensus from the anti-censorship community is that the evaders benefit more, outweighing what censors can gain. This is evident by numerous prior studies [13, 25, 42, 63] that have been released and incorporated into popular circumvention tools [1, 2, 44]. On balance, we believe DeResistor benefits anti-censorship efforts by equipping them with detection-resilient censor probing capabilities while also evolving their strategy generation schemes based on the detectability feedback from censors.

9 Conclusion

We have demonstrated that packet manipulation-based censorship evasion tools such as Geneva can be detected and blocked by censorship middleboxes. To enable resilient censorship evasion in the face of sophisticated censor-side detection, we designed DeResistor, as a detection-resilience extension for automated packet-manipulation-based strategy generation tools, such as Geneva. DeResistor factors in the likelihood of detectability of censorship strategies when it generates evasion strategies and guides Geneva's strategy generator to rely on less-detectable features. By doing so, it not only enables the generation of a successful censorship evasion strategy, but also shields Geneva from being detected by the censor. We evaluated DeResistor first using 11 mock censors and then against real-world censors, China's GFW, India and Kazakhstan. Our evaluations from multiple vantage points from within China, one vantage point in India and Kazakhstan suggest that DeResistor can evade censors with high success rate while remaining undetected. We also broadly discuss the prospect of potential countermeasures from the censor and applicability of DeResistor to other packet manipulation-based censorship evasion systems. DeResistor demonstrates a new frontier in making censorship evasion work for billions of users worldwide while minimizing the risk of being detected and blocked by a nation-state censor.

Acknowledgments

We thank the anonymous reviewers and our anonymous shepherd for their insightful feedback. We are grateful to Sadia Nourin, Kevin Bock, Dave Levin, activists at GreatFire.org, and others who preferred to remain anonymous for helpful discussions and suggestions. This work was supported in part by National Science Foundation (NSF) award CNS-2238084

and Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112190126. Approved for public release; distribution is unlimited.

References

[1] GoodbyeDPI - Deep Packet Inspection circumvention utility. https://github.com/ValdikSS/GoodbyeDPI.

[2] Psiphon VPN Packetman Library. https://pkg.go.dev/github.com/Psiphon-Labs/psiphon-tunnel-core/psiphon/common/packetman.

[3] Uyghur university student serving 13-year sentence for using VPN. https://www.rfa.org/english/news/uyghur/student-sentenced-06082023154805.html.

[4] Egypt keeps trying to block Signal, inadvertently blocking all of Google, and having to stop as a result. We'll also expand domain fronts., 2017. http://twitter.com/signalapp/status/817062093094604800.

[5] Giuseppe Aceto, Alessio Botta, Antonio Pescapè, M. Faheem Awan, Tahir Ahmad, and Saad Qaisar. Analyzing Internet censorship in Pakistan. In *Research and Technologies for Society and Industry*, 2016.

[6] Abderrahmen Amich, Birhanu Eshete, and Vinod Yegneswaran. Adversarial detection of censorship measurements. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society*, WPES'22, page 139–143, New York, NY, USA, 2022.

[7] Anonymous. The collateral damage of internet censorship by DNS injection. *Comput. Commun. Rev.*, 42(3):21–27, 2012.

[8] Anonymous, Arian Akhavan Niaki, Nguyen Phong Hoang, Phillipa Gill, and Amir Houmansadr. Triplet censors: Demystifying great firewall's DNS censorship behavior. In *Proceedings of the 10th USENIX Workshop on Free and Open Communications on the Internet*, FOCI '20. 2020.

[9] Simurgh Aryan, Homa Aryan, and J. Alex Halderman. Internet censorship in Iran: A first look. In *Free and Open Communications on the Internet*. USENIX, 2013.

[10] Kevin Bock, Yair Fax, and Dave Levin. Evading SNI Filtering in India with Geneva, 2021.

[11] Kevin Bock, Yair Fax, Kyle Reese, Jasraj Singh, and Dave Levin. Detecting and Evading Censorship-in-Depth: A Case Study of Iran's Protocol Filter. In *USENIX FOCI*, 2020.

[12] Kevin Bock, George Hughey, Louis-Henri Merino, Tania Arya, Daniel Liscinsky, Regina Pogosian, and Dave Levin. Come as you are: Helping unmodified clients bypass censorship with server-side evasion. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 586–598, New York, NY, USA, 2020. Association for Computing Machinery.

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

[14] Kevin Bock, Gabriel Naval, Kyle Reese, and Dave Levin. Even Censors Have a Backup: Examining China's Double HTTPS Censorship Middleboxes. *ACM FOCI*, 2021.

[15] Abdelberi Chaabane, Terence Chen, Mathieu Cunche, Emiliano De Cristofaro, Arik Friedman, and Mohamed Ali Kaafar. Censorship in the wild: Analyzing Internet filtering in Syria. In *Internet Measurement Conference*. ACM, 2014.

[16] Adriel Cheng. Pac-gan: Packet generation of network traffic using generative adversarial networks. *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0728–0734, 2019.

[17] Qiumei Cheng, Shiying Zhou, Yi Shen, Dezhang Kong, and Chunming Wu. Packet-level adversarial network traffic crafting using sequence generative adversarial networks, 2021.

[18] Jedidiah R. Crandall, Daniel Zinn, Michael Byrd, Earl Barr, and Rich East. ConceptDoppler: A Weather Tracker for Internet Censorship. In *ACM CCS '07*.

[19] M. Crete-Nishihata, R. Deibert, and A. Senft. Not by Technical Means Alone: The Multidisciplinary Challenge of Studying Information Controls. *IEEE Internet Computing*, 17(3):34–41, May 2013.

[20] Ronald Deibert. China's Cyberspace Control Strategy: An Overview and Consideration of Issues for Canadian Policy, 2010.

[21] Ronald Deibert. Reset: Reclaiming The Internet for Civil Society, 2020.

[22] Ronald J. Deibert. Dark Guests and Great Firewalls: The Internet and Chinese Security Policy. *Journal of Social Issues*, 58:143–159, 2002.

[23] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–319, August 2004.

[24] H. Duan, N. Weaver, Z. Zhao, M. Hu, J. Liang, J. Jiang, K. Li, and V. Paxson. Hold-On: Protecting Against On-Path DNS Poisoning. In *Securing and Trusting Internet Names*, 2012.

[25] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies*, 2015(2):46–64, 2015.

[26] Arturo Filastò and Jacob Appelbaum. OONI: Open observatory of network interference. In *Free and Open Communications on the Internet*. USENIX, 2012.

[27] Genevieve Gebhart, Anonymous Author, and Tadayoshi Kohno. Internet censorship in Thailand: User practices and potential threats. In *European Symposium on Security & Privacy*. IEEE, 2017.

[28] Geremie R. Barme And Sang Ye. The Great Firewall of China, 1997-06-01. https://www.wired.com/1997/06/china-3/.

[29] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[30] Devashish Gosain, Anshika Agarwal, Sambuddho Chakravarty, and H. B. Acharya. The devil's in the details: Placing decoy routers in the Internet. In *ACSAC*. ACM, 2017.

[31] Luchao Han, Yiqiang Sheng, and Xuewen Zeng. A packet-length-adjustable attention model based on bytes embedding using flow-wgan for smart cybersecurity. *IEEE Access*, 7:82913–82926, 2019.

[32] Seth Hardy, Masashi Crete-Nishihata, Katharine Kleemola, Adam Senft, Byron Sonne, Greg Wiseman, Phillipa Gill, and Ronald J. Deibert. Targeted Threat Index: Characterizing and Quantifying Politically-Motivated Targeted Malware. In *23rd USENIX Security Symposium*, pages 527–541. USENIX, 2014.

[33] Michael Harrity, Kevin Bock, Frederick Sell, and Dave Levin. GET /out: Automated discovery of Application-Layer censorship evasion strategies. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 465–483, Boston, MA, August 2022. USENIX Association.

[34] Nguyen Phong Hoang, Sadie Doreen, and Michalis Polychronakis. Measuring I2P Censorship at a Global Scale. In *9th USENIX Workshop on Free and Open Communications on the Internet (FOCI 19)*, Santa Clara, CA, 2019. USENIX Association.

[35] Nguyen Phong Hoang, Panagiotis Kintis, Manos Antonakakis, and M. Polychronakis. An Empirical Study of the I2P Anonymity Network and its Censorship Resistance. In *ACM IMC '18*.

[36] Nguyen Phong Hoang, Arian Akhavan Niaki, Jakub Dalek, Jeffrey Knockel, Pellaeon Lin, Bill Marczak, Masashi Crete-Nishihata, Phillipa Gill, and Michalis Polychronakis. How Great is the Great Firewall? Measuring China's DNS Censorship. In *USENIX Security Symposium*, 2021.

[37] Nguyen Phong Hoang, Arian Akhavan Niaki, Phillipa Gill, and Michalis Polychronakis. Domain Name Encryption Is Not Enough: Privacy Leakage via IP-based Website Fingerprinting. In *Proceedings of the 21st Privacy Enhancing Technologies Symposium*, PoPETs '21, 2021.

[38] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[39] Jaeyeon Jung, V. Paxson, A.W. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 211–225, 2004.

[40] Sheharbano Khattak, Mobin Javed, Philip D. Anderson, and Vern Paxson. Towards illuminating a censorship monitor's model to facilitate evasion. In *Free and Open Communications on the Internet*. USENIX, 2013.

[41] Gary King, Jennifer Pan, and Margaret E. Roberts. Reverse-engineering censorship in China: Randomized experimentation and participant observation. *Science*, 345(6199), 2014.

[42] Fangfan Li, Abbas Razaghpanah, Arash Molavi Kakhki, Arian Akhavan Niaki, David Choffnes, Phillipa Gill, and Alan Mislove. Lib•erate, (n): A library for exposing (traffic-classification) rules and avoiding them efficiently. In *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, page 128–141, New York, NY, USA, 2017. Association for Computing Machinery.

[43] Graham Lowe, Patrick Winters, and Michael L. Marcus. The great DNS wall of China. Technical report, New York University, 2007.

[44] M. Marlinspike. Doodles, stickers, and censorship circumvention for Signal Android., 2017. https://signal.org/blog/doodles-stickers-censorship.

[45] Soo-Jin Moon, Jeffrey Helt, Yifei Yuan, Yves Bieri, Sujata Banerjee, Vyas Sekar, Wenfei Wu, Mihalis Yannakakis, and Ying Zhang. Alembic: Automated model inference for stateful network functions. In *16th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2019, Boston, MA, February 26-28, 2019*, pages 699–718. USENIX Association, 2019.

[46] Zubair Nabi. The anatomy of web censorship in Pakistan. In *Free and Open Communications on the Internet*. USENIX, 2013.

[47] Arian Akhavan Niaki, Shinyoung Cho, Zachary Weinberg, Nguyen Phong Hoang, Abbas Razaghpanah, Nicolas Christin, and Phillipa Gill. ICLab: A global, longitudinal internet censorship measurement platform. In *Symposium on Security & Privacy*. IEEE, 2020.

[48] Daiyuu Nobori and Yasushi Shinjo. VPN Gate: A Volunteer-Organized Public VPN Relay System with Blocking Resistance for Bypassing Government Censorship Firewalls. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14)*, pages 229–241, 2014.

[49] Sadia Nourin, Van Tran, Xi Jiang, Kevin Bock, Nick Feamster, Nguyen Phong Hoang, and Dave Levin. Measuring and Evading Turkmenistan's Internet Censorship. In *The International World Wide Web Conference*, WWW '23, 2023.

[50] Jong Chun Park and Jedidiah R. Crandall. Empirical study of a national-scale distributed intrusion detection system: Backbone-level filtering of HTML responses in China. In *Distributed Computing Systems*, pages 315–326. IEEE, 2010.

[51] P. Pearce, Ben Jones, F. Li, Roya Ensafi, N. Feamster, N. Weaver, and V. Paxson. Global Measurement of DNS Manipulation. In *USENIX Security Symposium*, 2017.

[52] Ram Sundara Raman, Prerana Shenoy, Katharina Kohls, and Roya Ensafi. Censored Planet: An Internet-wide, longitudinal censorship observatory. In *Computer and Communications Security*. ACM, 2020.

[53] Ram Sundara Raman, Mona Wang, Jakub Dalek, Jonathan Mayer, and Roya Ensafi. Network measurement methods for locating and examining censorship devices. In *Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '22, page 18–34, New York, NY, USA, 2022. Association for Computing Machinery.

[54] Reethika Ramesh, Ram Sundara Raman, Matthew Bernhard, Victor Ongkowijaya, Leonid Evdokimov, Anne Edmundson, Steven Sprecher, Muhammad Ikram, and Roya Ensafi. Decentralized control: A case study of Russia. In *Network and Distributed System Security*. The Internet Society, 2020.

[55] W. Scott, T. Anderson, T. Kohno, and A. Krishnamurthy. Satellite: Joint Analysis of CDNs and Network-Level Interference. In *ATC '16*.

[56] Shawn Conaway. The Great Firewall: How China Polices Internet Traffic. Certification Magazine, 2009-09. http://certmag.com/the-great-firewall-how-china-polices-internet-traffic/.

[57] Stephen Simons. *Minimax Theorems and Their Proofs*, pages 1–23. Springer US, Boston, MA, 1995.

[58] Kushagra Singh, Gurshabad Grover, and Varun Bansal. How India censors the web. In *Web Science*. ACM, 2020.

[59] Joel Sommers, Hyungsuk Kim, and Paul Barford. Harpoon: A flow-level traffic generator for router and network tests. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '04/Performance '04, page 392, New York, NY, USA, 2004. Association for Computing Machinery.

[60] D. Turk. Configuring BGP to Block Denial-of-Service Attacks. RFC 3882, IETF, September 2004.

[61] Benjamin VanderSloot, A. McDonald, W. Scott, J. A. Halderman, and Roya Ensafi. Quack: Scalable Remote Measurement of Application-Layer Censorship. In *USENIX Security '18*.

[62] Pan Wang, Shuhang Li, Feng Ye, Zixuan Wang, and Moxuan Zhang. Packetcgan: Exploratory study of class imbalance for encrypted traffic classification using cgan, 11 2019.

[63] Zhongjie Wang, Yue Cao, Zhiyun Qian, Chengyu Song, and Srikanth V. Krishnamurthy. Your state is not mine: A closer look at evading stateful internet censorship. In *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, page 114–127, New York, NY, USA, 2017. Association for Computing Machinery.

[64] Zhongjie Wang, Shitong Zhu, Yue Cao, Zhiyun Qian, Chengyu Song, Srikanth V Krishnamurthy, Kevin S Chan, and Tracy D Braun. SymTCP: Eluding Stateful Deep Packet Inspection with Automated Discrepancy Discovery. In *NDSS*, 2020.

[65] Nicholas Weaver, Robin Sommer, and Vern Paxson. Detecting Forged TCP Reset Packets. In *Network and Distributed System Security*. Internet Society, 2009.

[66] Philipp Winter and Stefan Lindskog. How the Great Firewall of China is blocking Tor. In *Free and Open Communications on the Internet*. USENIX, 2012.

[67] Xueyang Xu, Z. Morley Mao, and J. Alex Halderman. Internet censorship in china: Where does the filtering occur? In *PAM*, 2011.

[68] Diwen Xue, Reethika Ramesh, ValdikSS, Leonid Evdokimov, Andrey Viktorov, Arham Jain, Eric Wustrow, Simone Basso, and Roya Ensafi. Throttling Twitter: an emerging censorship technique in Russia. In *Internet Measurement Conference*. ACM, 2021.

[69] Tarun Kumar Yadav, Akshat Sinha, Devashish Gosain, Piyush Kumar Sharma, and Sambuddho Chakravarty. Where the light gets in: Analyzing web censorship mechanisms in India. In *Internet Measurement Conference*. ACM, 2018.

[70] Pengxiong Zhu, Keyu Man, Zhongjie Wang, Zhiyun Qian, Roya Ensafi, J. Alex Halderman, and Hai-Xin Duan. Characterizing transnational internet performance and the great bottleneck of china. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4:1 – 23, 2020.

[71] Jonathan Zittrain and Benjamin Edelman. Internet Filtering in China. *IEEE Internet Computing '03*, 2003.

A Geneva Features Analysis

Table 4 summarizes features extracted from Geneva flows that are useful for the ML-based detection of Geneva probes. Figure 8 illustrates the difference in the distribution of payload hashes observed in Geneva and normal flows. We see that unlike the distribution for hashes from normal flows which are evenly distributed, hashes from Geneva and DeResistor tend be more skewed.

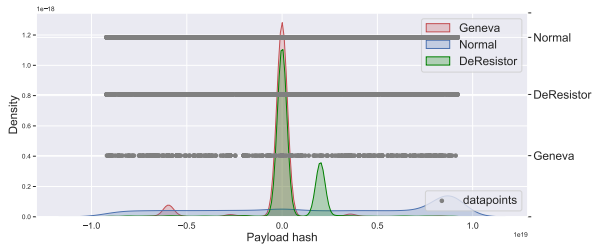

Figure 8: Hash values of packet payloads in Geneva, Normal and DeResistor flows.

B LSTM for IP-Level Detection

The choice of LSTM: LSTMs (Long Short-Term Memory) models were proposed as time-efficient RNNs (Recurrent Neural Networks). They have been widely used in sequence-based decision ML tasks such as time series for weather forecasting [38] or DNA sequence classification [38], etc.) In this paper, we train and test an LSTM model (Figure 9) to explore the usability of ML for IP-level Geneva detection based on the classification of a sequence of flow-level detection observations. As shown in Figure 9, our LSTM model has 3 layers: the input layer "LSTM1" takes as input a tensor of multiple sequences with size $L = 10$, a second layer "LSTM2" with 256 units, and a final *sigmoid* layer that returns the output label (i.e., malicious IP or not).

The choice of the sequence length L: Considering that DeResistor introduces a sequence of J jumps of normal flows, to ensure LSTM receives a sequence of DeResistor flows that include at least one Geneva flow, LSTM is trained with $L > J$ observations. Since J is usually less than 5 in the previous experiments (until experiment (5)), we select $L = 10$ to train on previous DeResistor traces.

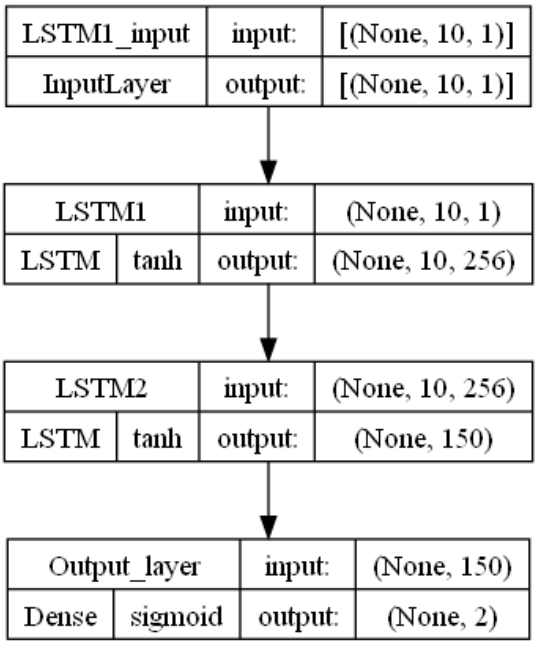

Figure 9: LSTM Model architecture for sequence-based IP-level detection.

TCP Flags	Geneva probes may tamper with TCP flags. Hence, we keep track of each packet flag's occurrence within a flow. We notice that Geneva probing can result in sending packets with unusual flags (e.g., SRPECN, FSRPUEN) that are unlikely to occur in a normal network flow.
Number of non-zero SYN packets	Geneva flows contain SYN packets with non-zero payload length, which can overlap with another packet if it has the same sequence number. In a normal flow, a SYN packet length is equal to zero. Figure 1 confirms that only Geneva flows (label=0) include SYN packets with non-zero payload, which makes it a significant feature for Geneva users detection.
Number of overlapping TCP segments	Geneva can fragment a packet into small segments and perform different actions on the resulting segments, which can lead to sending overlapping packets to hide a forbidden keyword. Figure 1 shows that compared to normal flows, Geneva flows tend to have a higher number of pairwise TCP overlaps, ranging from 0 to 60, while most normal flows do not surpass 10 overlaps.
Flow Size	Normal (benign) flows usually have higher flow sizes ($\approx$ 1000 - 30000 bytes) compared to an average of $\approx$ 150 bytes across all Geneva flows.
Maximum Packet Size	Inline with the previous observation, the maximum packet size within a flow is mostly higher in normal traffic compared to Geneva traffic.
Number of corrupt TCP data offsets	Geneva manipulations might lead to the corruption of the dataofs field of a TCP packet [13]. Figure 1 confirms that flows with corrupt dataofs are mainly observed in Geneva traces (label=0), while it is always 0 for benign flows.
Content (Payload)	Geneva might temper with packet payloads. Thus we consider it as another feature. Figure 8 shows that packet payloads of DeResistor flows resemble more normal flows compared to Geneva's.
More Features	**Corrupt TCP options** is a candidate feature. However, we choose not to consider it, since it is usually either correlated with the corrupt dataofs feature or the corrupt checksum feature. Additionally, we explore the **Destination IP** of Geneva packets as another feature since Geneva mostly attempts connections with censored URLs.

Table 4: Summary of Geneva features used by flow-level detection classifiers.