

# Using machine learning techniques for censorship-resistant communication



By  
**Zil-e-Huma**  
**206493**

Supervisor  
**Dr. Syed Taha Ali**  
**Department of Electrical Engineering**

A thesis submitted in partial fulfillment of the requirements for the degree  
of Masters of Science in Computer Science (MSCS)

In  
School of Electrical Engineering and Computer Science,  
National University of Sciences and Technology (NUST),  
Islamabad, Pakistan.

(August, 2021)

## Approval


It is certified that the contents and form of the thesis entitled "Using machine learning techniques for censorship-resistant communication" submitted by ZIL E HUMA BAJWA have been found satisfactory for the requirement of the degree

Advisor : Dr. Syed Taha Ali

Signature:  \_\_\_\_\_


Date: 26-Jul-2021

Committee Member 1:Dr. Wajahat Hussain

Signature:  \_\_\_\_\_


Date: 26-Jul-2021

Committee Member 2:Dr. Arsalan Ahmad

Signature:  \_\_\_\_\_

Date: 26-Jul-2021

Committee Member 3:Mr. Muhammad Imran  
Abeel

Signature:  \_\_\_\_\_

Date: 26-Jul-2021

# Dedication

*"For everything i am today, My mother's love showed me the way"*

## **Certificate of Originality**

I hereby declare that this submission titled "Using machine learning techniques for censorship-resistant communication" is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Student Name: ZIL E HUMA BAJWA

Student Signature: Zil e Huma

# Acknowledgment

I am grateful to Allah for blessing me with countless bounties, for giving me a life full of loving and supporting people. After Almighty, i would like to thank my mother, for her endless support and encouragement.

I would like to extend my gratitude to Dr. Syed Taha Ali, Assistant Professor, Department of Computing, SEECS NUST, for his constant guidance, devotion and motivation.

My heart-felt gratitude to my Friends who have been a constant source of love and support throughout my Masters. Without you, this would not have been possible. Thank you for always standing next to me and for being my strength. A special thanks to my husband for his constant nagging and taunts, which have made the completion of this document possible.

**Zil-e-Huma**

# Table of Contents

<b>List of Figures</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Problem Statement . . . . .	2
1.3 Objectives and Research Goals . . . . .	2
1.4 Thesis Organization . . . . .	2
<b>2 Background Information</b>	<b>4</b>
2.1 Introduction to Censors . . . . .	4
2.2 Censorship circumvention . . . . .	5
2.3 Machine Learning based Censor Models . . . . .	6
2.4 Generative Adversarial Network . . . . .	7
<b>3 Related Work</b>	<b>8</b>
3.1 Protocol Tunneling . . . . .	8
3.1.1 Through Relay Servers . . . . .	8
3.1.2 Using Multimedia Applications . . . . .	10
3.2 Generative Adversarial Network . . . . .	14
<b>4 Proposed Methodology</b>	<b>16</b>
4.1 Design of Censor . . . . .	16
4.1.1 Machine Learning Censor Models . . . . .	16
4.1.2 Training of ML models . . . . .	18
4.2 Proposed design of Multimedia Traffic Generator . . . . .	18
4.2.1 Generative Adversarial Network . . . . .	18
4.2.2 Network Traffic Generator - NTG . . . . .	18
4.2.3 pcap Generator . . . . .	19
4.2.4 Feature Extraction . . . . .	19
4.3 Implementation flow chart . . . . .	20
4.4 Final Objective . . . . .	20

<b>5</b>	<b>Implementation</b>	<b>21</b>
5.1	Machine Learning based Censors . . . . .	21
5.1.1	Choice of Machine Learning Models . . . . .	21
5.1.2	Datasets used . . . . .	21
5.1.3	Feature Extraction . . . . .	22
5.1.4	Implementation of Machine Learning Models . . . . .	22
5.2	Developing Traffic Generator . . . . .	23
5.2.1	GAN . . . . .	25
<b>6</b>	<b>Results and Discussion</b>	<b>27</b>
6.1	Background Information . . . . .	27
6.1.1	Model Accuracy . . . . .	27
6.1.2	AUC - ROC Curve . . . . .	27
6.2	Evaluation of CensorModels . . . . .	27
6.2.1	Decision Trees . . . . .	27
6.2.2	XGBoost . . . . .	28
6.2.3	AdaBoost . . . . .	29
6.2.4	Random Forests . . . . .	30
6.3	Discussion of Results . . . . .	31
6.4	Passing GAN Traffic through Censor Models . . . . .	32
<b>7</b>	<b>Conclusion and Future Work</b>	<b>33</b>
7.1	Conclusion . . . . .	33
7.2	Future Work . . . . .	33

# List of Figures

1.1	Thesis Organization . . . . .	3
4.1	High Level Layout of Proposed Methodology . . . . .	20
5.1	Censor List of Features taken from [1] . . . . .	26
6.1	Decision Tree- ROC Curve . . . . .	28
6.2	XGBoost - ROC Curve . . . . .	29
6.3	AdaBoost - ROC Curve . . . . .	30
6.4	Random Forests - ROC Curve . . . . .	31



# List of Tables

5.1	GAN's list of features . . . . .	25
6.1	Our GAN traffic through Censors . . . . .	32

# Abstract

To prevent citizens from accessing unfiltered information over the internet, repressive governments usually deploy national level censorship. To counter these censors, a number of systems have been introduced that make use of various allowed applications as covert channels and tunneling data through them. However, a drawback of these covert channels is their failure to maintain un-observability. The censors these days are capable enough to identify covert traffic amongst the stream of regular traffic. With censors getting more rigorous, some new ways of hiding blocked content in legitimate data streams have been introduced. These systems have very closely fooled the similarity based censors. However, Machine learning systems were still unbeaten. A recent study showed that some state-of-the-art Machine learning systems have been able to identify covert traffic, deeming the censorship-resistant models useless. In this thesis, we have used the strongest known Censorship-resistant system till date, to show that the system can be tweaked to defeat a machine learning based anomaly detection system. We have made use of adversarial machine learning techniques to beat some well-known classification techniques. Our results show that Adversarial examples can not only be used to hide blocked content in an application data stream but can also remain undetected

# Chapter 1

## Introduction

Internet censorship circumvention is a major problem with the ever growing restriction over data. This thesis studies the use of Deltashaper, the best known circumvention system, to establish a covert channel which is unobservable by state-of-the-art censorship systems. The idea of using various applications to establish a covert channel have been broadly studied over the last few years. However, till date, none of these systems have been able to beat the efficient Machine learning based censors. This thesis therefore presents a way to effectively bypass these censors and enable covert communication while preserving the characteristics of a regular traffic stream. Aim is to use Adversarial examples to design a data stream that can be transparent to the censor.

### 1.1 Motivation

Repressive regimes usually take control over the type of data users can access by deploying large scale surveillance systems. These systems violate people of their basic rights of data access. Data monitoring over the internet can be controlled at national level by, for example, blocking a particular site, or limiting access to a certain resource. [2]. However, luckily, even these restrictions can't completely block all external communications. For example, It is known that many countries allow skype communication but block access to several sites [3]. With these blockages, major circumvention systems have been introduced. One of the most common ways to hide behind a fake IP address is to use a trusted proxy to access blocked information. But even by using very refined systems like Tor [4], without proper obfuscation, the traffic can be easily recognized.

To hide traffic under any application, the covert traffic should mimic the reg-

ular traffic perfectly. But to mimic a protocol perfectly is almost impossible, particularly because of the complexities involved. With un-observability almost impossible, censors are hard to beat. The aim of this thesis is to preserve un-observability of a circumvention system, DeltaShaper [5], by generating adversarial examples similar to the original traffic and using them to reshape the obfuscated stream. This unobservability claim previously made by the author have been proven false by the same author in [1]

## 1.2 Problem Statement

Our goal is to develop a covert channel that can go undetected by various state-of-the-art Machine learning based censor models. To demonstrate this, several well known machine learning models will be trained on both regular and non-regular traffic. Our generated traffic stream will be able to beat all these systems by evading detection.

## 1.3 Objectives and Research Goals

The goal of this research is to propose and implement a traffic generator that makes use of Machine learning to masquerade its traffic as benign and goes undetected by modern machine learning censors . The highlights of the research work are as follows:

- Implementation of state-of-the-art machine learning censors.
- Training and testing the efficiency of these censors.
- Using GAN to generate features for Regular traffic dataset
- Implementation of a traffic generator that generates two-way traffic
- Bypassing detection of Deltashaper traffic

## 1.4 Thesis Organization

The presented thesis has been organized into different chapters in which each chapter gives certain aspects of our research, depicted in figure 1.1. Following is the brief description of all the thesis chapters.

- **Introduction:** A description of context, problem statement and scope.

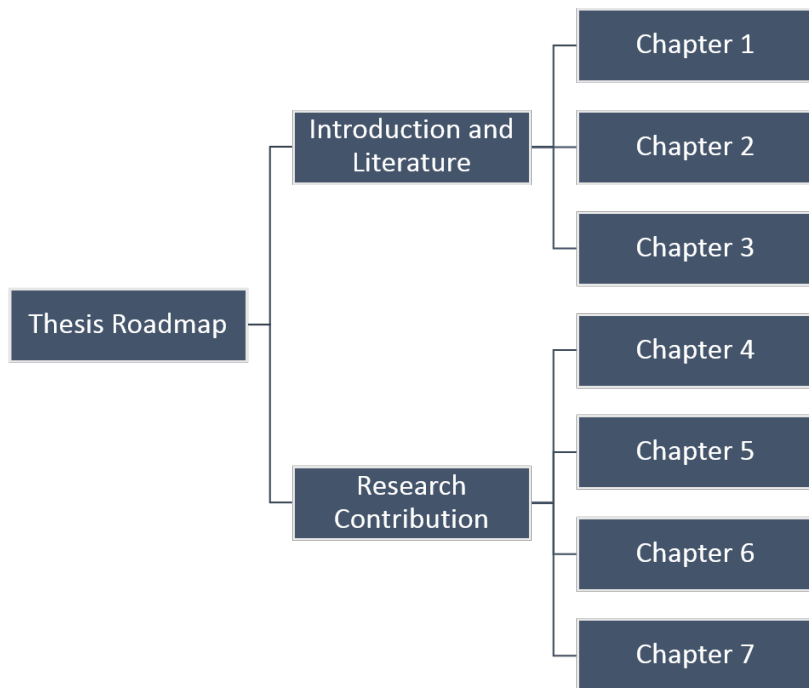


Figure 1.1: Thesis Organization

- **Background Information:** A discussion on the theoretical and technological background that this thesis is based upon.
- **Related Work:** A summary of related research that demonstrates the motivation to solve the problem statements.
- **Proposed Methodology:** A structured overview of proposed strategies that contribute to solving the problem statement.
- **Implementation:** A review of the implementation and code to achieve the objectives and goals.
- **Results and Discussion:** A discussion about results of the work and an assessment of the achievements.
- **Conclusion and Future Work:** A summary of the work that was done and contributions made to this project.

# Chapter 2

## Background Information

An introduction of some important concepts prerequisite to this research have been elaborated in this chapter. These concepts will help establish a background understanding of the underlying technology discussed in subsequent chapters

### 2.1 Introduction to Censors

Censorship as a broad term is known as the suppression of information based on some reasoning. State-level censorship is technically designed to ensure that only the white-listed information or resource is allowed to pass through, everything objectionable is suppressed. What's deemed objectionable is usually relative to either the government, nation or situation. 'The Great Firewall Of China' [6] is the perfect example of a state-level censorship which has created more than sixty restrictions till 2019. These restrictions have at times been imposed due to certain political changes in the country, like elections.

However, due to economic and social pressure, even censors can't just block anything and everything. This leaves room for a lot of circumvention techniques. A good censor has to identify banned traffic amongst the allowed traffic very efficiently. There are several ways censorship can be imposed:

- Traffic analysis
- Software level censorship
- Service Hindrance
- Promote domestic Servers

In the next chapter, various censorship models and their ways of circumvention will be discussed in detail

## 2.2 Censorship circumvention

Censorship circumvention follows two very important key concepts:

1. If a censorship circumvention technique can be identified by the censor correctly, it lacks the key criteria of the perfect censorship circumvention system, un-observability
2. A system is deemed unblockable if the censor cannot completely or partially block it without affecting actual traffic.

Following are some of the commonly known censorship techniques and their counter circumvention models:

- **Content With-holding**

Various internet services openly entertains government's request to either block the service or limit it in a particular area. This is especially done during times of Political instability. For example, twitter has been on/off withholding contents for certain countries during election campaigns. Many a times, certain accounts are suspended or blocked upon federal request, for avoid any civil disturbance situation.

To counter this censorship, various steganographic [7] techniques have been used, where a hidden message/blocked content is concealed under normal traffic. Steganography is a technique of hiding a message,file,image under another. The technique has been used widely in image processing as well

- **End-point Blocking**

The simplest strategy a censor uses to block access to a particular resource is by blocking that end point altogether. For example, if the source of the content to be blocked is known, e.g its IP address or host-name, a simple block-list or white-list can be established to get the desired result by the censor.

However, as simple as the technique it, its circumvention is even simpler. Using various proxy rerouting software, one can reroute the traffic to a middle proxy server which can then further directs the traffic to the destination. This proxy server does not reside in the censorship area. Some very famous example of such proxies are wingate, tinyproxy, freeproxy and ultrasurf.

- **Blocking publicly distributed proxies**

A censor can very easily block traffic every through the proxies, by blocking famous, publicly available proxies. For example, any connection made towards the ultrasurf proxy can be easily detected and blocked by the censor

As a counteract, traffic morphing technique has been established. It obfuscate the intended traffic and transforms and encrypts it in such a way that it cannot be identified as any particular stream. Instead, it comes as a random traffic and hence cannot be identified by statistical approaches. Tor browser is a famous example of this technique.

- **Protocol Randomization**

Protocol randomization is an example of traffic morphing where the generated traffic is changed so it doesn't belong to any known protocol and seems random. Censors having protocol level checks fail to identify such traffics as anomalies. However, with every growing detection efficiency of censors, this technique can be identified easily. A counter technique that has been deployed by many is Traffic mimicking, where instead of randomizing the traffic, traffic parameters are changed to mimic any unblocked protocol. Examples of such systems are SkypeMorph and CensorSpoofer.

- **Protocol tunneling**

A technique where data is embedded in a cover protocol so that perfect mimicking of any particular protocol is not required. This is an excellent technique for hiding data under as a tunnelling channel because mimicking any particular protocol exactly is almost impossible. However, embedding covert data in a legit traffic may hinder the network parameter like packet length or size. Such changes can be identified by the censors

## 2.3 Machine Learning based Censor Models

Machine learning is the gist of Artificial Intelligence, where, as the name implies, machines are made to 'learn' through various statistical models. Once the learning or as widely called, the training part has been completed, such machines are able to perform their defined task with an accuracy dependant on the machine's learning ability.

Machine learning can be Supervised, unsupervised or sometimes semi-supervised



- Supervised Machine Learning Supervised Machine learning involved training algorithms on labelled data sets. Labelled datasets not only help the algorithm to learn the input data more accurately, but also helps in better problem solving. Supervised machine learning problems are further grouped into two types: Regression: Where the output is a real value Classification: Where the out is a category
- Unsupervised Machine Learning Unsupervised Machine Learning, training of algorithm is done through unlabelled data. This means that the algorithm itself has to make sense out of the data. unsupervised ML problems are further grouped into Clustering: Finding groups in a data Association: defining rules to cover major chunk of data
- Semi-supervised Machine Learning A branch of ML where, training data contains both labelled and unlabelled data.

## 2.4 Generative Adversarial Network

Generative Adversarial Network or GAN [8] is a branch of machine learning where two neural networks work in competition with each other to win. The main idea behind this approach is to train the GAN through the discriminator model. This unsupervised learning of the generator is done to fool the discriminator rather than to get trained itself.

- Generative Model The Generative Model is supposed to generate new data examples after learning the pattern of the input data.
- discriminator model The discriminator model is basically a check to see if the example generated by the generative model is close to the input data or not. The difference in the generated data and the real data is fed back to the generative model where it relearns from the feedback. In this way, generative model is trained to produce dataset similar to the input data. The model may be said to reach an equilibrium when the discriminator is finally beaten.

# Chapter 3

## Related Work

This chapter reviews the relevant state of the related researches and their comparison with the proposed scheme.

### 3.1 Protocol Tunneling

Protocol tunneling has been used widely in the past to establish various covert systems. Some renowned ones are discussed below:

#### 3.1.1 Through Relay Servers

To hide covert data, many systems introduced in the past have established a relay server which entertains client requests, converts the data under a protocol and then forwards it to the destination

- **SWEET** [9]

'Serving the Web by Exploiting Email Tunnels' or SWEET is a protocol tunnelling system which makes use of email communication to hide data. Methodology: Sweet methodology is pretty simple. A Client sends out an email message which is directed towards the SWEET server. The server has the ability to open the email message, extract the hidden data or requested resource, fetch the request and reply back to the email with the requested content. The server is specifically designed to entertain requested resources through an indirect path.

The advantage of this approach is that since SWEET makes use of email communication, which is an important mode of communication in today's world, it is highly unlikely to be fully blocked by the censors.

Especially when famous email clients or services are used, blocking becomes extremely tricky.

The disadvantage and the weakness of this approach lies in the fact that any email message crafted with hidden data can be discriminated from a normal email message through network traffic analysis. Network parameters of such a traffic may differ far enough to be recognized as anomalous. Moreover, normal email communication takes place over a time span and may take hours or days to get a reply. SWEET server's instant and continuous emails can also look fishy to an intelligent sensor model

- **CloudTransport** [10]

CloudTransport makes use of cloud services to transfer covert data. The idea is to make use of a famous cloud service, setup an account and exchanges the credentials between the client and cloudtransport bridges.

Methodology involves generating a request through a CloudTransport client onto the cloud. The CloudTransport bridge periodically checks the cloud to see if any new request has been initiated. If so, it acts as a server and fetched the requested content and writes it back to the cloud from where the client can read it back.

The advantage of this approach is also similar to SWEET. Being a commonly used service, censor cannot possibly block client access to the cloud. The limitation of this approach lies again in traffic analysis. Moreover, a suspicious censor may redirect the traffic being generated by the cloudTransport client towards its own bridge and block the traffic altogether.

- **CASTLE** [11]

Castle makes use of RTS games to transfer covert data through specific commands. The system uses automation to give specific game commands in order to request/hide data. These commands have to be

decided after mutual understanding of the client and destination since it is important for the receiving end to understand and be able to read the covert data sent through system commands.

The advantage of this system is that censors are usually not monitoring game traffics. Moreover, since game traffic is already diverse, detection through traffic analysis might be harder than previously discussed systems.

The limitation of this system is overall throughput that can cause severe performance issues.

- **meeK** [12]

meeK is an intelligent tunneling system which makes use of domain fronting. It acts by having different domain names in different network layers and hiding the actual destination address in HTTPs traffic request. For example, the blocked domain is written in HTTPs header whereas the DNS or other packets correctly contain an allowed domain. When such a request reaches a frontend server, it decrypts the packets and redirects it to the intended domain.

This is an excellent idea and cannot be detected by normal sensors easily.

However, the limitation is again in traffic analysis where the normal TLS traffic may differ from meeK traffic. Moreover, the latency and lag observed in packet transmission for meeK is more than usual and may be suspicious.

### 3.1.2 Using Multimedia Applications

Apart from separate relay servers, some systems make use of common oblivious servers of various multimedia applications, to transfer covert data. Such systems have been discussed below. Our proposed idea is also a part of such system

- FreeWave [13]

Freewave works by modulating data generated from client into acoustic

signals and send it over a Voice over IP system. This data is sent to a freewave server which is able to extract and demodulate the covert data and give further access to the client. The system is specially designed to ensure unobservability through modulation. The idea has been proven using skype audio call. Data is modulated into skype audio call and is directly sent to freewave server. The server demodulates the data and entertains further request.

The advantage of this approach is that freewave cannot be fully blocked by censors as it makes use of some renowned VoIP systems. The disadvantage is the issue of packet length that may vary considerably from the normal skype voice call.

- CovertCast [14]

Covertcast makes use of images to send blocked data across. Any requested site or resource is scraped into images and modulated into the live video streaming channel. In this way the blocked content is transmitted further towards the receiver end. The receiver needs to know the url of the live video streaming channel where the covert data is being transmitted via images. the transfer rate is such that the data appears to be a video to the receiver.

The advantage of such a scheme is somewhat unobservability. However, modern day censors are able to detect covercast traffic easily. The main disadvantage and a known limitaion of covertcast is that its unidirectional.

- Facet [15]

Facet is somewhat similar to covercast but instead of sending images, facet works by sending video data. A facet client sends its request to Facet server through an instant message. The server gets the url from the message, downloads the requested content and initiates a video call to the source client. The requested content is shared through this video call. Although a good approach but is limited to video transmission only.

- DeltaShaper [5]

DeltaShaper is a system designed to enable resistance against the censorship over the Skype streams. This is done by enabling the TCP/IP tunnelling over the video calls in the Skype communication. The TCP/IP packets containing the covert information are encoded and

transmitted over the video streams of original Skype streams. Thus, DeltaShaper ensures that the encoding of TCP/IP packets is performed such that they remain unobservable and unrecognizable in the Skype stream.

DeltaShaper is useful for the client who connects to a censored ISP and want to communicate covert data to another user beyond this censored network using a covert TCP/IP connection.

At the sending side, the transmitter receives the data as payload. This data is encoded in a video stream which is sent to Skype using a camera interface. This video is transmitted to the remote Skype instance.

At the receiving end, the Skype stream received is captured using the Skype video buffer. The receiver can decode the video stream and extract the payload. This payload is delivered then to the receiving application.

Both these setups are implemented at both ends of the communication supporting a bi-directional channel. Additionally, to support generalization, upper layers have been given access to data-link layer protocol so that IP-packet can be manipulated remotely. This enables the system to support TCP/IP application with high latency and low through puts links.

### **Preserving unobservability**

The estimation of the features of the traffic of an encoded video stream becomes a difficult task to accomplish due to the complexity of the optimization achieved by the algorithms as the video is encoded/decoded. This adds to the difficulty in generating a Skype stream containing modulated covert data which is similar to the “normal” Skype stream. Thus, DeltaShaper is used to help generation of the covert videos which shows same features as “normal” Skype stream as it encodes a large sized payload data. Thus, it becomes important to first study how a “normal” Skype stream looks like and then use this information for development of a technique which can modulate the covert streams which shows similar patterns.

A regular Skype stream was categorized as the one which forms form a legitimate video call between the Skype users and it has no covert information encoded in it. For a normal Skype call, where user is sitting in front of the camera moving rarely, the traffic follows a specific pattern. While, the traffic patterns of any action movie transmitted using

Skype are very different resulting in large number of lengthy frames. A stream is considered to be irregular if it is different to a regular Skype stream more than a pre-set threshold,

Feature function detects some quantitative attributes from the packet traces. The feature distribution of packet length () as a valuable attribute to identify a pattern in Skype stream. depends on the input video and the compression applied by Skype. Thus, combining the payload frames with the carrier frames directly effects the packet length distributions. Such a differentiation scheme has been successful in distinguishing between Skype and Tor [25] streams. In [22], to differentiate between Skype and YouTube streams, a function based on 2-gram distribution of packet lengths has been presented. When studied with DeltaShaper, the results are similar to those obtained for .

DeltaShaper must be fed with experimentally chosen encoding selectors. DeltaShaper can select one of these encoding selectors which delivers the greatest throughput. Providing diverse options to the DeltaShaper as encoding selector can help generalize their selection mechanisms, independent of any specific video applications.

The advantage of deltashaper is that its highly unobservable for even many complex systems. However, this system has also been beaten by state of the art machine Learning models that can distinguish between Skype traffic and deltashaper traffic with great precision. This has been discussed in [1]

All the above multimedia systems shared above have been beaten by state of the art Machine learning systems as stated in [1]. In [1], Accuracy of CoverCast, Facet and Deltashaper has been compared and their unobservability have been questioned.

The main findings of the publications are as follow: Unobservability claims of all three systems are wrong. Similarity based classifiers are able to detect Facet traffic with very high accuracy as discussed below; Performance of X2 has been rated good with 74.3 accuracy, when Facet s=50 traffic is detected. KL and EMD supports optimistic support to the unobservability of Facet s=50 by showing results to accuracy of random guessing. In classifying the DeltaShaper and Facet traffic, X2 results in large false positive rates. While detecting the Facet and DeltaShaper traffic, Performance of X2 has been considered as fair in distinguishing the covert channels (e.g., AUC=0.83 for Facet s=50, AUC=0.74 for DeltaShaper). CovertCast results in failure to present unobservability. Classifier is able to discriminate CovertCast streams with only 2 FPR,

For decision Tree based models:

A combination of Random Forest/ XGBoost with the summary statistics negates the unobservability claims of the multimedia protocol tunneling systems. The detection of the Facet and DeltaShaper traffic using the summary statistic features A set of limited number of false positives can be used to flag most of the covert channels.

A recent study on censorship circumvention done by Diogo et al in [16] introduces a tool names Protozoa. It is a tunnelling tool that introduces a covert channel which has a good performance and is undetectable by state level censors. The idea is to make a video call to an uncensored party through a third party tool to establish a channel using a WebRTC streaming service. This is done by replacing the encoded video data with IP packet, which ensuring that the statistical properties of the stream remains genuine.

## 3.2 Generative Adversarial Network

Generative Adversarial Networks by Goodfellow et al was introduced in 2014. Since then, the machine learning field has seen drastic advancement and it has been termed as the most exciting field of recent times owing to its applications in almost all fields of life.

GAN is basically a contest or a competition between two neural networks, where both try to beat the other by being more accurate in their own predictions. The two neural networks that make up GAN are called Generator and discriminator. The generator tries to generate fake output data that looks similar to real data, whereas the discriminator distinguishes between real and fake data example. The most promising use of GANs have been seen in data imaging where the generator model is usually a convolutional network and the discriminator is a deconvolutional network. A well known image-to-image translation done by GAN is where sketches and real photos can be generated through one another. Similarly, totally new set of features and faces have also been derived through GAN [17] Generative adversarial

In [18], stacked GAN was introduced to generate realistic images based on text description. GAN have various applications in computer vision as well. However, its applications are not restricted to just imaging. GAN has been used widely for Captcha solver in [19] since its accuracy and efficiency is unbeatable.

In [20], medical applications of GAN have been discussed. GAN can be used to generate multicontrast MRI images for better picture quality and hence better diagnosis. Various other applications of GAN and its limitations have been discussed in [21]. Where GAN has numerous application, it has its own shortcomings as well. Real Images generated by GAN can be misleading



and can result in social discomfort at numerous levels. However, like every new tool, the mode of usage of GAN determines whether its a blessing or a curse.

In security domain, GAN has great contributions in steganography as was proposed in [22]. The architecture of this GAN had one generative and two discriminative models to hide information effectively. The work most relevant to our implementation was done by Maria et al in [23] where they used a GAN to beat network traffic classification. They used a total of 6 network traffic features to mimic Facebook chat network traffic by learning on it.

# Chapter 4

## Proposed Methodology

This chapter provides a high-level overview of the proposed methodology that will address the problem statement. As discussed in previous chapter, Deltashaper, the best known covert system has been beaten by the state-of-the-art Machine learning models in [1]. In this chapter, we propose a methodology whereby our covert deltashaper traffic will go undetected by the same censor models.

### 4.1 Design of Censor

#### 4.1.1 Machine Learning Censor Models

We propose using well-known state-of-the-art ML models to act as our censors. We will work with multiple censor models to ensure our system is robust and the claim of unobservability in our system is properly presented. We propose usage of all the Decision-tree based ML models that have been discussed in [1]. These models are best known for their accuracy and hence fooling them, will pretty much beat all known machine learning censors. Since these models were able to correctly beat deltashaper traffic, we will use the same models and beat those models through our covert tunnelling. Following Machine learning models have been used:

##### 4.1.1.1 Decision trees [24]

The network traffic generator aims takes the featureset produced y Gan as input, and generates a packet with these features and transmit it over to a peer receiver. This is a two way communication system, where, the sender not only sends the covert data encapsulated in a packet with generated network features, but also shares the destination point features with the destination

system. The next packet generated by the destination system holds the parameters it received in the sender packet. In this way, a communication is established to ensure both clients use the parameters generated by GAN to ensure two way traffic. A branch is split by the decision node using an attribute, which is chosen according to the information gain (i.e. a probable reduced entropy due to choice of the splitting attribute). Analysis of the tree structure can help in identifying the significance of each attribute, which reveals that nodes near the root are more importance as compared to the ones down the tree. Decision trees can still result in cumbersome ungeneralized models or unstable models due to greater number of correlated features. Decision tree ensemble scan be used to address this problem.

#### **4.1.1.2 Random Forests**

Random Forests [25] considers the outcome from multiple decision trees to predict a label as its an ensemble learning technique. Bootstrap aggregation is used by Random Forest to incorporate variance in the model to avoid overfitting. This is done to ensure that each tree is trained using random sample of data. Also, feature bagging is used by Random Forest to ensure that feature selection is random to build each tree. An average of the information gain among all the trees in the ensemble can be caluclated to appreciate the significance of an attribute.

#### **4.1.1.3 XGBoost**

eXtreme Gradient Boosting (XGBoost) [26] builds a model considering the ensemble of a decision trees through the use of gradient tree boosting. In XGBoost, a shallow decision tree is build. KGBost builds a new tree in the earlier steps reflecting towards positive predictions. XGBoost controls overfitting through recolutionarized model formalization. Similar to the Random Forest, significance of the individual attributes can be evaluated. Considering large number of classification algorithms, XGBoost shows promising outcomes. Recently, XGBoost has been in solutions which won different data mining competitions covering diverse fields including KDD cup 2016 [12, 44]. Logistic Regression [27] model (or logit model) is a widely used machine learning model. It is one of the basic models and is the first one to be taught. It uses cost functions and a logistic function to model a binary dependent variable.

It is basically a statistical model, but by applying a cutoff value, one can classify the inputs with probability greater than the cutoff as one class, and

below the cutoff as the other.

### **4.1.2 Training of ML models**

We will train our machine learning models on the dataset that was used by authors in [1]. The dataset includes a set of two way communication of pcaps for skype traffic and for deltashaper traffic. The goal is to train each ML model exactly how it was trained by the authors and beat these models.

## **4.2 Proposed design of Multimedia Traffic Generator**

### **4.2.1 Generative Adversarial Network**

For Generative model, or for generating adversarial examples, we propose using LSTM [28] Neural network for multimedia examples. LSTM is a type of RNN [29] with memory. It has been observed, that with LSTM used as the model function, the parameters generated by GAN are way more accurate and close to the original traffic. The activation layer or a neural network decides how to transform the weighted sum of input into the output. commonly known activation layer functions include tanh and sigmoid. For generative model, we have used Tanh function due to its ability to produce better discrete results. In our methodology, Skype normal traffic is given as the input to GAN. GAN uses features from this traffic to generate similar dataset. GAN outputs a set of network parameters, that when adopted, will generate a traffic similar to skype normal traffic. Idea is to use these generated parameters to shape our own network traffic with covert data.

### **4.2.2 Network Traffic Generator - NTG**

The network traffic generator aims to take the feature set produced by Gan as input, and generates a packet with these features and transmit it over to a peer receiver. This is a two way communication system, where, the sender not only sends the covert data encapsulated in a packet with generated network features, but also shares the destination point features with the destination system. The next packet generated by the destination system holds the parameters it received in the sender packet.

In this way, a communication is established to ensure both clients use the parameters generated by GAN to ensure two way traffic.

For traffic generation through this generator, One host and one guest machine

was setup. Both were connected via bridged networking. One DeltaClient was run on the host machine and the other on the guest machine. To ensure that both machines communicate perfectly, both were assigned an Ip in the same network and a ping request was sent from both to the other. Upon successful connection, the traffic generation was started. A UDP packet containing all info extracted from Gan was sent from the client to the guest. To ensure smooth two way traffic, immediately after reception a reply packet is sent, although no delay has been added. To ensure that the traffic was going smoothly, a text message was displayed by each client upon message reception. The communication continued, until total packets had been reached.

### **4.2.3 pcap Generator**

A small model in NTG has been integrated for pcap generation. So the traffic generator not only establishes udp connection between clients, but also captures the traffic being generated. It outputs each session in a pcap file.

### **4.2.4 Feature Extraction**

An extremely important part of our research was to extract 160+ features from a network pcap and use them as an input to several ML model. Feature extractor has been written in python where dpkt library is used to read a pcap file and extract several network parameters from it. Further features are derived through these parameters.

### 4.3 Implementation flow chart

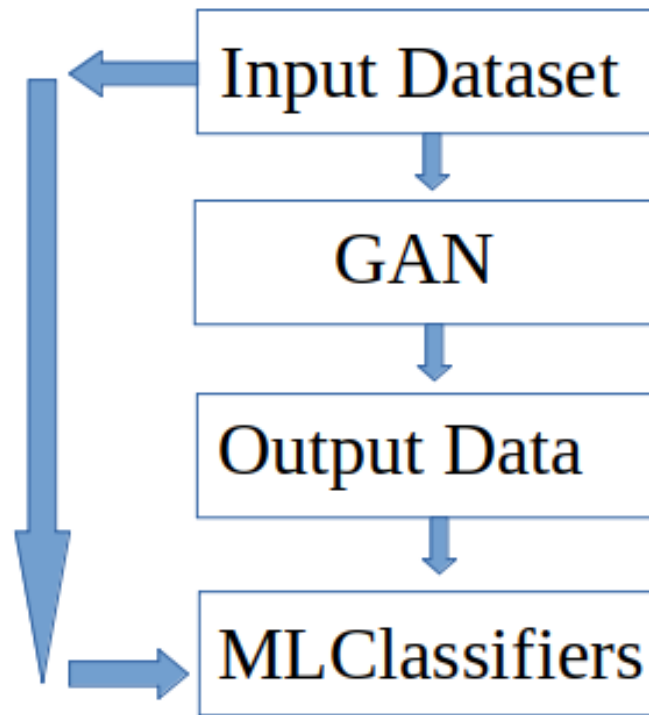


Figure 4.1: High Level Layout of Proposed Methodology

### 4.4 Final Objective

The final goal is to use NTG to generate a traffic based on features generated by GAN. Ideally, this traffic should be marked as a normal skype traffic by ML based sensors. And can be used to embed DeltaShaper payload in it.

# Chapter 5

## Implementation

This chapter explains the implementation of the proposed solution for the problem statement. The implementation has been divided into two main parts, the Machine Learning based censor models and traffic generator using GAN input. The two sections will be discussed in detail below.

### 5.1 Machine Learning based Censors

#### 5.1.1 Choice of Machine Learning Models

We have chosen all the well known ML models in order to cover wide range of censor implementations. The models also include the one's mentioned in [1] The list of them are given below:

- Decision Tree
- XGBoost
- AdaBoost
- Random Forests

#### 5.1.2 Datasets used

300 pcaps of normal and deltashaper traffic have been used to train our classifiers and GAN.

### 5.1.3 Feature Extraction

For feature extraction we used the dpkt library of python. The library was used to read each pcap packet wise. Packet size, times, length, burst and several other features are extracted. From them, Statistical indicators of these features are calculated as was done by [1]

All extracted and derived features have been used to train the censor models. However, feature reduction have been done in order to generate traffic from GAN as we've seen that not all features are needed for making a complete picture of network traffic.

A set of 160 features are chosen which are given taken from [1]. These features ensure the highest accuracy for the classifiers.

### 5.1.4 Implementation of Machine Learning Models

All ML models are implemeted in Python 2.7 virtual environment using the scikit library. This library contains readymade ML model functions that can be directly called over a preprocessed dataset

A single classifier function has been written which is called for each classifier model from main

```
def __name__ == "__main__":
    cfgs = [
        ["RegularTraffic",
         "DeltaShaperTraffic_320"],
        ["RegularTraffic",
         "DeltaShaperTraffic_160"]]

    if not os.path.exists('xgBoost'):
        os.makedirs('xgBoost')

    classifiers = [
        [DecisionTreeClassifier(), "DecisionTree"],
        [RandomForestClassifier(n_estimators=100,
                                max_features='auto', n_jobs=1), "RandomForest"],
        [XGBClassifier(), "XGBoost"],
        [MLPClassifier(), "NN"],
        [KNeighborsClassifier(), "KNN"]
    ]

    feature_set = 'Stats_60'
```



```

data_folder = 'FeatureSets/' + feature_set + '/'
if not os.path.exists('xgBoost/' + feature_set):
    os.makedirs('xgBoost/' + feature_set)

for cfg in cfgs:
    for classifier in classifiers:
        print "Running classifiers for " + cfg[0] + " and " +
cfg[1]
        runClassification_CV(data_folder, feature_set, cfg,
classifier)

```

As can be seen in the above python code, each ML classifier model is fed to the classification function, in which training and testing of the classifier takes place. Moreover, all performance metrics are also calculated in that one function. ROC curves [30] are also generated and saved in a separate file.

## 5.2 Developing Traffic Generator

Traffic Generator has also been implemented in python 2.7. A UDP packet with all essential information, including payload and network parameters are sent by the DeltaClient1 to client2. Packet has been designed via sock. The traffic generator has been coded so that it can generate both tcp and udp packets, although we only generate udp packets through it. As can be seen in the below code snippet, the input data is read from the file generated by GAN and it is fed to the traffic generator. The Proto in out case has been fixed as udp although, it can be changed to tcp for any other kind of application usage. Other features generated by GAN like total packets or total bytes have been read from the file and given as input.

It is important to note that not all features are generated through GAN. Since GAN works has generating dataset that following a particular statistical distribution, it is hard to generate dependant data through GAN. For example, for our testing , we require Total packets, ingress packets and outgress packets. However when all three features are generated through GAN, they don't add up as they should. Total packets should be a sum of incoming and outgoing packets, which is not.

Therefore, in our scenario, we have totally avoided derived features generation through GAN. Instead, we've generated total packets and total source packets, and then extracted destination packets ourself by subtracting both. Similar practice has been done for Total bytes as well.

```

for line in data:
    features = line.split(',')
    Proto = "udp"
    Sport = 53771
    Dport = 43969
    TotPkts = abs(int(float(features[0])))
    TotBytes = abs(int(float(features[2])))
    SrcPkts = abs(int(float(features[1])))
    SrcBytes = abs(int(float(features[3])))
    sTtl = 64
    dTtl = 64
    SIntPkt = abs(int(float(features[5])))
    DIntPkt = abs(int(float(features[4])))
    DstPkts = TotPkts - SrcPkts
    DstBytes = TotBytes - SrcBytes
    delim = ','
    message = str(DstPkts) + delim + str(DstBytes) + delim +
str(dTtl) + delim + str(DIntPkt)
    print message
    protocol_str = 'GET / HTTP/1.0'
    byteperpkt = SrcBytes/SrcPkts
    byteperpkt -= 56
    byteperpkt -= len(message)
    if byteperpkt > 1250:
        byteperpkt = 1250
        SrcPkts = SrcBytes/1250
    message = message + '-' + '*'*byteperpkt
    tcp_pkt = False
    if Proto == 'tcp': tcp_pkt = True
    if tcp_pkt: sock = tcp(DstAddr, Dport, SrcAddr, Sport, 1,
1, SrcWin, sTtl, protocol_str+delim+message)
    else: sock = udp(DstAddr, Dport, SrcAddr, Sport, sTtl,
protocol_str+delim+message)
    for i in range(1,SrcPkts):
        if tcp_pkt: sock = tcp(DstAddr, Dport, SrcAddr, Sport,
0, i+1, SrcWin, sTtl, message)
        else: sock = udp(DstAddr, Dport, SrcAddr, Sport, sTtl,
message)
        time.sleep(SIntPkt)
    time.sleep(2)

```

### 5.2.1 GAN

GAN ensures that our deltashaper traffic is marked as normal traffic by all Machine learning algorithm models. Therefore GAN is the most crucial part of our implementation. The generative model is LSTM based with tanh as its activation function. The implementation has been done in python 2.7 using tensorflow.

The discriminative model has sigmoid as its activation functions. GAN is trained on Normal Skype Traffic dataset. After this traffic, GAN produces a set of features that mimic skype traffic.

The batch size for training and number of epochs have been fixed after hit and trial. For any dataset, finding the best value of epoch is important to avoid overfitting and underfitting both.

<u>Feature</u>	<u>Description</u>
Sport	source port
TotPkts	total packets exchanged
TotBytes	total bytes exchanged
SrcPkts	source packets sent
SrcBytes	source bytes sent
sTtl	source time to live
dTtl	destination time to live
SIntPkt	source inter-packet arrival time
DIntPkt	destination inter-packet arrival time

Table 5.1: GAN's list of features

The list of features that are used to train GAN are different from the one's used to train classifiers.

For classifier models, all 160+ features have been used for training. However, for generative purposes, only a small subset of network features have been used. These features were selected based on the feature importance study done in [1]. The features on which classifiers were trained are mentioned below. As can be seen in below figure, the main features are around seven to eight. However, their mean statistics have been derived to get 6-7 sub-features from a single feature. Feature extraction has been done through a specialized script.

```

summary statistics:
1 - Total number of packets.
2 - Total number of packets - ingress.
3 - Total number of packets - egress.
4 - Total bytes transmitted.
5 - Total bytes transmitted - ingress.
6 - Total bytes transmitted - egress.

Global statistics:
7 - Mean of packet sizes.
8 - Std. deviation of packet sizes.
9 - Variance of packet sizes.
10 - Kurtosis of packet sizes.
11 - Skew of packet sizes.
12 - Maximum packet size.
13 - Minimum packet size.
14:22 - (10-90) percentile of packet sizes.

23 - Mean of packet times.
24 - Std. deviation of packet times.
25 - Variance of packet times.
26 - Kurtosis of packet times.
27 - Skew of packet times.
28 - Maximum packet times.
29 - Minimum packet times.
30:38 - (10-90) percentile of packet times.

Statistics for ingress/egress traffic:
39:70 - 7:38 computed over ingress traffic only.
71:102- 7:38 computed over egress traffic only.

Ingress Packet bursts statistics:
103 - Total number of bursts.
104 - Mean burst size.
105 - Std. deviation of burst sizes.
106 - Variance of burst sizes.
107 - Maximum burst size.
108 - Kurtosis of burst sizes.
109 - Skew of burst sizes.
110:118 - (10-90) percentile of burst sizes.

Ingress Bytes bursts statistics:
119 - Mean bytes transmitted across bursts.
120 - Std. deviation of bytes transmitted across bursts.
121 - Variation of bytes transmitted across bursts.
122 - Kurtosis of bytes transmitted across bursts.
123 - Skew of bytes transmitted across bursts.
124 - Maximum number of bytes in a burst.
125 - Minimum number of bytes in a burst.
126:134 - (10-90) percentile of bytes transmitted.

Egress Packet bursts statistics:
135 - Total number of bursts.
136 - Mean burst size.
137 - Std. deviation of burst sizes.
138 - Variance of burst sizes.
139 - Maximum burst size.
140 - Kurtosis of burst sizes.
141 - Skew of burst sizes.
142:150 - (10-90) percentile of burst sizes.

Egress Bytes bursts statistics:
151 - Mean bytes transmitted across bursts.
152 - Std. deviation of bytes transmitted across bursts.
153 - Variation of bytes transmitted across bursts.
154 - Kurtosis of bytes transmitted across bursts.
155 - Skew of bytes transmitted across bursts.
156 - Maximum number of bytes in a burst.
157 - Minimum number of bytes in a burst.
158:166 - (10-90) percentile of bytes transmitted.

```

Figure 5.1: Censor List of Features taken from [1]

# Chapter 6

## Results and Discussion

This chapter shows the results of the implementation and its evaluation

### 6.1 Background Information

Following are some of the terms that need explanation in order to comprehend the results properly

#### 6.1.1 Model Accuracy

Ratio of correctly predicted to the total.  $\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$

#### 6.1.2 AUC - ROC Curve

AUC-ROC curve is the representation of how accurate the model is at predicting the classes of any set of input data correctly. The higher the AUC, the better the model

### 6.2 Evaluation of CensorModels

#### 6.2.1 Decision Trees

Training Time: 0.06677s

Testing Time: 0.00811s

Accuracy: 97.743%

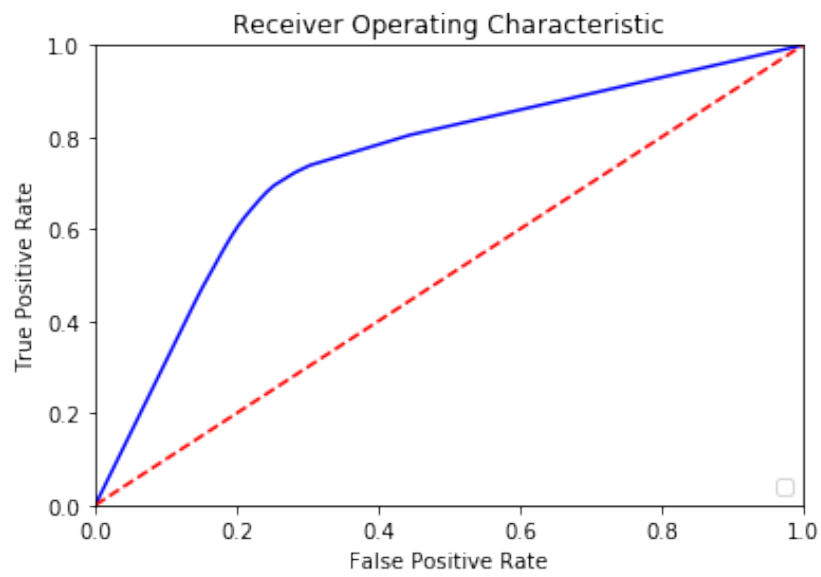


Figure 6.1: Decision Tree- ROC Curve

### 6.2.2 XGBoost

Training Time: 0.6385s  
Testing Time: 0.09771s  
Accuracy: 94.99%

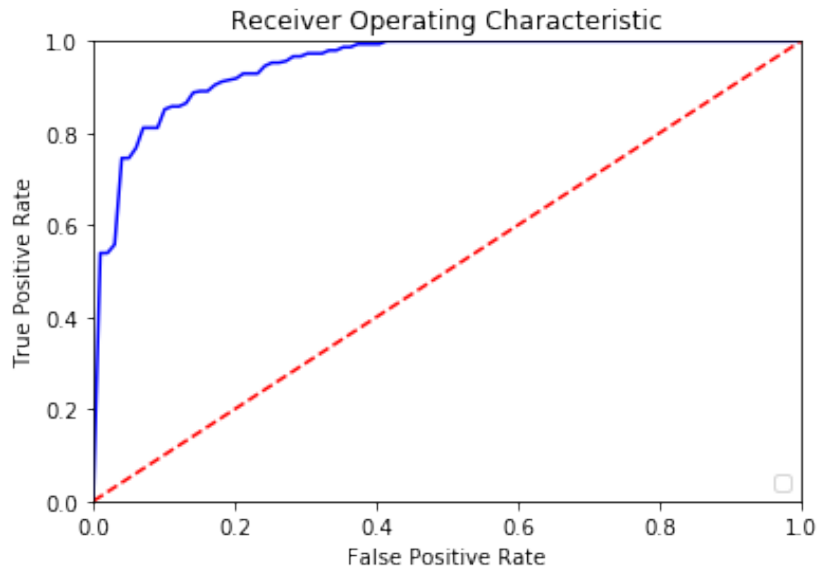


Figure 6.2: XGBoost - ROC Curve

### 6.2.3 AdaBoost

Training Time: 0.5744s  
Testing Time: 0.013995s  
Accuracy: 97.9644%

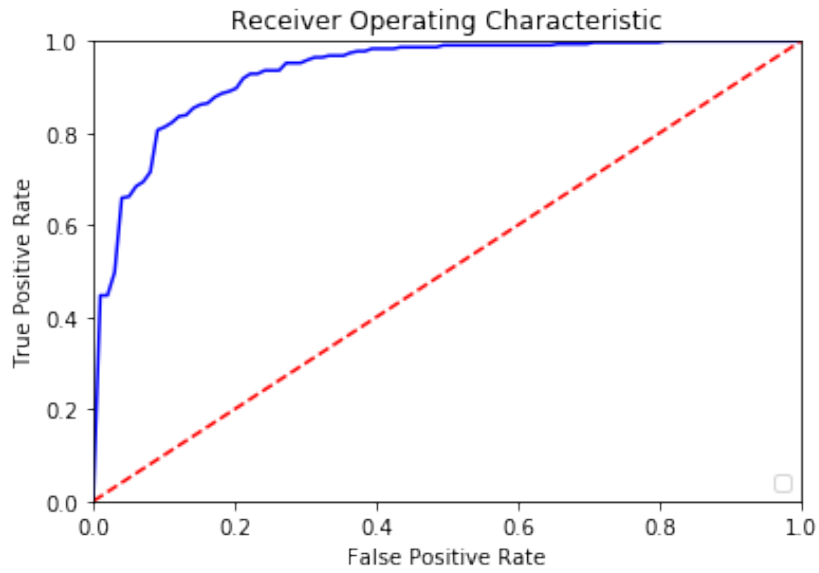


Figure 6.3: AdaBoost - ROC Curve

#### 6.2.4 Random Forests

Training Time: 0.5785s  
Testing Time: 0.018771s  
Accuracy: 98.9936%



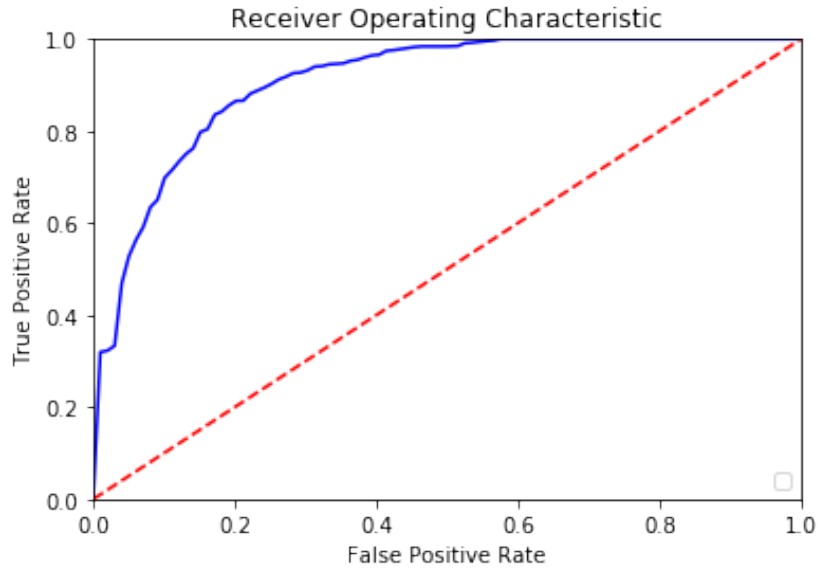


Figure 6.4: Random Forests - ROC Curve

### 6.3 Discussion of Results

All the ML censor models show remarkable performance. All the models have been trained with high accuracy. After being trained efficiently, the models have been beaten by testing them with GAN generated dataset. All models recognize the derived traffic of GAN as normal traffic. This was the main goal of this research, to fool the classifier into thinking that the GAN generated traffic is infact normal skype traffic.

The training and testing times for each model has been calculated. As can be seen, Adaboost and Random forest took around half a second to get trained over the given dataset. Training time for Decision tree is the fastest. The time for trainign and testing depends on the size of the dataset. The more the dataset, the more the time taken for mdoel training.

Accuracy of all the models was above 95%, Random forest showed highest accuracy. However, this accuracy can be further enhanced to 98-99% by simply increasing the training dataset. Although this will further increase the training time, but it would result in more accurate results.

The hardest model to beat still remains to be the Asaost model. However, as mentioned above, the model can be beaten with more efficiency if

the dataset is enhanced. Training an ML censor with 300 training examples is not enough.

## 6.4 Passing GAN Traffic through Censor Models

Once the ML censor models had been implemented and trained accurately, the next step was to test the GAN generated traffic and have the Models mark them as Normal traffic. This was done by extracting features from the pcap generated by our traffic generator.

The models classified the 300 flows given to it as either class and the percentage classified as each class is given below

<b>Classifier</b>	<b>Classified as Normal</b>	<b>Classified as DeltaShaper</b>
<b>Decision Trees</b>	97.333	2.66
<b>RandomForest</b>	99	1
<b>XGBoost</b>	99.33	0.66
<b>AdaBoost</b>	97	3

Table 6.1: Our GAN traffic through Censors

As seen in the table, almost all of our traffic went through without being detected.

These results show that indeed, a censor model can be beaten if The traffic is modelled in a way that it appears as normal traffic. This modelling can be excelled using Generative Adversarial Networks.

# Chapter 7

## Conclusion and Future Work

This chapter concludes and summarizes the thesis and discusses the future prospect of this research.

### 7.1 Conclusion

The rising trend of censorship in the world is a direct attack on the right of every individual of unbarred information access. With machine learning at its peak of research and implementation, defying these censorship models has become difficult.

This research work has not only proved that Protocol tunneling can be successfully implemented by deploying machine learning and Generative adversarial networks, but it has also shown the censor models the loopholes in their implementation. Using GAN, we have not only implemented some best known censor models, but also beaten them by exploiting their detection strategies

We have also designed a traffic generator, which can be used to generate network multimedia packets through the features proposed by GAN. We tested 300 gan generated flows and passed them through our censor models. All censor models detected it as normal skype traffic with great efficiency.

### 7.2 Future Work

Our thesis can be extended in a number of directions. Following are some areas that can be explored further:

- Developing ML-based censorship models that can distinguish GAN-generated traffic from original traffic
- Build a multimedia communication network using the above implementation and send payload for commercial purposes
- Using more datasets to enhance the accuracy of Censor Models and of GAN
- using Packet-length based feature models to beat ML censor models

Since GAN and protocol tunneling is a growing field, there is always room for improvement. Unless a practical system reaches an accuracy of 100, the system can be improved further by better training and testing. There is no end to it. With Machine learning, sky is the limit.

# Bibliography

- [1] D. Barradas, N. Santos, and L. Rodrigues, “Effective detection of multimedia protocol tunneling using machine learning,” 08 2018.
- [2] S. Aryan, H. Aryan, and J. A. Halderman, “Internet censorship in iran: A first look,” in *FOCI*, 2013.
- [3] D. Levine and T. Palfrey, “The paradox of voter participation? a laboratory study,” *UCLA Department of Economics, Levine’s Bibliography*, vol. 101, 01 2006.
- [4] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The second-generation onion router,” *Paul Syverson*, vol. 13, 06 2004.
- [5] D. Barradas, N. Santos, and L. Rodrigues, “Deltashaper: Enabling unobservable censorship-resistant tcp tunneling over videoconferencing streams,” *Proceedings on Privacy Enhancing Technologies*, vol. 2017, 10 2017.
- [6] G. de Seta, *Great Firewall, China*, pp. 232–235. 01 2016.
- [7] A. Rehman, T. Saba, T. Mahmood, Z. Mehmood, M. Shah, and A. Anjum, “Data hiding technique in steganography for information security using number theory,” *Journal of Information Science*, vol. 45, no. 6, pp. 767–778, 2019.
- [8] S. Wang, “Generative adversarial networks (gan): A gentle introduction [updated],” 04 2017.
- [9] A. Houmansadr, W. Zhou, M. Caesar, and N. Borisov, “Sweet: Serving the web by exploiting email tunnels,” *IEEE/ACM Transactions on Networking*, vol. PP, 11 2012.
- [10] C. Brubaker, A. Houmansadr, and V. Shmatikov, “Cloudtransport: Using cloud storage for censorship-resistant networking,” vol. 8555, pp. 1–20, 07 2014.

- [11] B. Hahn, R. Nithyanand, P. Gill, and R. Johnson, “Games without frontiers: Investigating video games as a covert channel,” pp. 63–77, 03 2016.
- [12] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson, “Blocking-resistant communication through domain fronting,” *Proceedings on Privacy Enhancing Technologies*, vol. 2015, 06 2015.
- [13] A. Houmansadr, T. Riedl, N. Borisov, and A. Singer, “Ip over voice-over-ip for censorship circumvention,” 07 2012.
- [14] R. McPherson, A. Houmansadr, and V. Shmatikov, “Covertcast: Using live streaming to evade internet censorship,” *Proceedings on Privacy Enhancing Technologies*, vol. 2016, 07 2016.
- [15] S. Li, M. Schliep, and N. Hopper, “Facet: Streaming over videoconferencing for censorship circumvention,” *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 163–172, 11 2014.
- [16] D. Barradas, N. Santos, L. Rodrigues, and V. Nunes, “Poking a hole in the wall: Efficient censorship-resistant internet communications by parasitizing on webrtc,” in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS ’20*, (New York, NY, USA), p. 35–48, Association for Computing Machinery, 2020.
- [17] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus, “Deep generative image models using a laplacian pyramid of adversarial networks,” *CoRR*, vol. abs/1506.05751, 2015.
- [18] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. N. Metaxas, “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks,” *CoRR*, vol. abs/1612.03242, 2016.
- [19] G. Ye, Z. Tang, D. Fang, Z. Zhu, Y. Feng, P. Xu, X. Chen, and Z. Wang, “Yet another text captcha solver: A generative adversarial network based approach,” (New York, NY, USA), Association for Computing Machinery, 2018.
- [20] M. Yurt, S. U. Dar, A. Erdem, E. Erdem, K. K. Oguz, and T. Çukur, “mustgan: multi-stream generative adversarial networks for mr image synthesis,” *Medical Image Analysis*, vol. 70, p. 101944, 2021.

- [21] A. Aggarwal, M. Mittal, and G. Battineni, “Generative adversarial network: An overview of theory and applications,” *International Journal of Information Management Data Insights*, vol. 1, no. 1, p. 100004, 2021.
- [22] H. Shi, J. Dong, W. Wang, Y. Qian, and X. Zhang, “SSGAN: secure steganography based on generative adversarial networks,” *CoRR*, vol. abs/1707.01613, 2017.
- [23] M. Rigaki and S. Garcia, “Bringing a gan to a knife-fight: Adapting malware communication to avoid detection,” pp. 70–75, 2018.
- [24] L. Rokach and O. Maimon, *Decision Trees*, vol. 6, pp. 165–192. 01 2005.
- [25] A. Cutler, D. Cutler, and J. Stevens, *Random Forests*, vol. 45, pp. 157–176. 01 2011.
- [26] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” pp. 785–794, 08 2016.
- [27] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*, vol. 398. John Wiley & Sons, 2013.
- [28] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [29] R. J. Williams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks,” *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.
- [30] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240, ACM, 2006.