5-3-2019

Internet and Tor Traffic Classification Using Machine Learning

Siddharth Palsambkar
ssp6148@rit.edu

Internet and Tor Traffic Classification Using Machine Learning

RIT | Rochester Institute of Technology

Under Supervision of:

Prof. Joseph Nygate, PhD

Submitted By:

Siddharth Palsambkar

Thesis submitted in partial fulfillment of requirements of Degree of Master of Science in

Telecommunication Engineering Technology

Department of Electrical, Computer & Telecom Engineering Technology

College of Engineering Technology

Rochester Institute of Technology

Rochester, NY 14623

May 3, 2019

Committee Approval

Joseph Nygate, Ph.D. Date

Associate Professor, Thesis advisor

William P. Johnson, J.D. Date

Graduate Program Director, MS Telecommunications Engineering Technology Program

Mark Indelicato Date

Associate Professor

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor Prof. Joseph Nygate, Associate Professor at the Rochester Institute of Technology for the continuous support during my graduate thesis and related research. The door to his office was always open whenever I had questions about my research and writing. He encouraged me to work independently on my research while regularly providing me the much-needed advice and insights.

I would also like to acknowledge Mr. Chris Brown, Systems Administrator, Laboratory Manager and Adjunct Faculty at the Rochester Institute of Technology for providing his support in setting up and maintaining the lab during the research.

Finally, I must express my very profound gratitude to my parents, teacher and friends for providing me with unfailing support and encouragement throughout my graduate study. This achievement would not have been possible without them. Thank you.

Abstract

Privacy has always been a concern over the internet. A new wave of privacy networks struck the world in 2002 when the TOR Project was released to the public. The core principle of TOR, popularly known as the onion routing protocol, was developed by the 'United States Naval Research Laboratory' in the mid-1990s. It was further developed by 'Defense Advanced Research Projects Agency'. The project that started as an attempt to create a secured communication network for the U.S. Intelligence was soon released as a general anonymous network. These anonymous networks are run with the help of volunteers that serve the physical need of the network, while the software fills up the gaps using encryption algorithms. Fundamentally, the volunteers along with the encryption algorithms are the network. Once a part of such a network, the identity, and activity of a user is invisible. The users remain completely anonymous over the network if they follow a few steps and rules. As of December 2017, there are more than 3 million TOR users as per the TOR Project's website. Today, the anonymous web is used by people of all kinds. While, some just want to use it to make sure nobody could possibly spy on them, others are also using it to buy and sell things. Thus, functioning as a censorship-resistant peer-to-peer network.

Through this thesis, we propose a novel approach to identifying traffic and without sacrificing the privacy of the Tor nodes or clients. We recorded traffic over our own Tor Exit and Middle nodes to train Decision Tree classifiers to identify and differentiate between different types of traffic. Our classifiers can accurately differentiate between regular internet and Tor traffic while can also be combined together for detailed classification. These classifiers can be used to selectively drop traffic on a Tor node, giving more control to the users while providing scope for censorship.

Table of Contents

List of Figures

Introduction

The internet that most of the general population uses is also known as ‘surface web’ which is just a part of the ‘World Wide Web (www)’. This is the visible part of the world wide web which is indexed by most of the search engines and can be accessed using a regular web browser with basic settings. The surface web consists of all the websites we use regularly, like google.com, facebook.com and other websites specific to different products and services. But, there also exists a hidden internet which is known as deep web. It is known as the ‘hidden web’ because it cannot be accessed easily. It requires a special browser and special settings to be accessed. This hidden internet is part of the anonymous networks like TOR and I2P. A user has to be a part of such a network to be able to access the content available on such a network. Websites hosted on these networks are known as deep web websites and are intended to keep it's content private while keeping the identity of the user accessing it anonymously. Everything from news, marketplace, blogs, video streaming, and social media websites exist on the deep web.

Research Problem

There also exists a part of the deep web where illicit websites are hosted. Since such activities are carried out in the dark, this part of the deep web is known as the ‘Dark Net’. Dark Net websites are infamous for hosting all kinds of illegal contents. The websites range from drugs, arms and ammunitions marketplaces, leaked information websites and other kinds of illegal activities. It is getting easier for kids to purchase drugs over the darknet. Leaked information like credit card details and Social Security information is also sold freely over the network. One huge part of it belongs to websites hosting ‘child pornography’ websites. Since using a regular credit/debit card to purchase any of the products or services being sold on the darknet can compromise the user’s and the website owner’s identity, these transactions are most often completed using cryptocurrencies.

Since the traffic over the nodes remains encrypted, it is difficult to censor any type of content over the Tor network. This is especially of concern to the users hosting Exit Nodes. These users are very likely to receive DMCA notices from BitTorrent and other traffic carrying copyright content flowing through it. There is no system in place for a host to control the traffic flowing to their owned node.

The research focuses on implementing a way to classify traffic without compromising the anonymity of the nodes, users or their traffic. Most techniques used try to find weaknesses in the Tor infrastructure to then exploit them. Through this research, we use a different approach by using Machine Learning to try and understand how the Tor network behaves with the different types of traffic flowing through it. This can be then compared to the activity of a node on the regular internet to try and develop and train algorithms that can differentiate between them.

Review of Literature and Deficiencies

The Federal Bureau of Investigation posted an article on their website in 2016 explaining Dark Net websites, what they are and what countermeasures are law enforcement authorities taking against them. Within this article, the organization explains how an international coalition of law enforcement agencies from five countries around the world named Five Eyes Law Enforcement Group (FELEG) are sharing intelligence to bust Dark Net related crimes (FBI, 2016). However, a review of the literature reveals that there is much to be explored in this field.

The existing research on TOR revolves around the statistics leading to the concentration of services. Most drugs on the Dark Net is sold in the United States of America. The greatest number of sellers are also from the United States of America (Dolliver, 2015). A paper published in the Journal of Computer Virology and Hacking Techniques in 2015 shows how a Botnet's Command and Control servers can be hidden in the TOR network and used to control several thousand or even millions of bots in that botnet to execute click fraud, data mining, bitcoin mining or perform a DDOS attack (Kang, 2015). The TOR network also has mechanisms to ban an Exit Node if it is known to tamper information before relaying it to its subsequent node. It does by verifying the signature of response that every TOR server creates when it replies to a query. If the signature received is different, the TOR network automatically bans the Exit node and thus no traffic is relayed through that node (Wagner, Wagener, State, Dulaunoy, & Engel, 2012).

There also exist research on deanonymizing traffic on the TOR network. A journal article explains how the HSDir (Hidden Services Directory) works and how it can be harvested to find more information about the various hidden services hosted on the TOR network (Biryukov, Pustogarov, & Weinmann, 2013). Privacy over the TOR network is also compromised when active plugins are active over a client machine or hidden server. Browser-based attacks can be carried out on such clients and server using HTML, Javascript and flash to expose their identity (Abbott, Lai, Lieberman, & Price, 2007).

Technical Background

The Tor Network

A conference paper submitted to the International Symposium on Privacy Enhancing Technologies Symposium explains the working of the TOR network. It is a kind of a mesh network consisting of volunteers serving as nodes. These nodes function as relay networks that is the fundamental way in which TOR functions (McCoy, Bauer, Grunwald, Kohno, & Sicker, 2008). TOR is an intelligent network that makes use of its relay nodes and incredibly complex cryptography to make sure that the user's identity remains anonymous.

The network is made up of three kinds of nodes: Entry Node, Intermediate Node, and the Exit Node. When a user connects to the TOR network, the network protocol looks into its database of available TOR nodes and assigns a combination of Entry, Intermediate and Exit nodes for the connection (McCoy, Bauer, Grunwald, Kohno, & Sicker, 2008). These nodes can be located several thousand miles apart across the globe. The user is connected to the Entry Node whereas the website server or service he is trying to connect to or access is connected to the end of the Exit Node. The TOR network also makes sure that the user's machine carries out secure key exchanges with all the three nodes. The key exchanges are executed in such a way that, the host has a unique key combination with each of the nodes (Abbott, Lai, Lieberman, & Price, 2007). When the user is sending traffic to the web server, which is general inquiry for the content they are requesting for, the host machine encrypts the data with the Exit Node's encryption key, followed by the Intermediate Node's key and then finally with the Entry Node's key. Thus, the data is now encrypted thrice with three different keys that not entirely known by any of the relay nodes (Abbott, Lai, Lieberman, & Price, 2007). A simple illustration can be used to explain this:

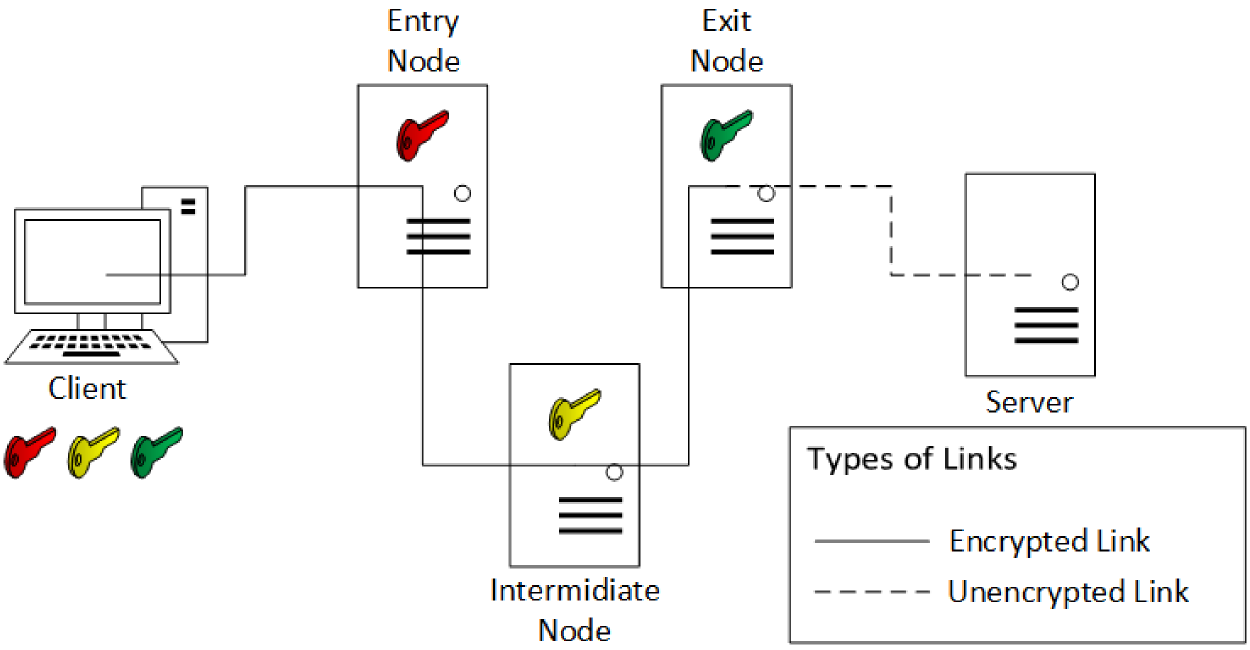


Figure 1: Tor Architecture

As can be seen above, each relay node has its own unique key which is exchanged with the host in a way that at any point in the connection, only the host machine has access to all the three keys. When the data that is encrypted thrice is sent over the network, it is first received by the Entry Node, which strips its key and forwards the packet to the Intermediate node, which then strips its key and forwards it to the exit node. The exit node on receiving the packet, strips its own key and forwards the data to the web server it is intended for (Abbott, Lai, Lieberman, & Price, 2007). When response data is traveling back from the web server to the host machine, the data is first received at the Exit Node, which encrypts the data using its key and forwards it to the Intermediate Node which encrypts the received data using its key and forwards it to the Entry Node, which encrypts it one last time using its unique key and forwards it to the host machine. Since the host machine has all the three keys available with it, it sequentially strips down all the three keys to reveal the data. As can be observed in the figure and explanation above, the data is always kept encrypted on all links in the network except one. The data flow between the Exit

Node and the web server occurs over an unsecured medium. This is where, the possibilities of potential attacks on the TOR network exists (McCoy, Bauer, Grunwald, Kohno, & Sicker, 2008). Active plugins operate in ways different than regular programs. They can run in the background can report user activity without even the user noticing it. Most attacks on the TOR network are carried out using active plugins. One paper published with the International Workshop on Privacy describes a few such attacks. It mentions how to browse the internet anonymously; a user must be using an HTTP proxy like 'Privoxy' so that traffic would be diverted over TOR instead of the regular internet. This is because active plugins not always use the browser's proxy to send data (Abbott, Lai, Lieberman, & Price, 2007). The article further goes on to explain the first attack which falls in the category of browser-based attacks. The attack is executed as follows:

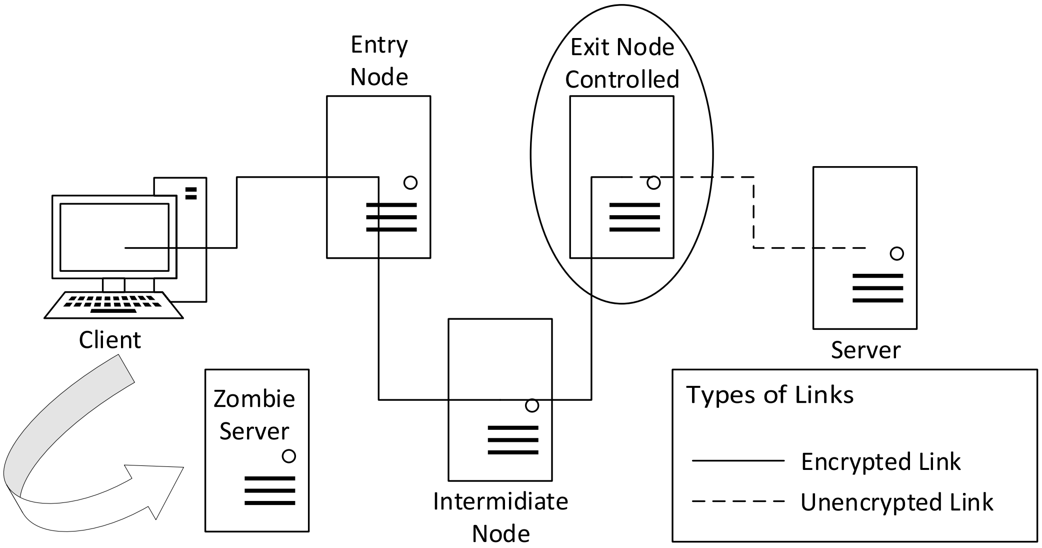


Figure 2: Tor Browser Based Attack

The above figure can be held as a reference for the explanation. The attacker sets up a malicious exit node in the TOR network to modify the HTTP traffic. Since the exit node is connected to a web server, the attacker modifies all the packets from the server to the client by adding an invisible ‘iframe’ with a unique cookie along with a referenced to a malicious web server owned by the attacker. When this frame is rendered by the client’s browser, if the flash plugin is still active on the browser, a flash movie starts playing in the background and the browser sends back the cookie previously received to the malicious web server. The attacker can then identify the user and the website visited using the combination of the unique cookie and the flash connection. In fact, this attack would work on all users connected to a website through the attacker owned exit node and a browser with flash enabled on it (Abbott, Lai, Lieberman, & Price, 2007).

A second attack is mentioned which can identify and locate hidden servers on the Network. The figure below can be used as a reference to better understand the attack:

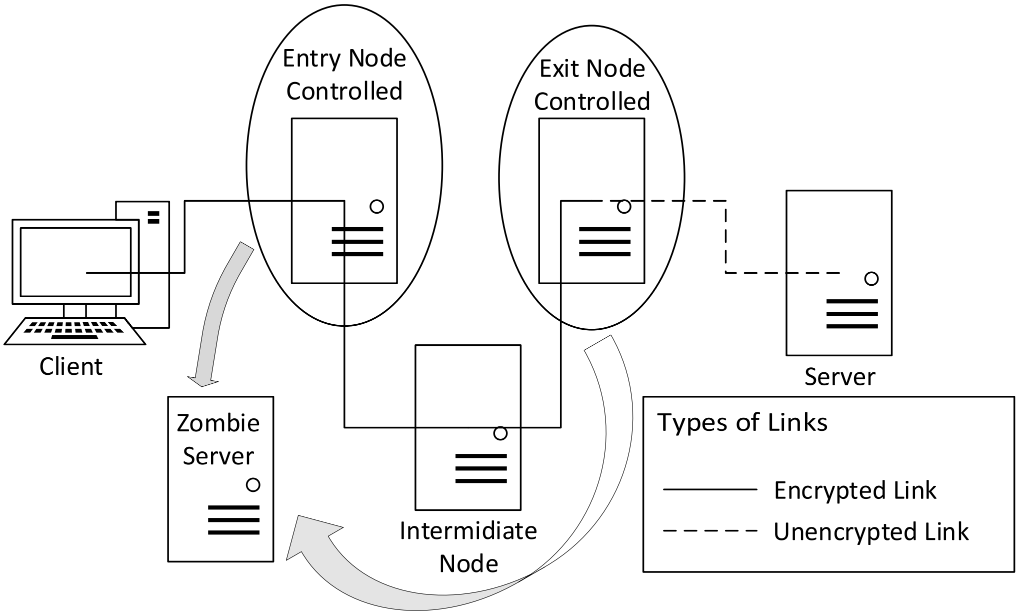


Figure 3: Setup to Identify Hidden Servers

The attacker introduces an exit node in the TOR network along with a malicious web server. The attacker uses his client machine to connect to the website whose location is to be identified. Since TOR chooses a random path to create a connection, the attacker makes multiple attempts to connect to the website until the node owned by him is selected as the exit node for the connection between the client and the website. This can be achieved since TOR treats every new query as a new connection and thus assigns a new path every other time. The attacker can verify that his node is working as the exit node for the connection by using traffic analysis. Once confirmed, the attacker can identify the location of the web server hosting the website using a predecessor attack (Abbott, Lai, Lieberman, & Price, 2007). On carefully studying both these attacks, we can see that both of the attacks make use of the unencrypted link and the exit node which are the weakest links to attack considering the TOR architecture (McCoy, Bauer, Grunwald, Kohno, & Sicker, 2008)

Machine Learning

Just like humans, machines can be trained in doing a task. Using artificial intelligence and computing, systems can be developed that can learn from the input given to them. These systems have the human-like the ability to improve with experience reaching a level where it can then predict outcomes of systems. There are various kinds of algorithms that can be used for machine learning. Which algorithm to use depends completely on what output is expected. A term called 'Supervised Learning' is often used. Supervised Learning is when the training data is a set of variables and both the input and the output object are to be specified. On the other hand, 'Unsupervised Learning' is when we let the algorithm find a pattern in the dataset provided. Unlike supervised learning, unsupervised learning does not have any right or wrong answers, the outcome of the algorithm is just the detection of a pattern. (Bell, 2014).

Any machine learning project has four cycles: 1) Acquisition, 2) Prepare, 3) Process and 4) Report. Data can be acquired from any source. When working with machine learning, the more the data, the better is the outcome. The collected data then must be prepared to be analyzed and once that is done, it is run through the algorithm and the outcome is studied. Though

‘Machine Learning’ sounds a lot like ‘Big Data’, they are very different from each other. While Big Data is used to analyze huge chunks of data to find patterns and statistics, Machine Learning starts with a question (Bell, 2014). The question is something that is to be investigated. It is a more customized approach towards a topic. Thus, there is no one machine learning solution that works for all the projects. It is not a super complex algorithm that can answer any question thrown at it. Rather, it is an algorithm that learns an extensive amount of data for a specific purpose and then predicts an outcome for an event. An important part of Machine Learning is Data Processing. Which large amounts of data, the processing power needed to run that data through the algorithm at the same time is a lot. For basic less processor intensive algorithms, a personal computer can be used. But, for large scale operations, a single processor is not efficient. Thus, a cluster of machines is required to process the exorbitantly high volume of data. The machines on such a clustered are preferred to be on the same local network to avoid delays and lags in data transfers. When machines are used in a cluster, they process the data in different parts and then sync up their results. Thus, all the processing power is used in the most efficient way possible. Another great way is to use ‘Cloud-Based Services’ provided by companies like Amazon and Rackspace. These services provide servers with variable processing power. They provide an option to increase or decrease the number of machines required to process a certain task. The only downside is that such a service might sometimes cost a lot (Bell, 2014).

Measurement Instruments

The experiment was conducted using multiple Tor nodes. Multiple instances of Exit and Middle nodes were hosted to collect data and analyze traffic. Each node was individually configured to only allow a certain type of traffic to flow through it. Certain Exit nodes were hosted with the default ‘exit policies’ while others were implemented with ‘reduced exit policies’ to reduce the amount of ‘BitTorrent’ and ‘Copyrighted’ traffic from flowing over it.

Procedure

The research was phased in two parts. In the first part, multiple nodes were set up on the Tor network to collect data. These devices would be set up as Relay or Exit. When a node is set up as a Relay, it is always selected as either Entry or Middle Node on the TOR network. Majority of the devices in the cluster were set up in an Exit Mode which gives them a chance to function as an Exit Node. Whenever one of the machines is selected as the exit node, we start sniffing on the node to inspect the outgoing packets. On analyzing the packets, the website being visited can be easily detected. These packets are then collected and added to a database which also consists of regular internet traffic generated on the same node. The traffic was then labeled to easily differentiate between them while we continue to process it before applying machine learning to it. The same methodology was applied to the traffic recorded on the ‘Middle Nodes’

Data Analysis and Interpretation

Data collected from the Tor Nodes were initially analyzed and filtered to avoid unnecessary information being added to the algorithm. Only the correct and required data was used to form a data set used to form an algorithm for Machine Learning. A ‘Thematic Analysis’ approach was used to identify, pinpoint and record patterns within the collected data. A ‘Decision Tree Classifier’ was trained to differentiate traffic in each of the tested scenarios. The decision tree was then used to plot a graphical tree to easily understand how certain decisions were obtained to solve the problem. Rules were also generated from the same tree, which could be used to be implemented in traffic sorting programs for a particular application.

The Setup & Experiments

Classifying Tor vs Surface-Net Traffic:

The first part of the experiment was implemented by hosting a node on the internet and capturing traffic through it. We would further refer to ‘Internet’ as ‘Surface-Net’ to easily separate it from the ‘Tor Network’. For the next stage, multiple instances of Tor Exit Nodes were hosted on Ubuntu machines. Traffic from one such instance was with the default ‘Exit Policies’ was used during the experiment. The ‘nyx’ application was used to monitor the nodes, keep track of the flags obtained and observe the performance of the node.

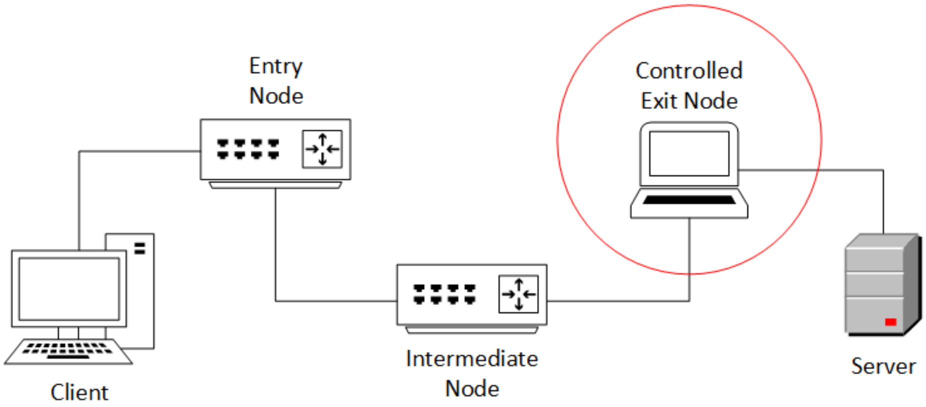
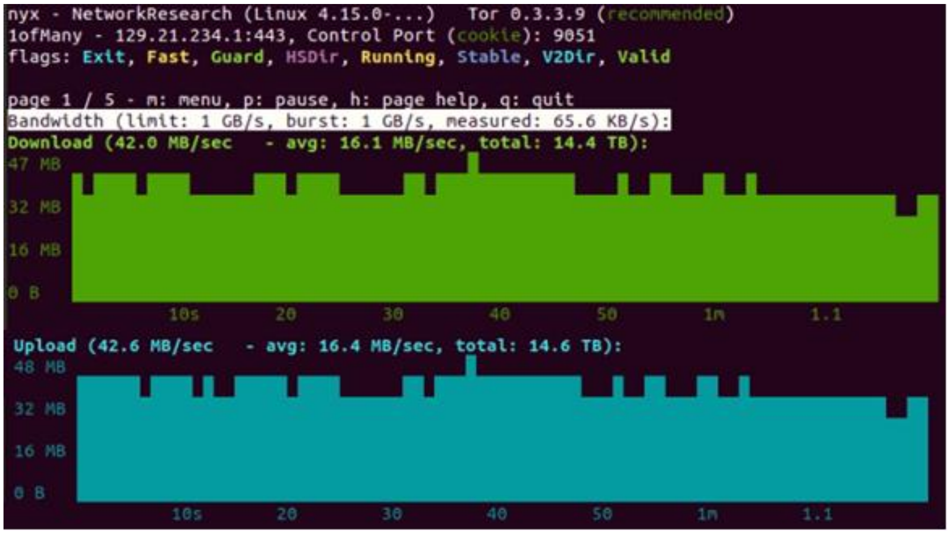


Figure 4: Setup for Classifying Tor vs Surface-Net Traffic

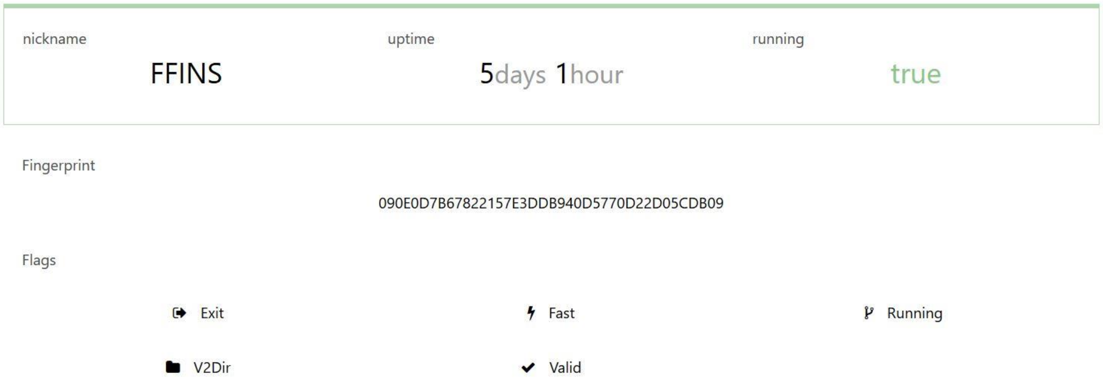


```
nyx - NetworkResearch (Linux 4.15.0-...)   Tor 0.3.3.9 (recommended)
1ofMany - 129.21.234.1:443, Control Port (cookie): 9051
flags: Exit, Fast, Guard, HSDir, Running, Stable, V2Dir, Valid

page 1 / 5 - m: menu, p: pause, h: page help, q: quit
Bandwidth (limit: 1 GB/s, burst: 1 GB/s, measured: 65.6 KB/s):
Download (42.0 MB/sec   - avg: 16.1 MB/sec, total: 14.4 TB):
```

```
Upload (42.6 MB/sec   - avg: 16.4 MB/sec, total: 14.6 TB):
```

Figure 5: Tor Exit Node Statistics



nickname
FFINS

uptime
5days 1hour

running
true

Fingerprint
090E0D7B67822157E3DDB940D5770D22D05CDB09

Flags
Exit
Fast
Running
V2Dir
Valid

Figure 6: Tor Exit Node Flags and Uptime

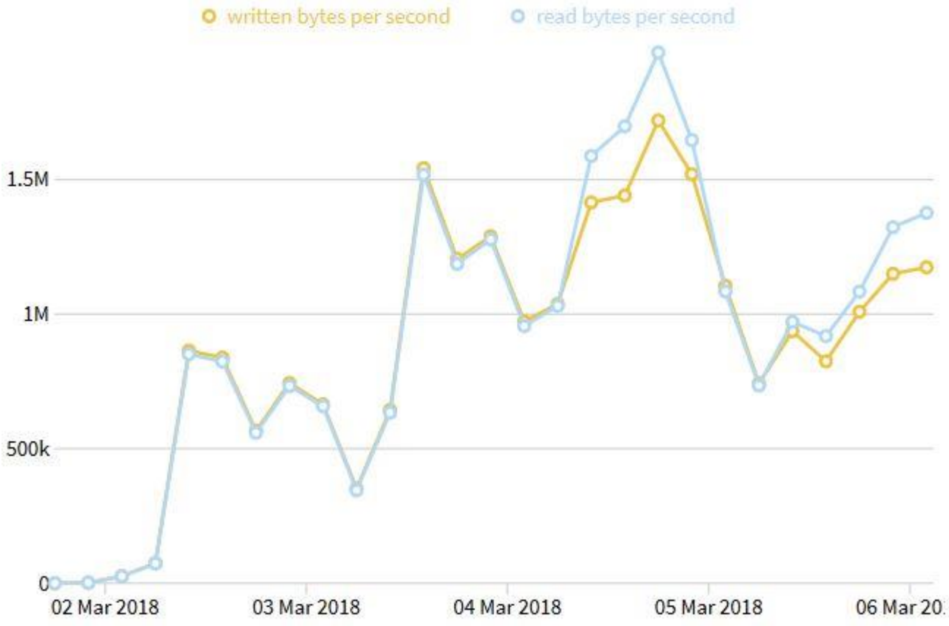
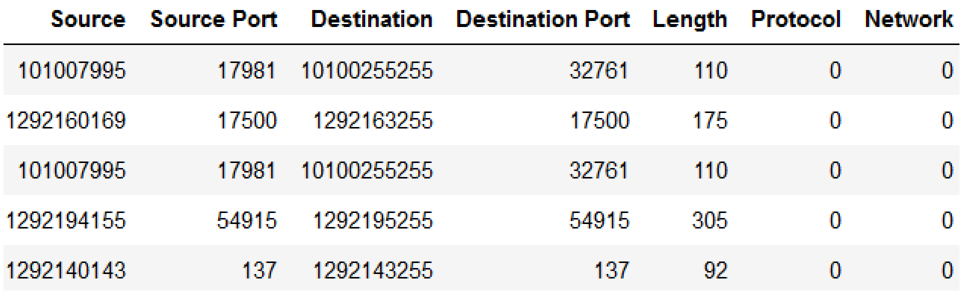


Figure 7: Tor Exit Node Traffic During Uptime

Wireshark was used to capture traffic over the nodes. The captured traffic was filtered and used to create a ‘Training Set’ along with a ‘Test Set’. ‘Jupyter Notebook’, which is part of the ‘Anaconda’ suite was used to create Python notebooks and run scripts. The necessary libraries were then imported and used to further process the data and create a decision tree algorithm for the required application.



Source	Source Port	Destination	Destination Port	Length	Protocol	Network
101007995	17981	10100255255	32761	110	0	0
1292160169	17500	1292163255	17500	175	0	0
101007995	17981	10100255255	32761	110	0	0
1292194155	54915	1292195255	54915	305	0	0
1292140143	137	1292143255	137	92	0	0

Figure 8: Traffic Captured Over The Node

Tor TCP vs UDP Traffic:

The same setup was used for this experiment with a different approach. The traffic captured was only captured on the Exit Node to be categorized into TCP and UDP traffic.

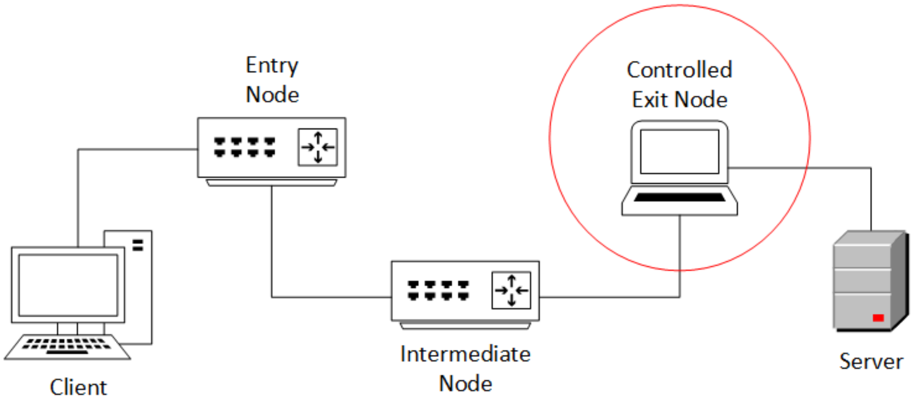


Figure 9: Setup for TCP vs UDP Classification

As can be seen in the following graphs, the volume of UDP traffic is significantly higher than the TCP traffic, thus the same number of captured TCP and UDP packets were then used to create a ‘Training Set’ and ‘Test Set’. As before, a Python notebook was used to import and process the data to create a decision algorithm.

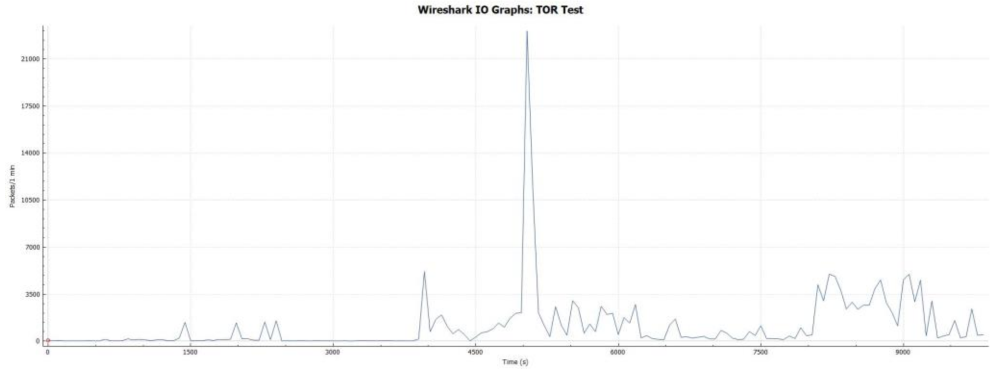


Figure 10: TCP Traffic Through Exit Node

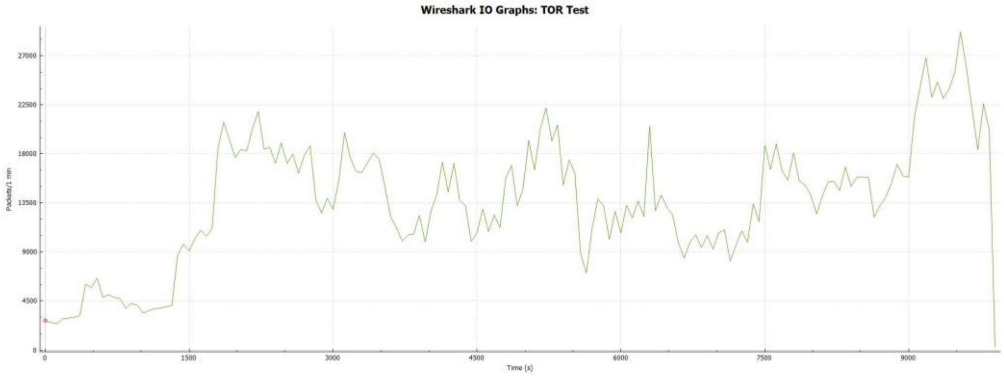


Figure 11: UDP Traffic Through Exit Node

Resolving the Location from IP addresses on Tor

Another possible requirement could be to identify the location of the nodes connected to our node. This can be done for both security and statistical reasons. The node can be attacked to gain access, to be then used as a botnet for malicious activity.

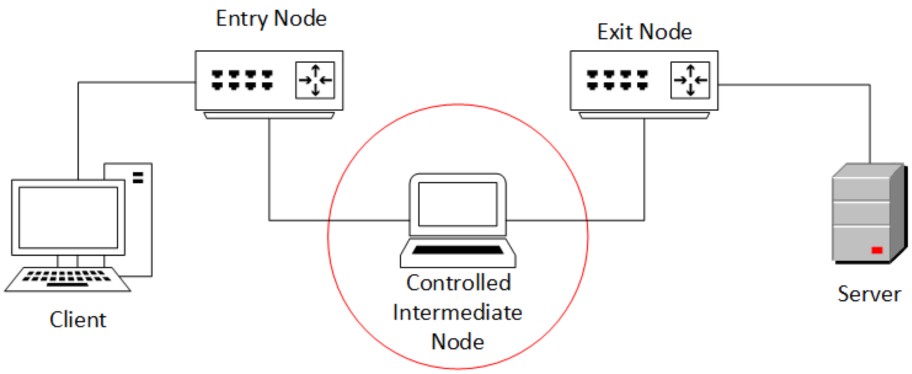
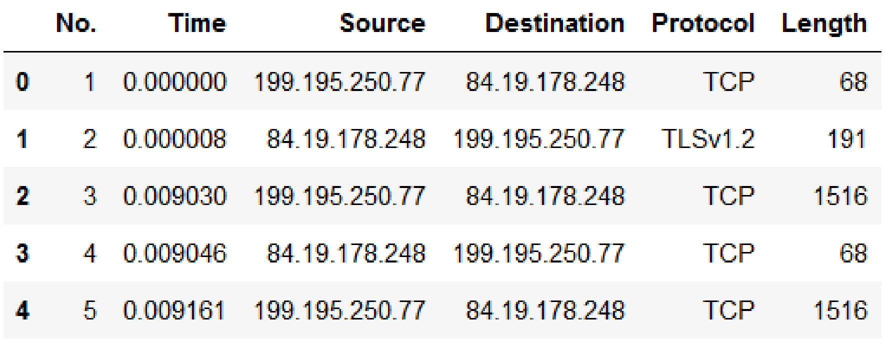


Figure 12: Setup for Resolving Location from IP Addresses

The packets were captured using Wireshark and imported into a Jupyter notebook to be processed with the help of ‘GeoIP2’ database.



	No.	Time	Source	Destination	Protocol	Length
0	1	0.000000	199.195.250.77	84.19.178.248	TCP	68
1	2	0.000008	84.19.178.248	199.195.250.77	TLSv1.2	191
2	3	0.009030	199.195.250.77	84.19.178.248	TCP	1516
3	4	0.009046	84.19.178.248	199.195.250.77	TCP	68
4	5	0.009161	199.195.250.77	84.19.178.248	TCP	1516

Figure 13: Traffic Captured on the Intermediate Node

Identifying Port-scan attacks on Tor node:

One common form of threat to all nodes on the internet is port-scans. These are in fact the first step in most major attack strategies. Tor nodes are also vulnerable to port scans. The question was to identify whether a port scan was carried out by a regular host on the internet or a node on the Tor network. The dataset was collected by capturing traffic from both a regular internet connection and that through the Tor network. The host was first used to carry out a port scan attack on our controlled Tor node through the internet and then the experiment was repeated by connecting the host to the Tor network (proxy chains) to carry out the same attack.

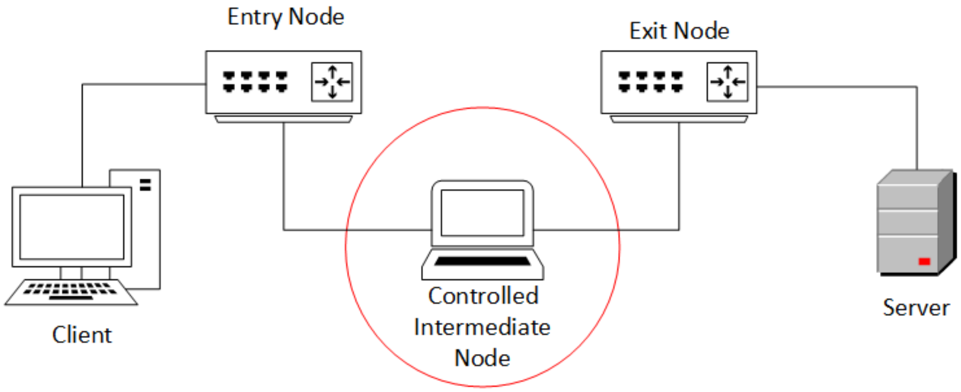
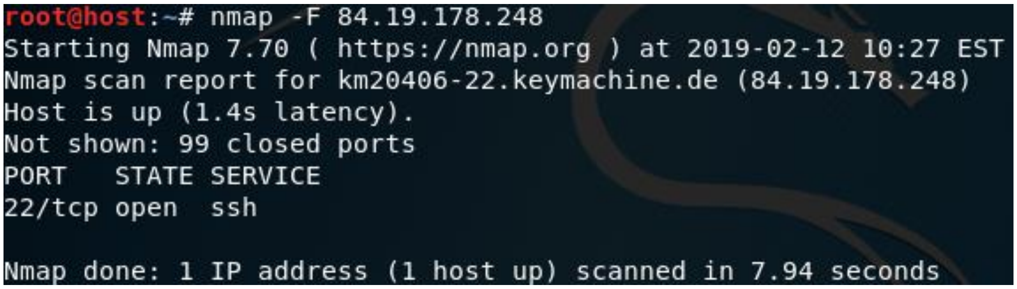


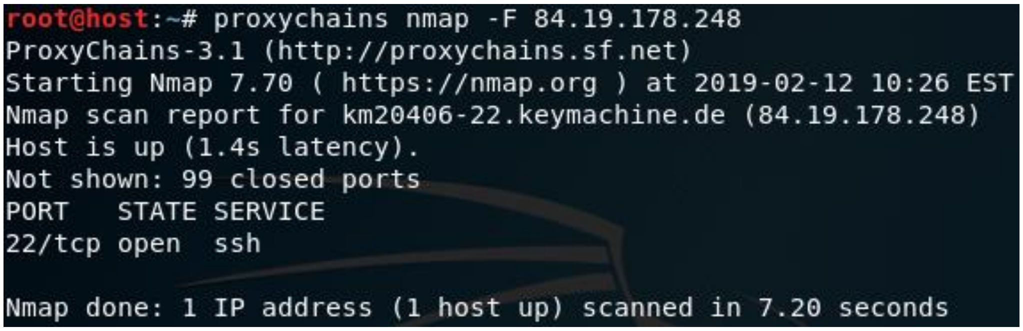
Figure 14: Setup for Identifying Port-Scan Attacks on Tor Node



```
root@host:~# nmap -F 84.19.178.248
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-12 10:27 EST
Nmap scan report for km20406-22.keymachine.de (84.19.178.248)
Host is up (1.4s latency).
Not shown: 99 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 7.94 seconds
```

Figure 15: Running ‘NMap’ Scan from A Host on the Internet

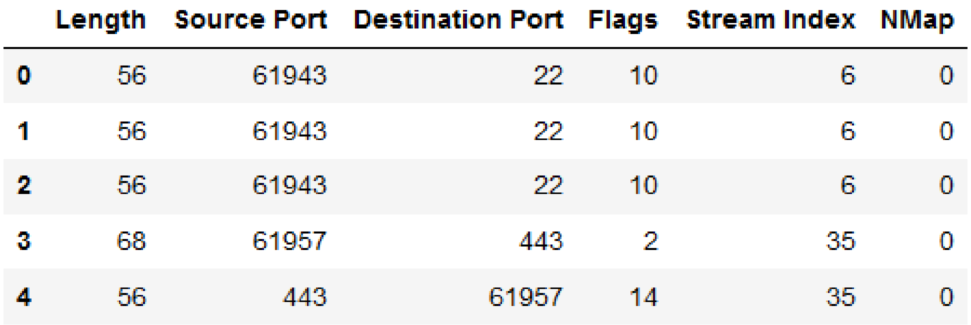


```
root@host:~# proxychains nmap -F 84.19.178.248
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-12 10:26 EST
Nmap scan report for km20406-22.keymachine.de (84.19.178.248)
Host is up (1.4s latency).
Not shown: 99 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 7.20 seconds
```

Figure 16: Running ‘NMap’ Scan from a Host on Tor Network

Packets were captured in both cases using Wireshark. The ‘Training Set’ and ‘Test Set’ were created by using packets from both stages of the experiment. A Python notebook was then used to process the data and create an algorithm to classify the scans.



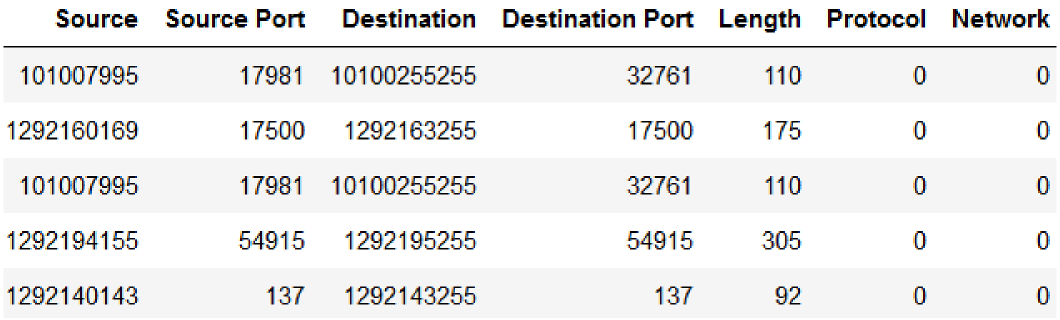
	Length	Source Port	Destination Port	Flags	Stream Index	NMap
0	56	61943	22	10	6	0
1	56	61943	22	10	6	0
2	56	61943	22	10	6	0
3	68	61957	443	2	35	0
4	56	443	61957	14	35	0

Figure 17: Captured Traffic on the Tor Node

The Results

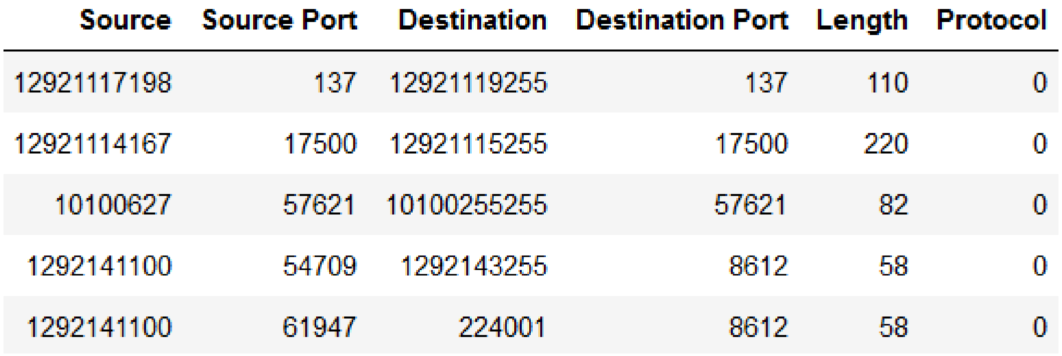
Classifying Tor vs Surface-Net Traffic:

Ten thousand packets of the Tor and Surface-Net traffic were used to form the Training Set, whereas two-thousand packets of each were used to form the Test Set. The Attributes filtered attributes used were ‘Source IP’, ‘Source Port’, ‘Destination IP’, ‘Destination Port’, ‘Packet Length’, ‘Protocol’ and ‘Network (Surface-Net/Tor)’. For ‘Network’, 0 represents the Tor network and 1 represents Surface-Net traffic. The decimal points in the IP addresses were removed to avoid processing errors in the algorithm. Twenty-thousand (ten-thousand each of Surface-net and Tor) packets were used in the ‘Training-Set’, while two-thousand packets (thousand each of Surface-net and Tor) were used in the ‘Test-Set’.



Source	Source Port	Destination	Destination Port	Length	Protocol	Network
101007995	17981	10100255255	32761	110	0	0
1292160169	17500	1292163255	17500	175	0	0
101007995	17981	10100255255	32761	110	0	0
1292194155	54915	1292195255	54915	305	0	0
1292140143	137	1292143255	137	92	0	0

Figure 18: Training Set for Classifying Tor vs Surface-Net Traffic



Source	Source Port	Destination	Destination Port	Length	Protocol
12921117198	137	12921119255	137	110	0
12921114167	17500	12921115255	17500	220	0
10100627	57621	10100255255	57621	82	0
1292141100	54709	1292143255	8612	58	0
1292141100	61947	224001	8612	58	0

Figure 19: Test Set for Classifying Tor vs Surface-Net Traffic

The ‘pandas’ library in Python was used for the manipulation and analysis of the data set. The ‘sklearn’ library was used for the machine learning while using the ‘DecisionTreeClassifier’ for performing binary classification on the dataset. The decision tree was converted in a graphical form using the ‘graphviz’ library. The original tree obtained has a depth of five, but the more important rules can also be seen at much lower levels. Important rules were extracted based on their individual classification ability.

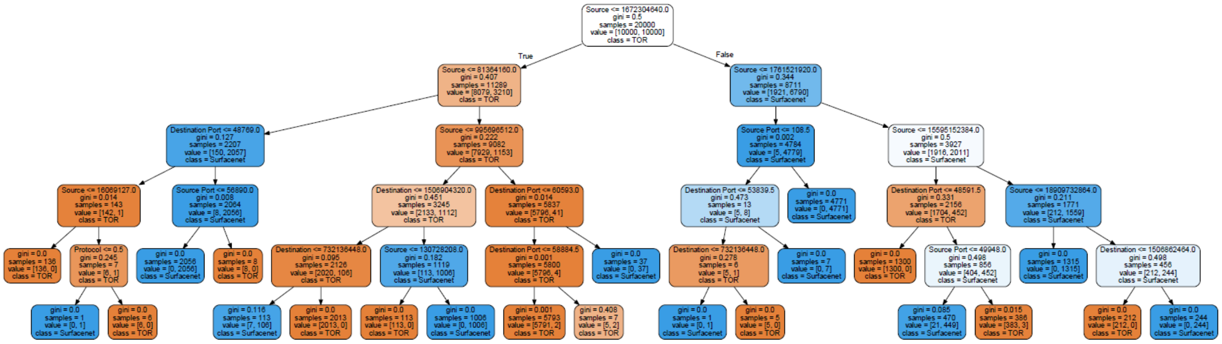

Source <= 1672304640.0 gini = 0.5 samples = 20000 value = [10000, 10000] class = TOR
True
False
Source <= 81364160.0 gini = 0.407 samples = 11289 value = [8079, 3210] class = TOR
Source <= 1761521920.0 gini = 0.344 samples = 8711 value = [1921, 6790] class = Surfacenet
Destination Port <= 48769.0 gini = 0.127 samples = 2207 value = [150, 2057] class = Surfacenet
Source <= 995696512.0 gini = 0.222 samples = 9082 value = [7929, 1153] class = TOR
Source Port <= 108.5 gini = 0.002 samples = 4784 value = [5, 4779] class = Surfacenet
Source <= 15595152384.0 gini = 0.5 samples = 3927 value = [1916, 2011] class = Surfacenet
Source <= 16069127.0 gini = 0.014 samples = 143 value = [142, 1] class = TOR
Source Port <= 56890.0 gini = 0.008 samples = 2064 value = [8, 2056] class = Surfacenet
Destination <= 1506904320.0 gini = 0.451 samples = 3245 value = [2133, 1112] class = TOR
Destination Port <= 60593.0 gini = 0.014 samples = 5837 value = [5796, 41] class = TOR
Destination Port <= 53839.5 gini = 0.473 samples = 13 value = [5, 8] class = Surfacenet
gini = 0.0 samples = 4771 value = [0, 4771] class = Surfacenet
Destination Port <= 48591.5 gini = 0.331 samples = 2156 value = [1704, 452] class = TOR
Source <= 18909732864.0 gini = 0.211 samples = 1771 value = [212, 1559] class = Surfacenet
gini = 0.0 samples = 136 value = [136, 0] class = TOR
Protocol <= 0.5 gini = 0.245 samples = 7 value = [6, 1] class = TOR
gini = 0.0 samples = 2056 value = [0, 2056] class = Surfacenet
gini = 0.0 samples = 8 value = [8, 0] class = TOR
Destination <= 732136448.0 gini = 0.095 samples = 2126 value = [2020, 106] class = TOR
Source <= 130728208.0 gini = 0.182 samples = 1119 value = [113, 1006] class = Surfacenet
Destination Port <= 58884.5 gini = 0.001 samples = 5800 value = [5796, 4] class = TOR
gini = 0.0 samples = 37 value = [0, 37] class = Surfacenet
Destination <= 732136448.0 gini = 0.278 samples = 6 value = [5, 1] class = TOR
gini = 0.0 samples = 7 value = [0, 7] class = Surfacenet
gini = 0.0 samples = 1300 value = [1300, 0] class = TOR
Source Port <= 49948.0 gini = 0.498 samples = 856 value = [404, 452] class = Surfacenet
gini = 0.0 samples = 1315 value = [0, 1315] class = Surfacenet
Destination <= 1506862464.0 gini = 0.498 samples = 456 value = [212, 244] class = Surfacenet
gini = 0.0 samples = 1 value = [0, 1] class = Surfacenet
gini = 0.0 samples = 6 value = [6, 0] class = TOR
gini = 0.116 samples = 113 value = [7, 106] class = Surfacenet
gini = 0.0 samples = 2013 value = [2013, 0] class = TOR
gini = 0.0 samples = 113 value = [113, 0] class = TOR
gini = 0.0 samples = 1006 value = [0, 1006] class = Surfacenet
gini = 0.001 samples = 5793 value = [5791, 2] class = TOR
gini = 0.408 samples = 7 value = [5, 2] class = TOR
gini = 0.0 samples = 1 value = [0, 1] class = Surfacenet
gini = 0.0 samples = 5 value = [5, 0] class = TOR
gini = 0.085 samples = 470 value = [21, 449] class = Surfacenet
gini = 0.015 samples = 386 value = [383, 3] class = TOR
gini = 0.0 samples = 212 value = [212, 0] class = TOR
gini = 0.0 samples = 244 value = [0, 244] class = Surfacenet

Figure 20: Decision Tree for Classifying Tor vs Surface-Net Traffic

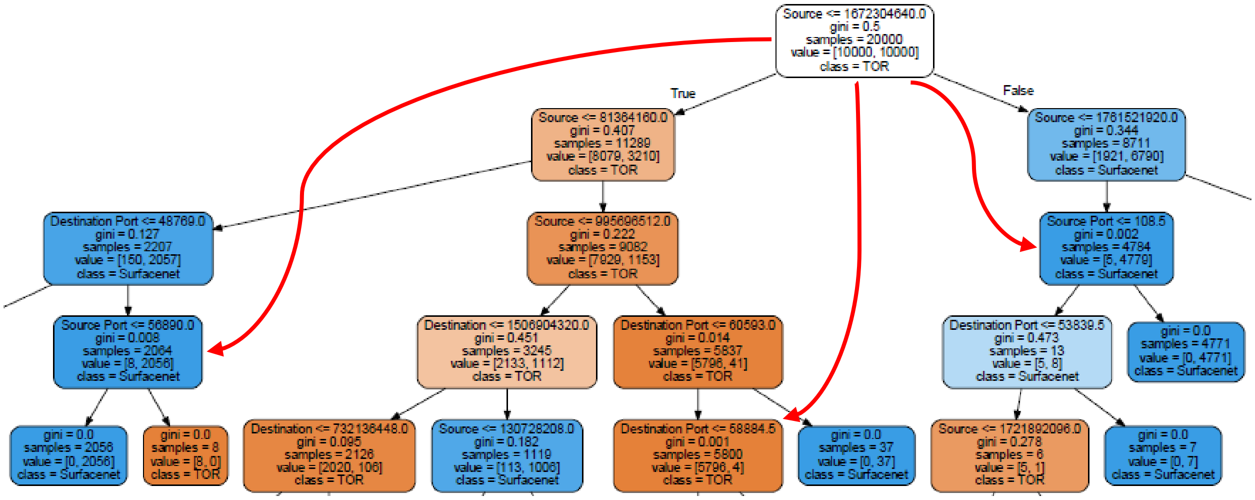
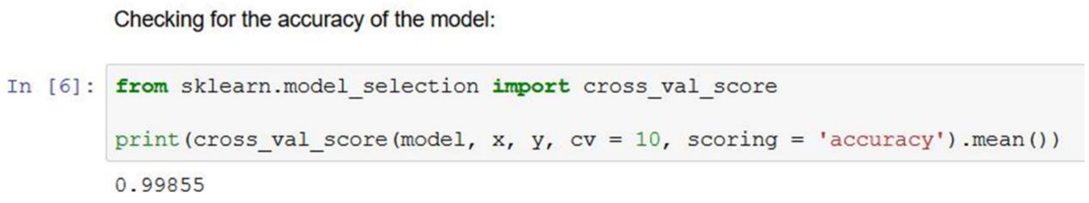

Source <= 1672304640.0 gini = 0.5 samples = 20000 value = [10000, 10000] class = TOR
True
False
Source <= 81364160.0 gini = 0.407 samples = 11289 value = [8079, 3210] class = TOR
Source <= 1761521920.0 gini = 0.344 samples = 8711 value = [1921, 6790] class = Surfacenet
Destination Port <= 48769.0 gini = 0.127 samples = 2207 value = [150, 2057] class = Surfacenet
Source <= 995696512.0 gini = 0.222 samples = 9082 value = [7929, 1153] class = TOR
Source Port <= 108.5 gini = 0.002 samples = 4784 value = [5, 4779] class = Surfacenet
Source Port <= 56890.0 gini = 0.008 samples = 2064 value = [8, 2056] class = Surfacenet
Destination <= 1506904320.0 gini = 0.451 samples = 3245 value = [2133, 1112] class = TOR
Destination Port <= 60593.0 gini = 0.014 samples = 5837 value = [5796, 41] class = TOR
Destination Port <= 53839.5 gini = 0.473 samples = 13 value = [5, 8] class = Surfacenet
gini = 0.0 samples = 4771 value = [0, 4771] class = Surfacenet
gini = 0.0 samples = 2056 value = [0, 2056] class = Surfacenet
gini = 0.0 samples = 8 value = [8, 0] class = TOR
Destination <= 732136448.0 gini = 0.095 samples = 2126 value = [2020, 106] class = TOR
Source <= 130728208.0 gini = 0.182 samples = 1119 value = [113, 1006] class = Surfacenet
Destination Port <= 58884.5 gini = 0.001 samples = 5800 value = [5796, 4] class = TOR
gini = 0.0 samples = 37 value = [0, 37] class = Surfacenet
Source <= 1721892096.0 gini = 0.278 samples = 6 value = [5, 1] class = TOR
gini = 0.0 samples = 7 value = [0, 7] class = Surfacenet

Figure 21: Important Rules for Classifying Tor vs Surface-Net Traffic

As can be seen below, the tree obtained for the experiment yields an accuracy of 99.85%. Out of the 200 packets tested, 199 packets were predicted accurately.



Checking for the accuracy of the model:

```
In [6]: from sklearn.model_selection import cross_val_score

        print(cross_val_score(model, x, y, cv = 10, scoring = 'accuracy').mean())

        0.99855
```

Figure 22: Accuracy of the Decision Tree Algorithm for Classifying Tor vs Surface-Net Traffic

Top Rules Obtained:

1. When ‘Source’ <= 81364160, ‘Destination Port’ <= 48769 and Source Port <=56890, 2056 out of 2064 connections are to the ‘Surface-net’. This rule gives us an accuracy of 99.61% and applies to roughly 10 % of the dataset
2. When ‘Source’ > 995696512 and ‘Destination Port’ <= 58884.5, 5796 out of 5800 connections are to the ‘Tor network’. This rule gives us an accuracy of 99.93% and applies to roughly 29 % of the dataset
3. When 1672304640 < ‘Source’ < 1761521920 and ‘Source Port’ <= 108.5, 4779 out of 4784 connections are to the ‘Surface-net’. This rule gives us an accuracy of 99.89 and applies to roughly 24 % of the dataset

Decision Tree Without Using IP Addresses:

Since IP Addresses can change from node to node, it is also important to test the condition where IP addresses do no matter. Using the same dataset used above, we created another dataset without the ‘Source IP Addresses’ and the ‘Destination IP Addresses’. Doing so will give us an idea of how well a Machine Learning algorithm can classify generic information from any node.

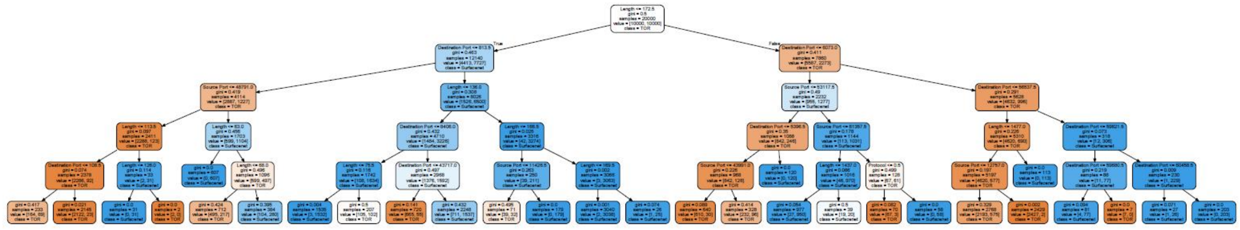


Figure 23: Decision Tree for Classifying Tor vs Surface-Net Traffic (Without IP Addresses)

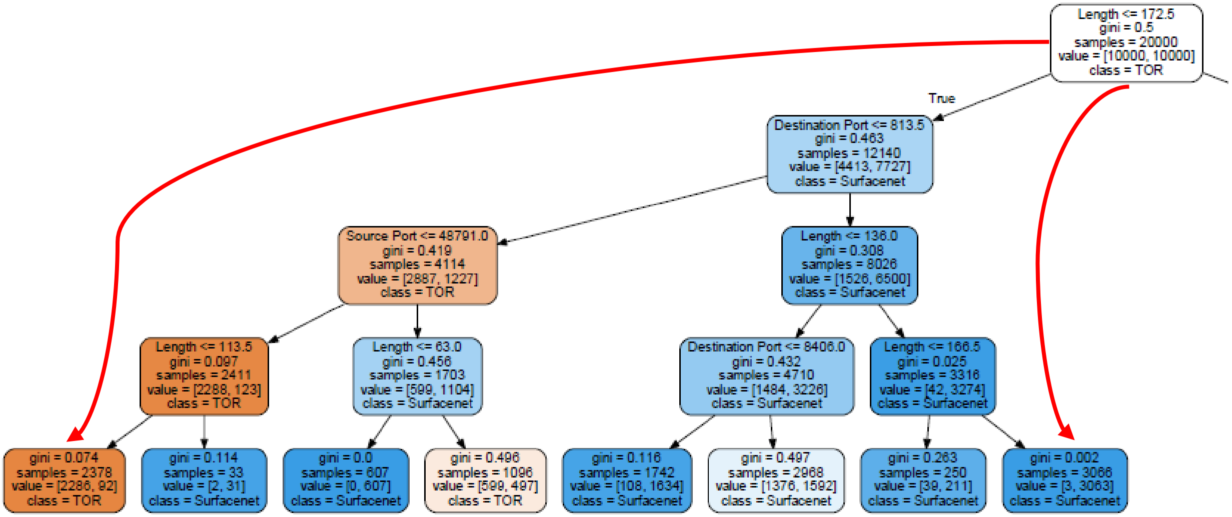
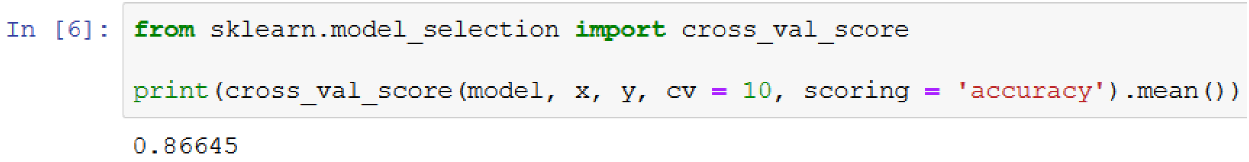


Figure 24: Important Rules for Classifying Tor vs Surface-Net Traffic (Without IP Addresses)

The Decision Tree generated is wider than the one with IP Addresses. This gives us an indication that the tree would have fewer significant rules than before. On careful observing, we only find two significant rules on the left side of the tree. Also, this tree has an accuracy of 86 % which is acceptable but about 14 % less than before.



```
In [6]: from sklearn.model_selection import cross_val_score

        print(cross_val_score(model, x, y, cv = 10, scoring = 'accuracy').mean())

        0.86645
```

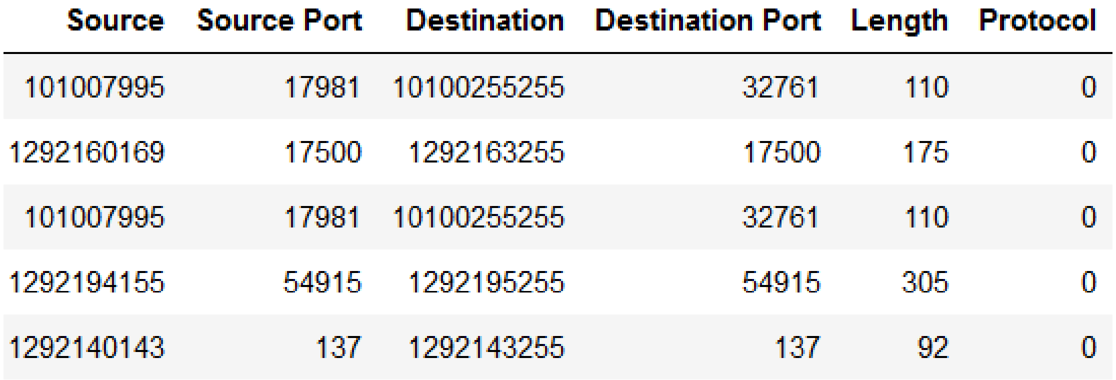
Figure 25: Accuracy of the Decision Tree Algorithm for Classifying Tor vs Surface-Net Traffic (Without IP Addresses)

Top Rules (Without Using IP Addresses):

1. When ‘Length’ <= 113.5 and ‘Destination Port’ <= 813.5 and ‘Source Port’ <= 48791, 2286 out of 2378 connections are to the ‘Tor network’. This rule gives us an accuracy of 96 % and applies to roughly 12 % of the dataset
2. When ‘Length’ > 166.5 and ‘Destination Port’ > 813.5, 3063 out of 3066 connections are to the ‘Surface-net’. This rule gives us an accuracy of 99.9 % and applies to roughly 15% of the dataset

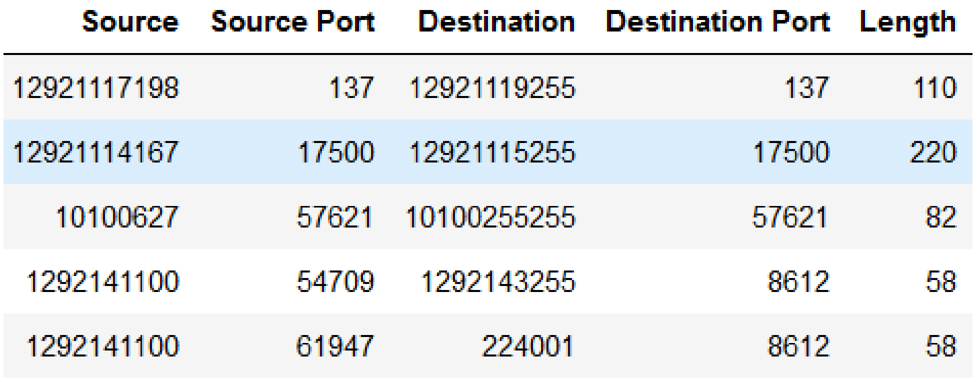
Tor TCP vs UDP Traffic:

The same approach could be used to check if a machine learning algorithm can identify TCP and UDP traffic through the Tor node. Most video streaming services use UDP and creating a rule that can identify such traffic along with a combination of other rules would be useful in accurately identifying traffic. Though only knowing if certain traffic is TCP or UDP is not of much use, combining the information with other information obtained using machine learning can be advantageous.



Source	Source Port	Destination	Destination Port	Length	Protocol
101007995	17981	10100255255	32761	110	0
1292160169	17500	1292163255	17500	175	0
101007995	17981	10100255255	32761	110	0
1292194155	54915	1292195255	54915	305	0
1292140143	137	1292143255	137	92	0

Figure 26: Training Set for Classifying TCP vs UDP Traffic



Source	Source Port	Destination	Destination Port	Length
12921117198	137	12921119255	137	110
12921114167	17500	12921115255	17500	220
10100627	57621	10100255255	57621	82
1292141100	54709	1292143255	8612	58
1292141100	61947	224001	8612	58

Figure 27: Test Set for Classifying TCP vs UDP Traffic

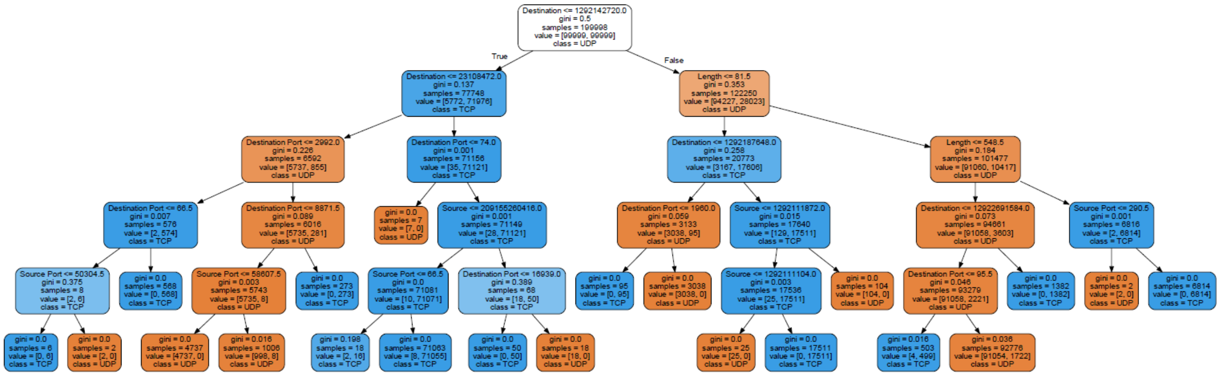

Destination <= 1292142720.0 gini = 0.5 samples = 199998 value = [99999, 99999] class = UDP
True
False
Destination <= 23108472.0 gini = 0.137 samples = 77748 value = [5772, 71976] class = TCP
Length <= 81.5 gini = 0.353 samples = 122250 value = [94227, 28023] class = UDP
Destination Port <= 2992.0 gini = 0.226 samples = 6592 value = [5737, 855] class = UDP
Destination Port <= 74.0 gini = 0.001 samples = 71156 value = [35, 71121] class = TCP
Destination <= 1292187648.0 gini = 0.258 samples = 20773 value = [3167, 17606] class = TCP
Length <= 548.5 gini = 0.184 samples = 101477 value = [91060, 10417] class = UDP
Destination Port <= 66.5 gini = 0.007 samples = 576 value = [2, 574] class = TCP
Destination Port <= 8871.5 gini = 0.089 samples = 6016 value = [5735, 281] class = UDP
gini = 0.0 samples = 7 value = [7, 0] class = UDP
Source <= 209155260416.0 gini = 0.001 samples = 71149 value = [28, 71121] class = TCP
Destination Port <= 1960.0 gini = 0.059 samples = 3133 value = [3038, 95] class = UDP
Source <= 1292111872.0 gini = 0.015 samples = 17640 value = [129, 17511] class = TCP
Destination <= 12922691584.0 gini = 0.073 samples = 94661 value = [91058, 3603] class = UDP
Source Port <= 290.5 gini = 0.001 samples = 6816 value = [2, 6814] class = TCP
Source Port <= 50304.5 gini = 0.375 samples = 8 value = [2, 6] class = TCP
gini = 0.0 samples = 568 value = [0, 568] class = TCP
Source Port <= 58607.5 gini = 0.003 samples = 5743 value = [5735, 8] class = UDP
gini = 0.0 samples = 273 value = [0, 273] class = TCP
Source Port <= 66.5 gini = 0.0 samples = 71081 value = [10, 71071] class = TCP
Destination Port <= 16939.0 gini = 0.389 samples = 68 value = [18, 50] class = TCP
gini = 0.0 samples = 95 value = [0, 95] class = TCP
gini = 0.0 samples = 3038 value = [3038, 0] class = UDP
Source <= 1292111104.0 gini = 0.003 samples = 17536 value = [25, 17511] class = TCP
gini = 0.0 samples = 104 value = [104, 0] class = UDP
Destination Port <= 95.5 gini = 0.046 samples = 93279 value = [91058, 2221] class = UDP
gini = 0.0 samples = 1382 value = [0, 1382] class = TCP
gini = 0.0 samples = 2 value = [2, 0] class = UDP
gini = 0.0 samples = 6814 value = [0, 6814] class = TCP
gini = 0.0 samples = 6 value = [0, 6] class = TCP
gini = 0.0 samples = 2 value = [2, 0] class = UDP
gini = 0.0 samples = 4737 value = [4737, 0] class = UDP
gini = 0.016 samples = 1006 value = [998, 8] class = UDP
gini = 0.198 samples = 18 value = [2, 16] class = TCP
gini = 0.0 samples = 71063 value = [8, 71055] class = TCP
gini = 0.0 samples = 50 value = [0, 50] class = TCP
gini = 0.0 samples = 18 value = [18, 0] class = UDP
gini = 0.0 samples = 25 value = [25, 0] class = UDP
gini = 0.0 samples = 17511 value = [0, 17511] class = TCP
gini = 0.016 samples = 503 value = [4, 499] class = TCP
gini = 0.036 samples = 92776 value = [91054, 1722] class = UDP

Figure 28: Decision Tree for Classifying TCP vs UDP Traffic

As with the previous experiment, we can obtain important rules in as low as the third level of the tree. What can be observed with both these experiments is that the lower the depth at which a rule occurs, the more packets it applies to. Thus, if a rule occurs at depth of two would apply to more packets than a rule at occurs at depth five. Along with creating a Decision Tree, it is also important to evaluate rules that can be significant. Another important part of the process of evaluating the rules by depth is to understand the effect of a rule. A rule might be able to narrow down the result, but it is also important to consider how much the rule narrows down the result. Thus, the tree should be carefully evaluated to only extract the most important and significant rules, thus optimizing the performance of the algorithm.

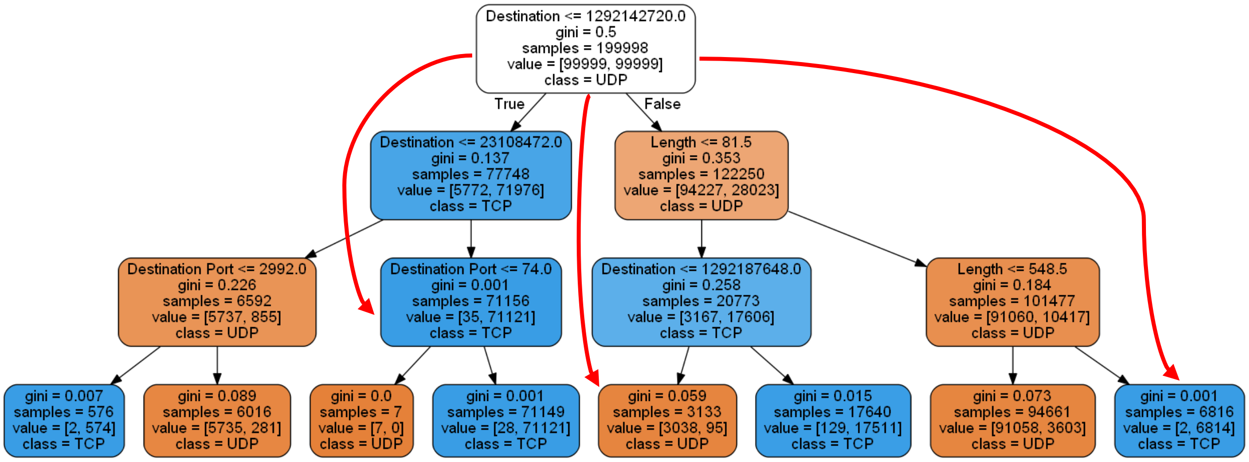
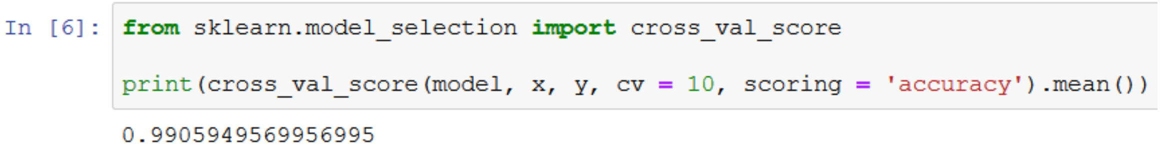


Figure 29: Important Rules for Classifying TCP vs UDP Traffic

Our machine learning algorithm was able to attain an accuracy of 99.05%. Out of the total of 2000 packets tested, 1997 packets were correctly predicted.


```
In [6]: from sklearn.model_selection import cross_val_score

        print(cross_val_score(model, x, y, cv = 10, scoring = 'accuracy').mean())

        0.9905949569956995
```

Figure 30: Accuracy of the Algorithm for Classifying TCP vs UDP Traffic

Top Rules:

3. When ‘Source’ <= 1292142720 and ‘Destination’ > 23108472, 71121 out of 71156 connections use ‘TCP’. This rule gives us an accuracy of 99.9% and applies to roughly 36% of the dataset
4. When ‘Source’ > 1292142720, ‘Length’ <= 81.5 and ‘Destination’ <= 1292187648, 3038 out of 3133 connections use ‘UDP’. This rule gives us an accuracy of 96.96% and applies to roughly 1.5% of the dataset
5. When ‘Source’ > 1292142720 and ‘Length’ > 548.5, 6814 out of 6816 connections use ‘TCP’. This rule gives us an accuracy of 99.97 and applies to roughly 3.5% of the dataset

Decision Tree Without Using IP Addresses:

The following Decision Tree was realized after eliminating the IP addresses from the dataset. As discussed earlier, IP addresses can change from node to node.

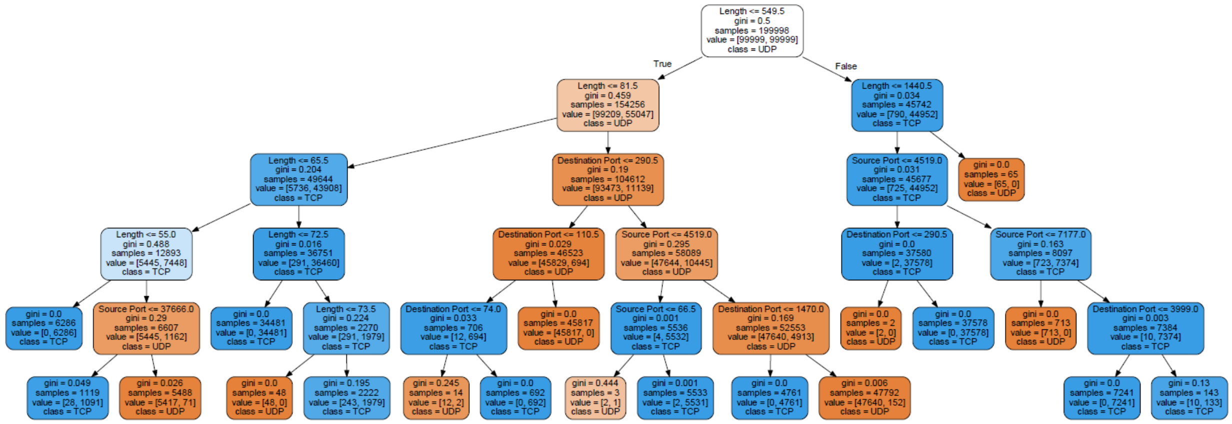


Figure 31: Decision Tree for Classifying TCP vs UDP Traffic (Without IP Addresses)

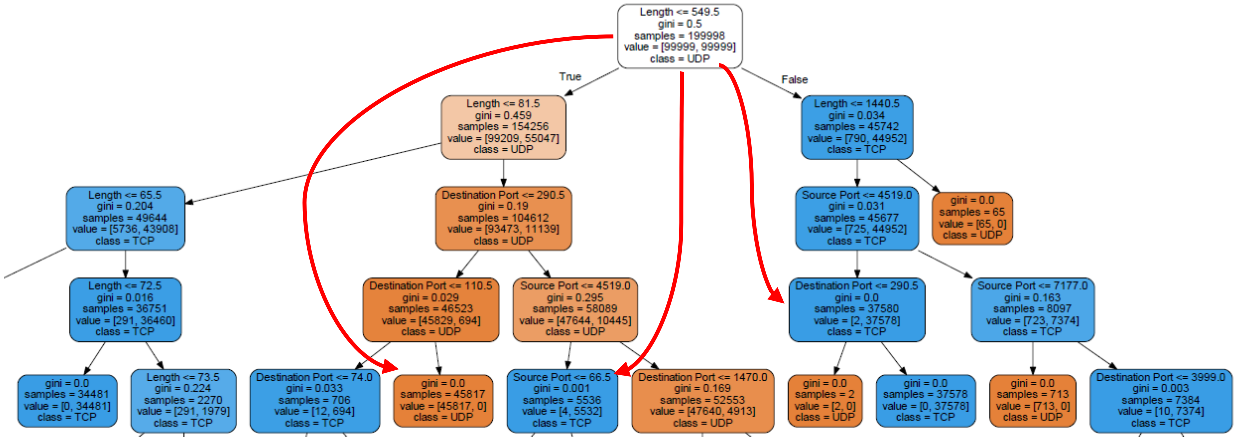
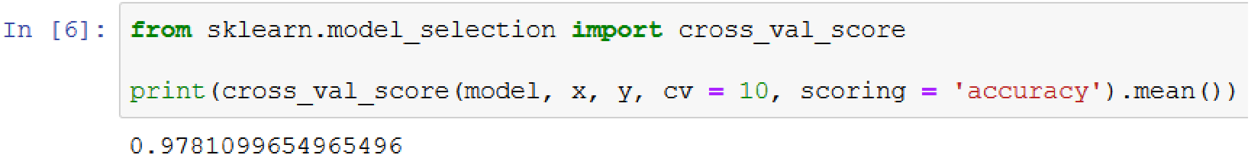


Figure 32: Important Rules for Classifying TCP vs UDP Traffic (Without IP Addresses)

We also obtain an accuracy of 97 %, which is just 2% less than before. But, since there are no IP addresses involved, it is a more generic rule than before.



```
In [6]: from sklearn.model_selection import cross_val_score

        print(cross_val_score(model, x, y, cv = 10, scoring = 'accuracy').mean())

        0.9781099654965496
```

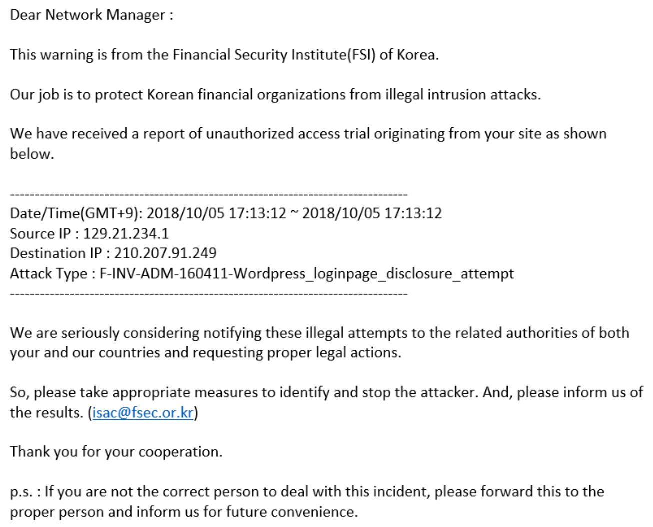
Figure 33: Accuracy of the Algorithm for Classifying TCP vs UDP Traffic (Without IP Addresses)

Top Rules (Without using IP Addresses):

1. When 81.5 < ‘Length’ < 549.5 and ‘Destination Port’ <= 110.5, 45817 out of 45817 connections use ‘UDP’. This rule gives us an accuracy of 100 % and applies to roughly 23% of the dataset
2. When 81.5 < ‘Length’ < 549.5, ‘Destination Port’ > 290.5 and ‘Source Port’ < 4519, 5532 out of 5536 connections use ‘UDP’. This rule gives us an accuracy of 99.9 % and applies to roughly 2 % of the dataset
3. When 549.5 < ‘Length’ < 1440.5, ‘Destination Port’ <= 290.5 and ‘Source Port’ < 4519, 37578 out of 37580 connections use ‘UDP’. This rule gives us an accuracy of 99.9 % and applies to roughly 18 % of the dataset

Warnings Received While Hosting Tor Exit Node:

1. Numerous complaints from REN-ISAC (Research and Education Network, Information Sharing and Coordination) which comprises mainly of about 2000 universities, some not-for-profit, and public sector research labs
2. Reports of scanning/hacking attempts on the Financial Security Institute (FSI) of Korea:



Dear Network Manager :

This warning is from the Financial Security Institute(FSI) of Korea.

Our job is to protect Korean financial organizations from illegal intrusion attacks.

We have received a report of unauthorized access trial originating from your site as shown below.

Date/Time(GMT+9): 2018/10/05 17:13:12 ~ 2018/10/05 17:13:12
Source IP : 129.21.234.1
Destination IP : 210.207.91.249
Attack Type : F-INV-ADM-160411-Wordpress_loginpage_disclosure_attempt
----

We are seriously considering notifying these illegal attempts to the related authorities of both your and our countries and requesting proper legal actions.

So, please take appropriate measures to identify and stop the attacker. And, please inform us of the results. (isac@fsec.or.kr)

Thank you for your cooperation.

p.s. : If you are not the correct person to deal with this incident, please forward this to the proper person and inform us for future convenience.

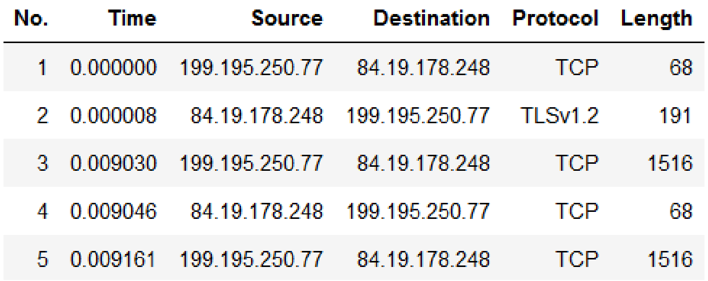
Figure 34: E-mail from the Financial Institute of Korea

3. Complaints of ‘Phone-Home’ operations of malware on other computers using the hosted Tor Exit Node to route their traffic
4. Fourteen notifications of distinct malware phone-home activity in a single day
5. Forty-eight notifications of malware phone-home activity in a week
6. About forty-three Digital Millennium Copyright Act (DMCA) violations in a week
7. Other scanning/hacking attempt reports from multiple sources including at least one university
8. Around one-third of all the reports over a span of one and a half week involved the Exit Node

Resolving the Location from IP addresses on Tor

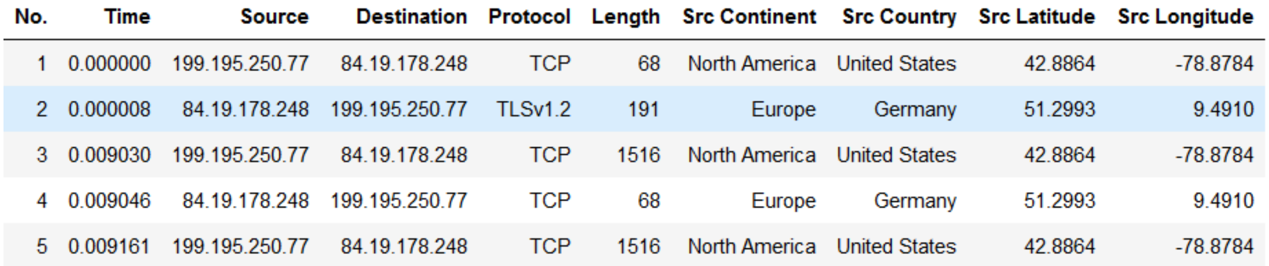
Knowing where the traffic on a Tor node is coming from is always an advantage. Tor nodes are regularly under attacks from hackers trying to convert them into a botnet. These hackers often try using multiple machines to carry out the attack. Attacks can often originate from a certain location in the world. In such a case, knowing the location of the origin of certain packets can be useful in blocking attacks. If an attack is observed to be coming from a certain country, state or city, it can be efficiently blocked while maintaining service to other parts of the world.

One way of achieving this is by using the GeoIP2 database and comparing the incoming traffic to it. The traffic used below was captured on a Tor node using Wireshark and then imported into a Python notebook for processing. The source IP addresses were compared to the GeoIP2 database to find the ‘Source Continent’, ‘Source Country’, ‘Source Latitude’ and ‘Source Longitude’. These obtained attributes can also be then used to train a machine learning algorithm for a specific application.



No.	Time	Source	Destination	Protocol	Length
1	0.000000	199.195.250.77	84.19.178.248	TCP	68
2	0.000008	84.19.178.248	199.195.250.77	TLSv1.2	191
3	0.009030	199.195.250.77	84.19.178.248	TCP	1516
4	0.009046	84.19.178.248	199.195.250.77	TCP	68
5	0.009161	199.195.250.77	84.19.178.248	TCP	1516

Figure 35: Captured Packets Used for Identifying Location



No.	Time	Source	Destination	Protocol	Length	Src Continent	Src Country	Src Latitude	Src Longitude
1	0.000000	199.195.250.77	84.19.178.248	TCP	68	North America	United States	42.8864	-78.8784
2	0.000008	84.19.178.248	199.195.250.77	TLSv1.2	191	Europe	Germany	51.2993	9.4910
3	0.009030	199.195.250.77	84.19.178.248	TCP	1516	North America	United States	42.8864	-78.8784
4	0.009046	84.19.178.248	199.195.250.77	TCP	68	Europe	Germany	51.2993	9.4910
5	0.009161	199.195.250.77	84.19.178.248	TCP	1516	North America	United States	42.8864	-78.8784

Figure 36: Location Details of the Source IP Addresses

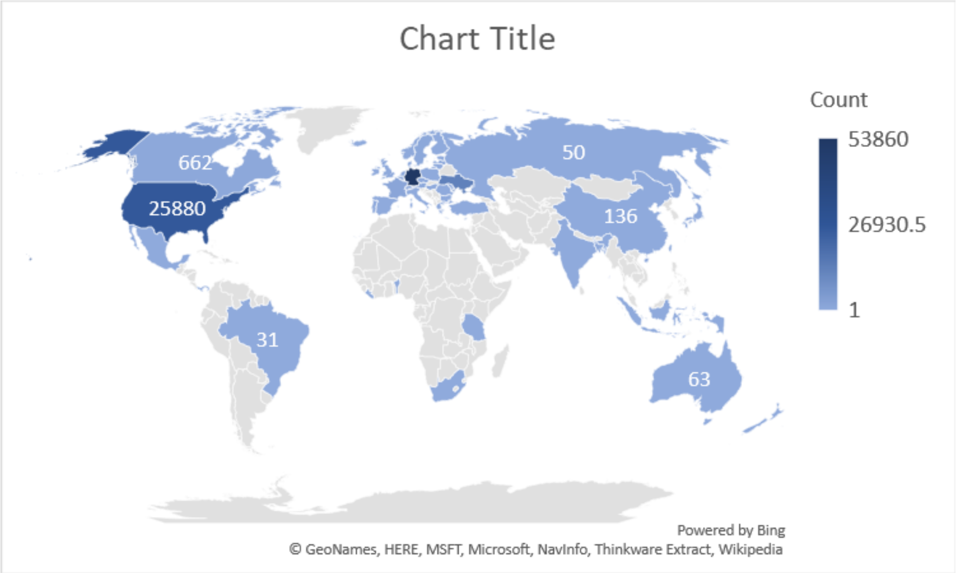
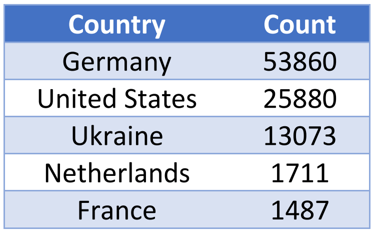


Chart Title
Count
53860
26930.5
1
662
25880
50
136
31
63
Powered by Bing
© GeoNames, HERE, MSFT, Microsoft, NavInfo, Thinkware Extract, Wikipedia

Figure 37: World Map Representing the Countries with the Highest Number of Connections

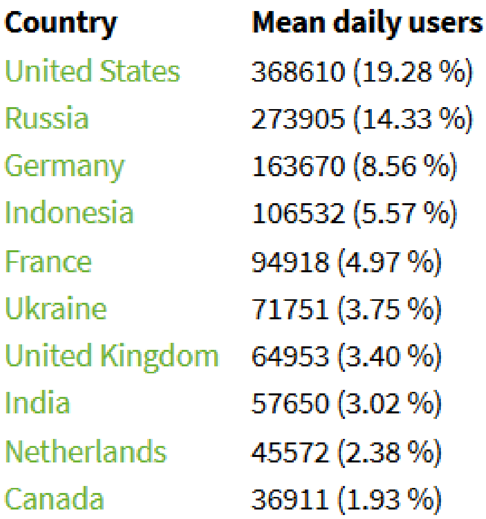
The figure above shows a world map with the countries with the highest number of connections to the node. The highest number of connections to the node is from Germany followed by the United States of America. The number of connections from Germany does not show up on the above map because of the text being larger than the size of the country represented on the map. Below is the list of the top five countries with the highest connections to the node.



Country	Count
Germany	53860
United States	25880
Ukraine	13073
Netherlands	1711
France	1487

Figure 38: Top Five Countries with the Highest Number of Connections to the Node

The above can be compared with the list of countries with the highest number of Tor users published on the Tor Metrics website. We can see that all the countries in our list are also on the list of the top-ten countries published by Tor Metrics.



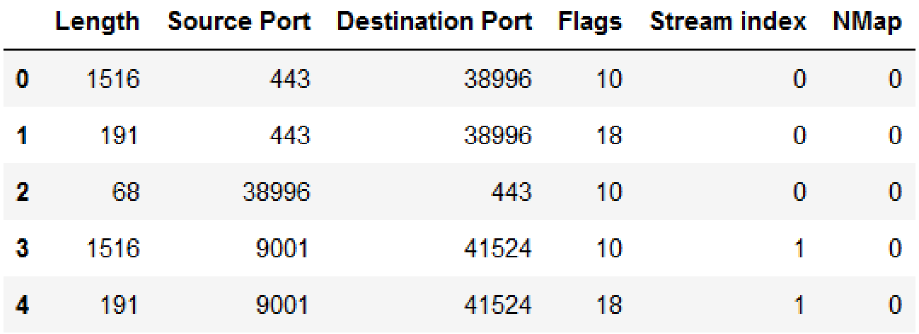
Country	Mean daily users
United States	368610 (19.28 %)
Russia	273905 (14.33 %)
Germany	163670 (8.56 %)
Indonesia	106532 (5.57 %)
France	94918 (4.97 %)
Ukraine	71751 (3.75 %)
United Kingdom	64953 (3.40 %)
India	57650 (3.02 %)
Netherlands	45572 (2.38 %)
Canada	36911 (1.93 %)

Figure 39: List of Countries with Highest Tor Users According to Tor Metrics

Identifying Port-scan attacks on Tor node:

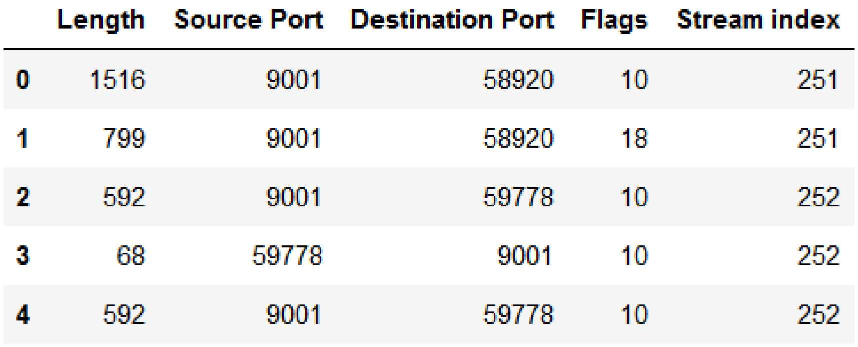
Port scan attacks are generally the first steps in any attack strategy. A port scan attempt on a system should never be ignored and always be taken seriously. Generally, the IP addresses of the machines from which the port scans are attempted can be blacklisted for security. To avoid this, modern attacks methodologies initiate port scan attacks over the Tor network. This also allows the attacker to use multiple Tor connections to carry out the attack. The experiment was focused on identifying a Nmap scan originating from the Tor network and comparing it to one originating from the internet. Machine learning was used in this case to identify and differentiate between the two. A Nmap attack originating from the Tor network would require special attention since in most cases, it is an attempt to take control over the node and turn it into a ‘zombie node’ or use it in the form of a ‘botnet’.

For this experiment, we eliminated the ‘Source IP Address’ and ‘Destination IP Address’ attributes to since the attack can potentially originate from any IP address, but would be intended towards our controlled node. The used attributes were ‘Packet Length’, ‘Source Port’, ‘Destination Port’, ‘Flags’ and ‘Stream Index’.



	Length	Source Port	Destination Port	Flags	Stream index	NMap
0	1516	443	38996	10	0	0
1	191	443	38996	18	0	0
2	68	38996	443	10	0	0
3	1516	9001	41524	10	1	0
4	191	9001	41524	18	1	0

Figure 40: Training Set for Classifying Regular Port-Scan vs Port-Scan from a Tor Node



	Length	Source Port	Destination Port	Flags	Stream index
0	1516	9001	58920	10	251
1	799	9001	58920	18	251
2	592	9001	59778	10	252
3	68	59778	9001	10	252
4	592	9001	59778	10	252

Figure 41: Test Set for Classifying Regular Port-Scan vs Port-Scan from a Tor Node

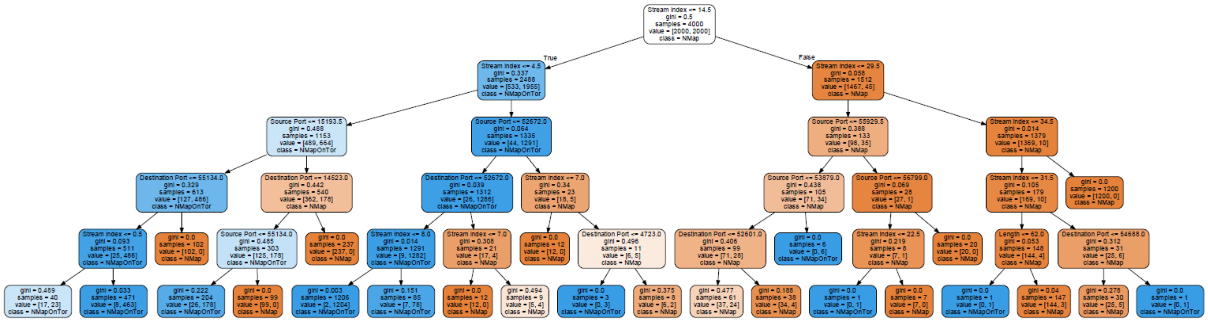


Figure 42: Decision Tree for Classifying Regular Port-Scan vs Port-Scan from a Tor Node

The above figure is the Decision Tree generated with a depth of five. As we can see, the tree spreads wide, but the important rules are close to the root. The figure below shows two important rules occurring at a depth of two and three. We can also clearly see that the decision tree is quite literally dividing traffic into two parts with more rules for 'NmapOnTor' on the left and regular 'Nmap' on the right.

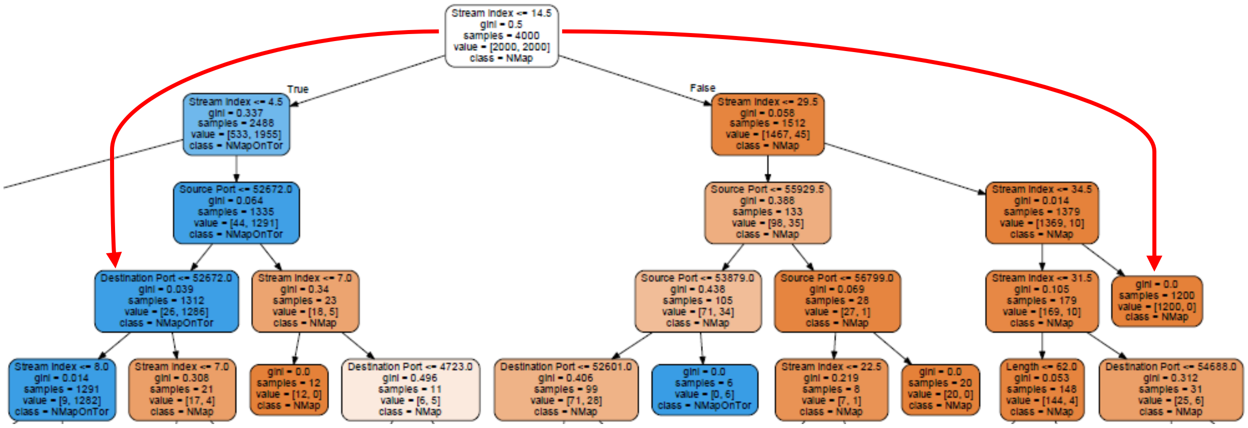
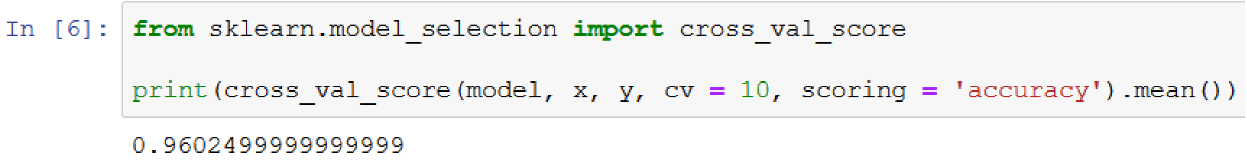


Figure 43: Important Rules for Classifying Regular Port-Scan vs Port-Scan from a Tor Node



```
In [6]: from sklearn.model_selection import cross_val_score

        print(cross_val_score(model, x, y, cv = 10, scoring = 'accuracy').mean())

        0.9602499999999999
```

Figure 44: Accuracy of the Algorithm for Classifying Regular Port-Scan vs Port-Scan from a Tor Node

TOR: Internet and Tor Traffic Classification Using Machine Learning

Top Rules:

1. When ‘Stream Index’ <= 4.5 and ‘Source Port’ <= 52672, 1286 out of 1312 connections are NMap scans from a ‘Tor node’. This rule gives us an accuracy of 98 % and applies to roughly 33 % of the dataset
2. When ‘Stream Index’ > 34.5, 1200 out of 1200 connections are NMap scan from a machine on the ‘Surface-net’. This rule gives us an accuracy of 100 % and applies to roughly 30 % of the dataset

Conclusion

Identifying traffic over the Internet has been part of network security practices for a long time. Tor, which focuses on a client’s privacy is tricky to deal with. Identification of the type of traffic is important though if we want to maintain the security of our systems. Tor is now not only used as a tool of privacy but also to successfully carry out complicated attacks while maintaining anonymity. Another requirement of knowing the type of traffic is applicable to individuals that want to support the Tor Project and hosting ‘Exit Nodes’ on the network. As can be seen from our experiments, a Tor exit node is highly susceptible to getting DMCA notices, unknowingly getting involved in a hacking attempt, or warnings for other illicit traffic being relayed through their node. Though Tor is completely legal, and the individual is not liable for the traffic relayed through the node, it might be difficult to prove that the traffic was only relayed through the node and did not originate there.

A solution was proposed where multiple Tor nodes were hosted and the traffic flowing through them was compared to regular internet activity using Machine Learning. The aim of the experiments conducted was to check if Machine Learning can identify and differentiate between activities over the internet to that on the Tor network. We found that it was possible to implement Decision Trees to classify different types of traffic efficiently. In all our experiment, we were successfully able to classify traffic with an accuracy of more than ninety-five percent. Though these rules can be implemented individually, they become more powerful when combined with each other. An algorithm with all the rules that we generated can be used to identify four parameters of a packet at the same time. Thus, we can not only identify certain packets to be originating from the Tor network but also know that it is part of a TCP, which is an ‘NMAP’ scan while also knowing the location of the hosts they are originating from. Combining rules in such a way would allow us to customize our filter to segregate traffic efficiently. Firewalls can be created using these rules to block certain type of Tor traffic while allowing uninterrupted service to other users. All this can be done without sacrificing the privacy of users over the network. More importantly, we concluded that Machine Learning can be an efficient tool for security over the Tor network and more such stackable algorithms can be created to make Tor more flexible.

Annotated Bibliography

Abbott, T. G., Lai, K. J., Lieberman, M. R., & Price, E. C. (2007). Browser-Based Attacks on Tor. *International Workshop on Privacy Enhancing Technologies*, 184-199.

This paper explains the traditional ways to attack the TOR network and discusses the various loopholes that can be used to compromise identity over the Dark Net

Bell, J. (2014). *Machine Learning: Hands-On for Developers and Technical Professionals.* Wiley.

This book is one of the best resources available to start with machine learning. It starts with a brief introduction to machine learning and then goes into deeper concepts right from data collection to complex algorithm.

Biryukov, A., Pustogarov, I., & Weinmann, R.-P. (2013). Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. *Security and Privacy (SP), 2013 IEEE Symposium on.* IEEE. Retrieved 9 14, 2017, from http://www.tor.com/blogs/2013/03/tor-books-announces-new-dragon-age-novel-with-bioware-senior-writer-patrick-weeks

This paper was instrumental in understanding how the TOR network works from the service point of view. It explains how the services over the TOR network process information and gives an overview of the various mechanisms working underneath the architecture.

Bollier, D. (2010). *The Promise and Peril of Big Data.* The Aspen Institute.

Chang, C.-H., Kayed, M., Girgis, M., & Shaalan, K. (2006). A Survey of Web Information Extraction Systems. *IEEE Transactions on Knowledge and Data Engineering, 18*(10), 1041-4347.

A short paper that gives an overview of data mining over the internet.

Davenport, T. H., Barth, P., & Bean, R. (2012). How Big Data is Different. *MIT Sloan Management Review*.

The paper was a brief introduction to the basics of Big Data, the various Big Data tools, and procedures. It also explains how Big Data is different than other data collection methods.

Dolliver, D. S. (2015). Evaluating drug trafficking on the TOR Network: Silk Road 2, the sequel. *International Journal of Drug Policy, 26*(11), 1113-1123.

The paper discusses how Silk Road 2 was busted by the FBI. Details the mistakes that were made by the owners of the website and the ways in which the law enforcement agencies were able to crack them down.

FBI. (2016, 11 1). *A Primer on DarkNet Marketplaces: What They are and What Law Enforcement is Doing to Combat Them*. Retrieved from FBI: https://www.fbi.gov/news/stories/a-primer-on-darknet-marketplaces

This article on the FBI's website explains the basics of the operation of TOR and then explains the judicial standpoint related to the illicit activities.

Kang, L. (2015). Efficient botnet herding within the TOR network. *Journal of Computer Virology and Hacking Techniques*, 19-26.

This is a great resource for understanding how clusters can be formed by infecting devices using botnets and then using their resources for various applications. Though generally implemented for the wrong causes, it can be a great tool for collective computations.

McCoy, D., Bauer, K., Grunwald, D., Kohno, T., & Sicker, D. (2008). Shining Light in Dark Places: Understanding the Tor Network. *International Symposium on Privacy Enhancing Technologies Symposium* (pp. 63-76). Springer.

One of the best resources to start understanding the fundamentals of TOR and other anonymous networks. This paper talks about the architecture and working of the various components in the network.

Wagner, C., Wagener, G., State, R., Dulaunoy, A., & Engel, T. (2012). Breaking Tor anonymity with game theory and data mining. *Concurrency and Computation*, 1052–1065.

This paper explains about research conducted over breaking the TOR network using a game theory which is one of the more nontraditional ways of attack. It helped in developing a new approach to understanding things differently