Characterization and Classification of Malware Traffic over the Tor Network

Marie Betel B. de Robles, Joseph Anthony C. Hermocilla, and Jaderick P. Pabico
Institute of Computer Science, College of Arts and Sciences
University of the Philippines Los Baños, College, Laguna
{mbderobles2,jchermocilla,jppabico}@up.edu.ph

ABSTRACT

Tor is a popular anonymity tool for digital-based communication that is used to protect the users' identity and is also utilized to avoid any eavesdropping and man-in-the-middle attacks. It implements the concept of onion routing where traffic is being routed to relay nodes which hide users' identity and secure the data being transferred over the internet. As its use increases among internet users, there is a need to monitor the traffic within Tor to ensure that it is not being misused. There are a number of ways Tor is used for malicious purposes, one of which is to hide malware traffic. Since Tor traffic is encrypted, traditional approaches such as port examination and packet inspection are ineffective. In this study, the automatic classification of Tor web traffic into malware and non-malware types was conducted using machine learning approaches. A dataset of positive and negative malware web traffic examples was generated from VirusTotal's malware applications and regular web traffic. The dataset was generated from a controlled network environment using an automated system that was developed in this work. Various machine learning (ML) techniques were employed to evaluate their respective effectiveness in classifying between malware and non-malware Tor traffic. The ML considered were C4.5 variant of decision tree, K-nearest neighbors, Naive Bayes, and Random Forest. The ML respective effectiveness in classification were evaluated against the classification metrics Precision, Recall, False Positive Rate, and False Negative Rate using a standard statistical analysis of variance (ANOV) procedure. ANOV results show that the MLs respective performances in evaluating the metrics are not statistically different from each other at a significance of $\alpha = 0.05$.

CCS CONCEPTS

• **Security and privacy → Network security**;

KEYWORDS

Malware, Tor, Flows, Machine Learning

ACM Reference Format:
Marie Betel B. de Robles, Joseph Anthony C. Hermocilla, and Jaderick P. Pabico. 2020. Characterization and Classification of Malware Traffic over the Tor Network. In *Proceedings of Philippine Computing Science Congress (PCSC2020).* Baguio City, Philippines, 10 pages.

1 INTRODUCTION

The internet has been continuously evolving and expanding its reach to people's everyday lives. Today, users can access new ideas and information and perform activities because innovations in communication, information sharing, education, and other related technologies can be easily utilized. As the use of internet is increasing, anonymity while on it became one of the most appealing services for some users. Encrypting information about users' activities allows users to exercise freedom of speech, avoid network surveillance, and seek discomfiting information. There are many ways for users to obtain anonymity on the internet, one of which is through Tor.

Tor is a popular anonymous proxy tool that is being used by millions of users worldwide. It protects users' data using the onion routing concept where data is wrapped in multiple layers of encryption. As it was originally developed for protecting government communications, it is now being used by a wide range of users such as journalists, activists, business executives, and others who prefer to protect their identities while exchanging data and communicating over the internet. It also hosts hidden services accessible only through the Tor network itself [3].

Users with malicious intent have also become attracted to anonymity tools like Tor. Today, numerous unlawful services and activities are being conducted over an encrypted network. Even malware has also utilized encryption for its communication [8]. In the study by Anderson et al. [6], about 10% of their collected malware communication samples were encrypted. For example, the malware Locky is a ransomware that prompts the victim to visit a Tor hidden service to pay the ransom and retrieve stolen files [7]. Since Tor can be used by anyone, attackers can easily hide any malware tracks through it. This threat makes it important to monitor Tor traffic to determine if its use is for malicious purposes.

Traditional traffic classification approaches such as Deep Packet Inspection (DPI) and signature-matching are ineffective over encrypted traffic. Today, machine learning (ML) techniques are being applied where the features used are derived from statistical characteristics of internet traffic [23]. Certain classes of applications hold these unique features and are used in ML algorithms for traffic classification. This study aims to classify Tor web traffic into malware and non-malware types and to develop a computational workflow that can classify malware traffic over the Tor network through ML approaches.

2 RELATED WORK

Several works have already been done to analyze Tor traffic but only a few studies in malware detection.

Ling et al. [20] developed a system which integrates an intrusion detection system at Tor exit routers to detect malicious Tor traffic. They were able to monitor traffic and deploy the system in a university. Results of their work showed that 10% of the traffic is malicious, where the majority of it is P2P traffic and almost 9% is malware traffic. They further improved their system in their succeeding work [21] where they proposed a way to block malicious Tor traffic. They configured the intrusion detection system to send alerts to a sentinel agent which will tear down the specific connection from the exit router. Results of their work showed that 8.99% of the received alerts were malware-related.

There are studies which only focused on the malware-specific detection such as worms [10, 12] and botnets [5, 15]. Al-Bataineh and White [5] focused on detecting web-based data-stealing botnets. They conducted an experiment using Naive Bayes, Multi-layer Perception (MLP), and J48 [1] classifiers where entropy and byte frequency distribution from HTTP POST requests were used as features. As J48 yielded the best results, it was fine-tuned where results produced no false positives and minimum false negatives.

Other works focused on correlating malware families with their usage of different protocols. Bekerman et al. [9] focused on features from commonly-used application layer protocols (i.e. DNS, HTTP and SSL) which yielded better performance over other intrusion detection systems. Anderson et al. [6], on the other hand, correlated the detection of malware family through its Transport Layer Security (TLS) usage. They observed the differences on malware families through ciphersuite selection, TLS client parameters, TLS extensions, and many others. In this study, however, features from TLS usage is not possible since Tor traffic implements TLS.

The closest work related to this study is from Jiménez and Goseva-Popstojanova [18] where they used flows-based features and system logs for malware detection. They used ML approaches such as J48, Naive Bayes, PART, and Random Forest and evaluated their tests using accuracy, precision, recall, 1-FPR, F-score, and G-score. In this study, only flow-based features were considered since it is not possible to monitor user's computer state. A flow is a series of packets with the same 5-tuple values (i.e., source IP, destination IP, source port, destination port, protocol). The relationship between traffic classes and statistical properties like flow-based features has been discussed in previous studies where the researchers analyzed and constructed empirical models of connection characteristics [24]. This study tests if these features are also effective in classifying Tor malware traffic.

3 ONION ROUTING

Onion routing was first published in 1996. It was mainly designed to provide anonymity to users. As a result of its use, eavesdropping is avoided while traffic analysis within the network is impossible to conduct. Onion routing establishes an anonymous connection by creating a path through the network where each node can only identify adjacent nodes along the path. Over time, the first generation onion routing was improved but was abandoned in 2002. In 2004, Tor, a second generation onion router was introduced. It improved and addressed some design and deployment issues of the first generation onion routing design [3, 11].

[1]Java implementation of C4.5

Figure 1 shows a simple illustration of how a client connects to the internet over the Tor network. For instance, if a user wants to access a web service in the internet through the Tor network, a local software in the client's machine called an onion proxy (OP) will establish a path between the client and the web service. The OP will choose three random onion routers (ORs), respectively called as entry, middle, and exit routers, which will constitute as a Tor circuit. Traffic between the client and the web service is routed through the created Tor circuit. Data from the client is bundled with layers of encryption before it passes the Tor circuit. For every OR the data passes, a layer of encryption is removed to determine the next destination of the data. When the data reaches the exit router, the last layer of encryption will be removed and will be finally sent to the web service [3].

Several issues from the previous onion router design were addressed by the second generation onion router. One is the perfect forward secrecy where ORs are prevented from recording the traffic passing through them. This is done through the use of the TLS connections with ephemeral keys for ORs to communicate with each other. Tor also uses SOCKS proxy interface where they can support more TCP-based applications. Tor also maintains directory servers where trusted relay nodes and their state are listed and can be publicly downloaded [11].

4 METHODOLOGY

4.1 Dataset Generation

The dataset consists of malware and non-malware Tor web traffic, labelled as TorMal2019. To generate malware Tor web traffic, malware samples need to be executed over a controlled environment. Recent samples (October 2017 to May 2019) of different malware types such as trojans, worms, ransomware, and viruses were collected from VirusTotal [1]. These samples were selected based on their network communication capabilities whose information is already provided by the VirusTotal website.

The non-malware traffic was captured by browsing HTTP and HTTPS websites via web browsers Chrome and Mozilla. The malware traffic were captured from Dexter, Kazy, Locky, Parite, and WannaCry malware families. Samples of each family were searched in the VirusTotal database and used for the dataset. Each sample collected was detected by at least 47 anti-virus software and identified under one of the mentioned malware families by at least one of the anti-virus software. Table 1 shows the malware hash samples collected to create TorMal2019.

The setup to generate the datasets was adopted from the work of Lashkari et al. [19]. Two virtual machines (VMs) were set up as the client workstation and gateway where Windows XP SP3 32-bit and Whonix were used, respectively. VirtualBox 6.0.10 was used as the hypervisor where it was installed on the host operating system Ubuntu 18.04. Whonix is an open source operating system bundled with two VMs: a workstation and a gateway which uses the Tor network. Though it consists of a workstation and a gateway, only the Whonix gateway was used for this study. A licensed Windows XP VM, on the other hand, was configured to act as a workstation instead of the Whonix workstation. This choice is because most malware are written to attack Windows machines and not Linux-based machines, such as that of the Whonix workstation [2, 22].

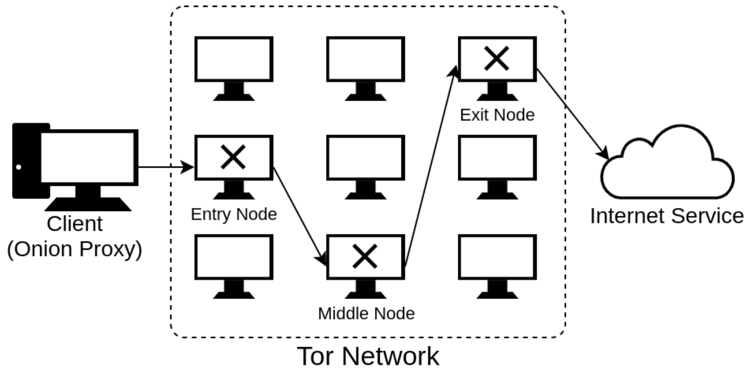


Figure 1: A simplified Tor architecture that anonymously connects the client to the internet.

Table 1: Malware hash samples used in TorMal2019.

Malware	Hash (SHA256)
Dexter	49cd5c4a5b94db4243f67517d8e0eb3fef4e975cf055033eaac9a63e6b2739a9
Kazy	0a576137cef647a59cf341180a07807f50cc1781125b0c7b505067083777480a
	1f8f8766e297716c61c4afa6508f0d214e4d2d72ee7752e3a9dc931737501a28
Locky	5b712f3ced695dd1510320494ecac67b277c0b386ee465303504c89431f87c78
	5bf84469051c85bd684e03eb46f774cb1e913884c95acf7b210a8a4469da8d9f
Parite	4e37fa3eb78d435e8d2773a7e8f2a64b11e1f7a78e20085ac90d2ae1f3b3d9b2
	5cf65eedc95b1da835f5ee03f42511b02c25d7f95f80524fe3d5c5f7685d3e58
WannaCry	4b6ae9f815889068f00c485cf3dd4588869cea36518d0ecf1af39bd146dfdc92
	09a46b3e1be080745a6d8d88d6b5bd351b1c7586ae0dc94d0c238ee36421cafa

The Windows XP workstation was configured [2] to use an internal virtual network interface that only connects to the Whonix gateway. Its firewall service was disabled to allow malware to freely infect the workstation and connect to the internet. Figure 2 illustrates how the traffic passes from the workstation to the gateway. By default, the Whonix gateway is configured to use two virtual network interfaces where one is connected to an internal network that connects with the workstation and the other is connected through a bridged interface to access the internet using the Tor network. The gateway basically acts as the onion proxy for the client workstation that routes traffic to a Tor relay. Figure 2 shows the system setup for capturing Tor and non-Tor traffic.

Generating traffic for every malware sample was done separately to easily label the data. Two types of traffic were collected: Tor and non-Tor. For every malware and non-malware sample ran on the workstation, two Wireshark or tcpdump instances were running in the gateway, collecting the incoming (non-Tor) traffic from the internal network and outgoing (Tor) traffic to the NAT network. Each run lasted for five minutes. After every run, the workstation is reset back to its initial state to ensure that the data collected on the previous run does not provide residual and combinatorial effects on the data on the next run. Resetting to the initial state was done by saving the snapshot[3] of the initially configured state of the workstation which was used to restore the workstation when needed. The collection of data was replicated three times to ensure statistical relevance and capture the variability brought about by each specific run.

4.2 Malware and Non-malware Traffic Classification

Figure 3 shows the layout of the experiment done to classify malware and non-malware traffic over the Tor network. From the TorMal2019 dataset, flows were generated to extract 76 flow-based features. A flow timeout of 15 seconds was used in generating flows from Tor traffic. To filter relevant features, the combination of CFS Subset Evaluator and Greedy Stepwise were used for feature selection. These are already implemented in Weka. C4.5, KNN, Naive Bayes, and Random Forest were built as classifiers. These classifiers were also implemented in Weka where the default parameters set in Weka were adopted for the experiment. Table 2 shows the list of default parameters used. Precision, recall, false positive rate, and false negative rate were the metrics used for the evaluation.

To evaluate what classifier is best for determining the metrics, an analysis of variance (ANOV) was used as given by the statistical model

$$M_{i,j} = \rho_i + \mu_j + \epsilon_{i,j}, \tag{1}$$

where $M_{i,j}$ is the metric's output of the jth classifier for the ith replicate, ρ_i is the mean effect of the ith replicate, μ_j is the mean effect of the jth classifier, and $\epsilon_{i,j}$ is the error contributed by the

[2]https://www.whonix.org/wiki/Other_Operating_Systems
[3]https://www.techrepublic.com/article/how-to-use-snapshots-in-virtualbox/

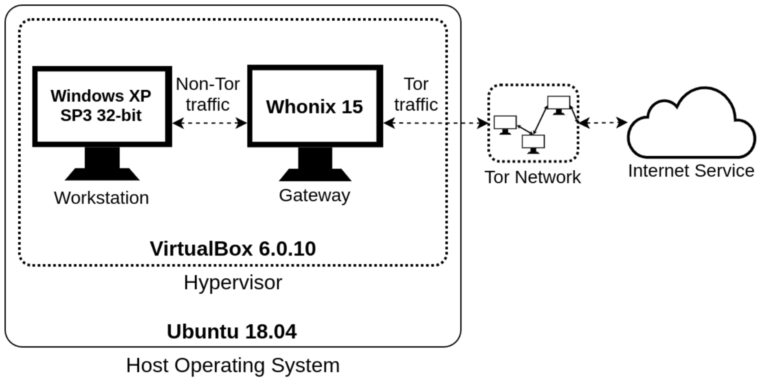


Figure 2: The setup for capturing Tor and non-Tor traffic of every traffic type and malware sample.

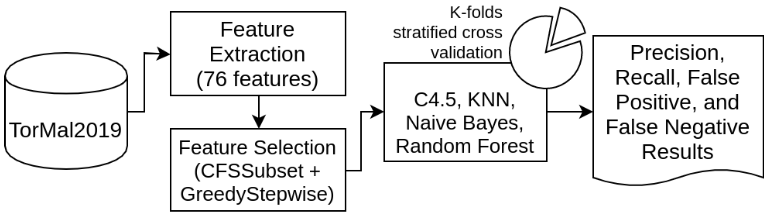

TorMal2019
Feature Extraction (76 features)
Feature Selection (CFSSubset + GreedyStepwise)
K-folds stratified cross validation
C4.5, KNN, Naive Bayes, Random Forest
Precision, Recall, False Positive, and False Negative Results

Figure 3: The experimental layout done on Tor malware and non-malware traffic classification.

Table 2: List of parameters with their description used for C4.5, KNN, Naive Bayes, and Random Forest classifiers.

Classifier	Parameters	Description
C4.5	C=0.25	Confidence threshold for pruning
	M=2	Minimum number of instances per
KNN	K=1	Number of nearest neighbors K
	A="EuclideanDistance"	Nearest neighbour search algorithm
Naive Bayes	None	
Random Forest	P=100	Size of each bag, as a percentage of the training set size
	I=100	Number of iterations
	num-slots=1	Number of execution slots
	K=0	Number of attributes to randomly investigate
	M=1	Minimum number of instances per leaf
	V=0.001	Minimum numeric class variance proportion of train variance for split
	S=1	Seed for random number generator

*i*th replicate and the *j*th classifier [17]. ANOV tests the following null hypotheses:

(1) H_0^{ρ}: That the mean effects of replicates are the same at a confidence level of $\alpha = 0.05$:
i.e., $\rho_1 = \rho_2 = \cdots = \rho_3$; and
(2) H_0^{μ}: That the mean effects of classifiers are the same at $\alpha = 0.05$:
i.e., $\mu_1 = \mu_2 = \cdots = \mu_4$.

These means that the corresponding alternate hypotheses are:

(1) H_1^{ρ}: That at least one pair of replicates has different means for all possible pairs of replicates at $\alpha = 0.05$:
i.e., $\exists\ \rho_m \neq \rho_n,\ \forall\ m \neq n$; and
(2) H_1^{μ}: That at least one pair of classifiers has different means for all pairs of classifiers at $\alpha = 0.05$:
i.e., $\exists\ \mu_m \neq \mu_n,\ \forall\ m \neq n$.

The ANOV for each metric was summarized in a table similar to Table 3. The F values were tested against $\alpha = 0.05$. For each source of variation, we accepted H_0 and rejected the corresponding H_1 if the probability of F, denoted as $P(F)$, is found to be greater than α. We rejected H_0 and accepted the corresponding H_1 if $P(F) \leq \alpha$.

Table 3: A sample ANOV table for the metric M.

Sources of Variation	DF	Sum of Squares	Mean Square	F
ρ	2			
μ	3			
ϵ	6			
Total	11			

For any source of variation with an F statistic that is less than $\alpha = 0.05$, a mpairwise comparison of means using Duncan's multiple range test (DMRT) was employed to see which means provide the significant variation [13]. Even though the standard pairwise comparison of means for ANOV is Fisher's Least Significant Difference (LSD) test [14, 25], DMRT was used because it is more strict than LSD.

4.3 System Development

To automate the workflow, TracTor system was created by integrating CICFlowMeter (2017) and Weka [16]. Figure 4 shows the architecture of the system. There are two main functions in TracTor: feature extraction and traffic classification.

Feature extraction involves generating flows and extracting flow-based features using CICFlowMeter. This is done by aggregating packets with the same 5-tuple values and computing different properties from it. The flow timeout is done by dividing the flow into 15-second intervals. Flow labelling and flow merging were also developed as additional features of TracTor. Flows were automatically labelled by obtaining the first substring of the filename containing the Tor packets before the first dot (e.g., malware in the filename malware.Dexter.pcap). For multiple files, TracTor still produces an output file for every input, but will also merge the content of those output files to a single attribute relation file format (arff) file. Traffic classification involves the use of Weka API to integrate test options, feature selection, and ML algorithms.

5 RESULTS AND DISCUSSION

5.1 TorMal2019

TorMal2019 generated a total of around 200MB of data for every replicate. Figure 5 shows how Tor and non-Tor traffic were captured on the gateway VM while a WannaCry malware sample was running in the workstation VM. The Wireshark instance on the left side captured the traffic before encryption while the Wireshark instance on the right side captured the same traffic but is now encrypted through Tor. The series of packets from the Tor traffic shows that the traffic captured was the communication between the gateway VM (10.0.2.15) and a Tor entry node (173.249.8.113). Figure 6 shows the state of the workstation VM as a WannaCry malware sample infected it.

Table 4 shows the Tor malware and non-malware web traffic. The variations in the number of flows for malware traffic per replicate may be due to the unexpected behavior of malware sample for every execution.

Table 4: The number of flows generated for three replicates of TorMal2019.

Traffic Types	Rep 1	Rep 2	Rep 3	Mean
Malware	164	189	211	188.000
Non-malware	41	39	40	40.000

5.2 Malware and Non-Malware Traffic Classification

Table 5 shows the filtered features used for testing TorMal2019. From 76 features, they were filtered to 8, 4, and 4 for each replicate, respectively. Among those features, only 2 were common from the three results.

Table 6 shows the precision, recall, false positive, and false negative values obtained from the experiment. Results show that all classifiers obtained high precision and recall values of above 0.95.

Table 5: The list of features selected using CFS+Greedy algorithm for each TorMal2019 replicate.

	Rep 1	Rep 2	Rep 3
1	TotLen Fwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Max
2	Fwd Pkt Len Mean	Fwd IAT Min	Fwd IAT Min
3	Fwd Pkt Len Std	Pkt Len Std	Pkt Len Std
4	Flow IAT Max	Fwd Act Data Pkts	Fwd Act Data Pkts
5	Fwd IAT Min		
6	Pkt Len Std		
7	Down/Up Ratio		
8	Subflow Fwd Byts		

The results of ANOV to determine which classifier is best for measuring precision, recall, false positive, and false negative values are respectively shown in Tables 7 through 10.

For all ANOV tables (Tables 7 through 10), the F statistics for μ (i.e., classifiers) for the respective metrics precision, recall, false positive, and false negative were found to be not significant at $\alpha = 0.05$. This means that we respectively accept each null hypotheses for μ: The classifiers perform statistically the same compared to each other in measuring the metrics. In Table 9, the F statistics for replication ρ was found to be statistically significant at $\alpha = 0.05$. Here. we reject the null hypothesis and accept the alternate that at least one pair of replicates are not statistically the same with each other in measuring false positives. However, since we are much more interested in false negatives (i.e., classifying malware as non-malware) than false positives (i.e., classifying non-malware as malware), we argue that the differences in replicates for measuring false positives is not much of importance. In other words, it is quite acceptable for us to "err on the side of caution."

Figure 7 shows the computed mean μ and standard deviation σ of false negatives. Results of C4.5, KNN, Naive Bayes, and Random Forest have overlapped, e.g.,

$$[\mu - \sigma, \mu + \sigma]_{\text{C4.5}} \cap [\mu - \sigma, \mu + \sigma]_{\text{Naive Bayes}} \neq \emptyset,$$

where $[\mu - \sigma, \mu + \sigma]_{\text{C4.5}}$ is the inclusive range $\mu \pm \sigma$ for C4.5 and $[\mu - \sigma, \mu + \sigma]_{\text{Naive Bayes}}$ for Naive Bayes. This may mean that all

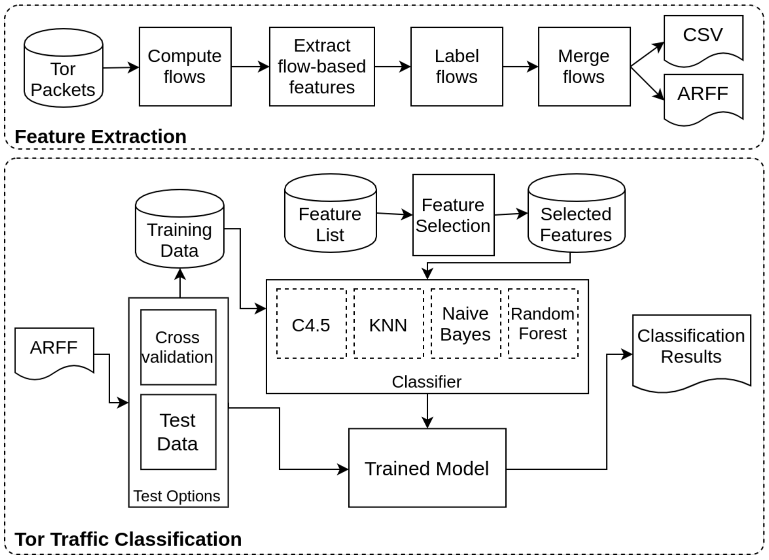
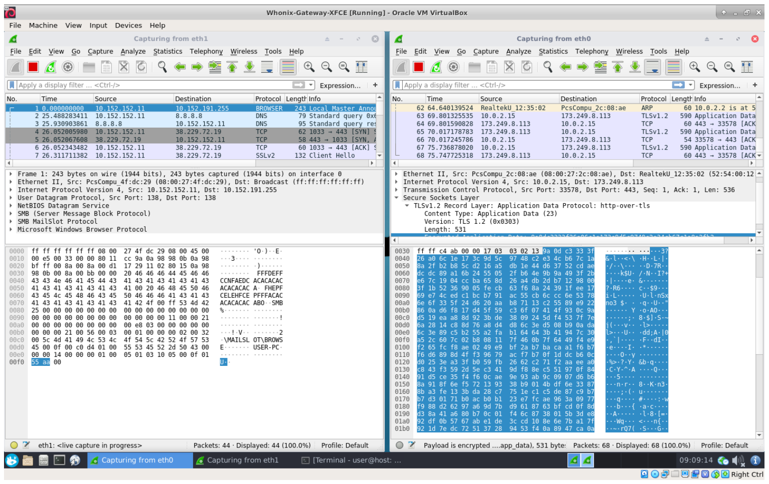


Figure 4: The overview of TracTor consisting two main functions: feature extraction and traffic classification.


Whonix-Gateway-XFCE [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Capturing from eth1
Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Capturing from eth0
Capturing from eth1
[Terminal - user@host: ...
09:09:14
Right Ctrl

```
No.  Time          Source          Destination      Protocol Length Info
  1  0.000000000   10.152.152.11   10.152.191.255   BROWSER   243 Local Master Annou
  2 25.488283411   10.152.152.11   8.8.8.8          DNS        79 Standard query 0x6
  3 25.930903861   8.8.8.8         10.152.152.11    DNS        95 Standard query res
  4 26.052005980   10.152.152.11   38.229.72.19     TCP        62 1033 → 443 [SYN] S
  5 26.052067608   38.229.72.19    10.152.152.11    TCP        58 443 → 1033 [SYN, A
  6 26.052343482   10.152.152.11   38.229.72.19     TCP        60 1033 → 443 [ACK] S
  7 26.311711382   10.152.152.11   38.229.72.19     SSLv2     132 Client Hello

▸ Frame 1: 243 bytes on wire (1944 bits), 243 bytes captured (1944 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_4f:dc:29 (08:00:27:4f:dc:29), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Internet Protocol Version 4, Src: 10.152.152.11, Dst: 10.152.191.255
▸ User Datagram Protocol, Src Port: 138, Dst Port: 138
▸ NetBIOS Datagram Service
▸ SMB (Server Message Block Protocol)
▸ SMB MailSlot Protocol
▸ Microsoft Windows Browser Protocol

0000  ff ff ff ff ff ff 08 00  27 4f dc 29 08 00 45 00   ········ 'O·)··E·
0010  00 e5 00 33 00 00 80 11  cc 9a 0a 98 98 0b 0a 98   ···3···· ········
0020  bf ff 00 8a 00 8a 00 d1  17 29 11 02 80 15 0a 98   ········ ·)······
0030  98 0b 00 8a 00 bb 00 00  20 46 46 46 44 45 46 46   ········  FFFDEFF
0040  43 43 4e 46 41 45 44 43  41 43 41 43 41 43 41 43   CCNFAEDC ACACACAC
0050  41 43 41 43 41 43 41 43  41 00 20 46 48 45 50 46   ACACACAC A· FHEPF
0060  43 45 4c 45 48 46 43 45  50 46 46 46 41 43 41 43   CELEHFCE PFFFACAC
0070  41 43 41 43 41 43 41 43  41 42 4f 00 ff 53 4d 42   ACACACAC ABO··SMB
0080  25 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   %······· ········
0090  00 00 00 00 00 00 00 00  00 00 00 00 11 00 00 21   ········ ·······!
00a0  00 00 00 00 00 00 00 00  00 e8 03 00 00 00 00 00   ········ ········
00b0  00 00 00 21 00 56 00 03  00 01 00 00 00 02 00 32   ···!·V·· ·······2
00c0  00 5c 4d 41 49 4c 53 4c  4f 54 5c 42 52 4f 57 53   ·\MAILSL OT\BROWS
00d0  45 00 0f 00 c0 d4 01 00  55 53 45 52 2d 50 43 00   E······· USER-PC·
00e0  00 00 14 00 00 00 01 00  05 01 03 10 05 00 0f 01   ········ ········
00f0  55 aa 00                                           U·

eth1: <live capture in progress>   Packets: 44 · Displayed: 44 (100.0%)   Profile: Default
```

```
No.  Time          Source            Destination        Protocol Length Info
 62 64.640139524   RealtekU_12:35:02 PcsCompu_2c:08:ae  ARP        60 10.0.2.2 is at 5
 63 69.801325535   10.0.2.15         173.249.8.113      TLSv1.2   590 Application Data
 64 69.801590828   173.249.8.113     10.0.2.15          TCP        60 443 → 33578 [ACK
 65 70.017178783   173.249.8.113     10.0.2.15          TLSv1.2   590 Application Data
 66 70.017245786   10.0.2.15         173.249.8.113      TCP        54 33578 → 443 [ACK
 67 75.736878020   10.0.2.15         173.249.8.113      TLSv1.2   590 Application Data
 68 75.747725318   173.249.8.113     10.0.2.15          TCP        60 443 → 33578 [ACK

▸ Ethernet II, Src: PcsCompu_2c:08:ae (08:00:27:2c:08:ae), Dst: RealtekU_12:35:02 (52:54:00:12
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 173.249.8.113
▸ Transmission Control Protocol, Src Port: 33578, Dst Port: 443, Seq: 1, Ack: 1, Len: 536
▾ Secure Sockets Layer
  ▾ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 531

0030  ff ff c4 ab 00 00 17 03  03 02 13 9a 0d c3 33 3f   ········ ·····3?
0040  26 a0 6c 1e 17 3c 9d 5c  97 48 c2 e3 4c b6 7c 1a   &·l··<·\ ·H··L·|·
0050  8a 2f b2 b8 5c d2 16 a5  db 1e 44 d6 37 52 cd ae   ·/··\··· ··D·7R··
0060  dc dc 89 a1 6b 24 55 05  2f b6 4e 9b 9a 49 3f 2b   ····k$U· /·N··I?+
0070  e6 7c 19 04 cc ba 65 8d  26 a4 db 2d b7 12 98 00   ·|····e· &·······
0080  3f 1b 52 36 90 05 fe cb  63 f6 8a 24 39 1f ee 17   ?·R6···· c··$9···
0090  69 e7 4c ed c1 bc b7 91  ac 55 cb 6c cc 6e 53 78   i·L····· ·U·l·nSx
00a0  6e 6f 33 5f 24 d6 20 aa  b8 71 13 c2 55 89 e9 22   no3 $· · ·q··U··"
00b0  86 0a d6 f8 17 d4 5f 59  c3 6f 07 41 4f 93 0c 9a   ······ Y ·o·AO···
00c0  d5 19 ea a8 8d 92 3b de  38 09 24 5d f4 53 7f 7e   ······;· 8·$]·S·~
00d0  6a 28 14 c8 8d 76 a8 d4  d8 6c 3e d5 08 b9 0a da   j(···v·· ·l>·····
00e0  6c 3e 89 c5 b2 55 a2 fa  b1 64 64 3b 41 94 7c 30   l>···U·· ·dd;A·|0
00f0  a5 2c 60 7c 02 b8 08 11  7f 46 0b 7f 64 49 f4 e9   ·,`|···· ·F··dI··
0100  f2 65 fc f8 ae 02 49 e9  bf 2a b7 ba ca a1 f6 b7   ·e····I· ·*······
0110  f6 d6 89 8d 4f f3 96 79  ac f7 b7 0f 1d dc b6 0c   ····O··y ········
0120  d0 25 3e a3 3f b0 59 fb  26 62 c2 71 f2 aa ee a0   ·%>·?·Y· &b·q····
0130  c8 43 f3 59 2d 5e c3 41  9d f8 8e c5 51 97 0f 84   ·C·Y-^·A ····Q···
0140  91 d5 ce 35 f4 f6 0c ae  9e 93 ab 9c 09 07 d6 b6   ···5···· ········
0150  8a 91 8f 6e f5 72 13 93  38 b9 01 4b df 6e 33 87   ···n·r·· 8··K·n3·
0160  8b a3 fe 13 3b da 28 c7  75 1e c1 c5 de 87 c9 b7   ····;·(· u·······
0170  b7 d3 01 71 b0 ac b0 b1  23 e7 fc ae 96 3a 09 77   ···q···· #····:·w
0180  f9 88 d2 62 97 a6 9d 7b  d9 61 87 63 bf cd 0f 8d   ···b···{ ·a·c····
0190  d3 8a 41 a6 80 b7 0c 01  f4 6c 87 38 01 5b 3d e8   ··A····· ·l·8·[=·
01a0  92 df 0b 57 67 ab e1 de  3c cd 10 8e 6e 7b a1 7f   ···Wg··· <···n{··
01b0  92 1d 7e dc 72 51 37 28  94 53 f4 0a 89 47 ca 0a   ··~·rQ7( ·S···G··

Payload is encrypted ....app_data), 531 byte:   Packets: 68 · Displayed: 68 (100.0%)   Profile: Default
```

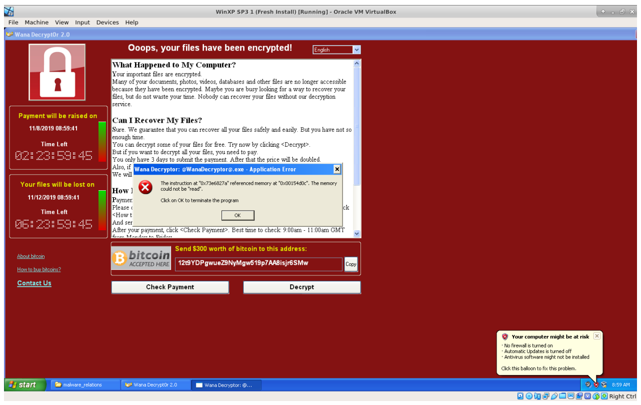
Figure 5: A screenshot of two Wireshark instances running to capture Tor (right side) and non-Tor (left side) traffic.

classifiers are statistically equivalent in minimizing malware types being classified as non-malware. However, Random Forest obtained the smallest spread of values from the replicates around the mean. This observation was statistically verified from a pairwise comparison of means using DMRT at the same $\alpha = 0.05$. This verification is albeit redundant as the F statistics from Table 10 already shown that the classifiers' performances are not statistically different from each other.

5.3 TracTor: A System to Automate the Computational Workflow for Tor Traffic Classification

TracTor was developed to automate the flow-based feature extraction and traffic classification process. It can also be used for other types of network traffic that utilizes flow-based features and machine-learning algorithms. There are two main sections in TracTor: extract flows and classify traffic.

5.3.1 Extract Flows. Figure 8 shows a screenshot of the TracTor extract flows section. This section takes a pcap file or directory as input and produces an output in csv and arff file formats. The



Ooops, your files have been encrypted!

What Happened to My Computer?
Your important files are encrypted.
Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?
Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if
We will

How I
Paymer
Please ck
<How t
And ser
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am GMT

Wana Decryptor: @WanaDecryptor@.exe - Application Error
The instruction at "0x73e6827a" referenced memory at "0x00154d0c". The memory could not be "read".
Click on OK to terminate the program

Send $300 worth of bitcoin to this address:
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw

Your computer might be at risk
· No firewall is turned on
· Automatic Updates is turned off
· Antivirus software might not be installed
Click this balloon to fix this problem.

Figure 6: A screenshot of a WannaCry malware sample infecting the workstation VM.

Table 6: Precision, recall, false positive, and false negative results obtained from testing TorMal2019 replicates using 10-folds cross validation.

Classifier	Precision	Recall	FP Rate	FN Rate
	Rep 1			
C4.5	0.976	0.988	0.100	0.012
KNN	0.970	0.988	0.125	0.012
NB	0.982	0.976	0.075	0.024
RF	0.976	0.944	0.100	0.006
	Rep 2			
C4.5	0.935	0.995	0.333	0.005
KNN	0.964	0.979	0.179	0.021
NB	0.931	1.000	0.359	0
RF	0.959	0.995	0.205	0.005
	Rep 3			
C4.5	0.986	0.995	0.075	0.005
KNN	0.986	0.972	0.075	0.028
NB	0.986	0.986	0.075	0.014
RF	0.977	0.991	0.125	0.009

Table 7: ANOV table for determining the best classifier for measuring precision.

Sources of Variation	DF	Sum of Squares	Mean Square	F
ρ	2	0.00296	0.00147	10.49^{ns}
μ	3	0.00012	0.00004	0.28^{ns}
ϵ	6	0.00085		
Total	11	0.00393		

^{ns} Not significant at $\alpha = 0.05$.

pre-set list of dataset labels is an editable list of comma-separated values used to filter valid labels in the filenames of the dataset.

Table 8: ANOV table for determining the best classifier for measuring recall.

Sources of Variation	DF	Sum of Squares	Mean Square	F
ρ	2	0.00077	0.00039	1.68^{ns}
μ	3	0.00062	0.00021	0.89^{ns}
ϵ	6	0.00138		
Total	11	0.00277		

^{ns} Not significant at $\alpha = 0.05$.

Table 9: ANOV table for determining the best classifier for measuring false positive.

Sources of Variation	DF	Sum of Squares	Mean Square	F
ρ	2	0.08221	0.04111	10.51^{*}
μ	3	0.00404	0.00135	0.34^{ns}
ϵ	6	0.02348		
Total	11	0.10973		

^{ns} Not significant at $\alpha = 0.05$.
* Significant at $\alpha = 0.05$.

A text area for output is also included to show the details of the feature extraction.

5.3.2 *Classify Traffic.* Figure 9 shows a screenshot of the TracTor classify traffic section. This section takes an arff file as input and provides details of classification results as output. Users can choose from the different test options provided. By default, 10-folds stratified cross validation is performed for the Cross Validation test option. Another option is the Test Data where TorMal2019 can be used as training data or the user can provide a training data or model. Once the training data is specified, TracTor performs feature selection and automatically checks the filtered features in the features list. Users can also choose which classifier to use on a certain

Table 10: ANOV table for determining the best classifier for measuring false negative.

Sources of Variation	DF	Sum of Squares	Mean Square	F
ρ	2	0.00010	0.00005	0.79 ^{ns}
μ	3	0.00036	0.00012	1.97 ^{ns}
ϵ	6	0.00036		
Total	11	0.00082		

^{ns} Not significant at $\alpha = 0.05$.

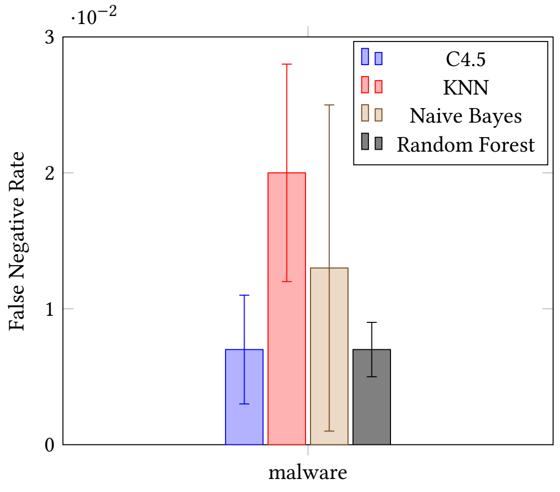


Figure 7: Mean and standard deviation error bar of malware false negative rate computed from testing TorMal2019.

test, unless a user-supplied model is loaded. TracTor also has an option to save a model that can be reused for testing. Classification results provide computations of the following performance metrics: Accuracy, kappa statistic, mean absolute error, root mean squared error, relative absolute error, and root relative squared error. TP rate, FP rate, precision, recall, f-measure, MCC, ROC Area, and PRC Area were also computed for every traffic type.

6 CONCLUSION AND FUTURE WORK

Nowadays, internet anonymity is a growing need for users who want to protect their identity and avoid network surveillance. Tor is one of the most popular anonymity tool today where it implements the concept of onion routing. Since anyone can use Tor, it can be used for legitimate and illegitimate purposes. Unfortunately, Tor has already been used for malicious purposes, one is hiding malware traffic.

This study presented how the TorMal2019 dataset was generated to serve as input to the classification of malware and non-malware traffic that passes through the Tor network. A system called TracTor was also developed to automate the classification (using flow-based features) workflow by integrating CICFlowMeter and Weka. Lastly, through experimentation, results showed that all classifiers have the same performance statistically. Even though Random Forest posted with high precision and recall values with the lowest false negative rate, it performed statistically the same as the others. Results also proved that malware and non-malware traffic can be successfully classified using only flow-based features.

For future work, additional features and other features selection algorithms can be implemented on TracTor. This can be done to further minimize false negative rate of malware classification. Moreover, TracTor can be extended to classify real-time Tor traffic by utilizing its load and save model features.

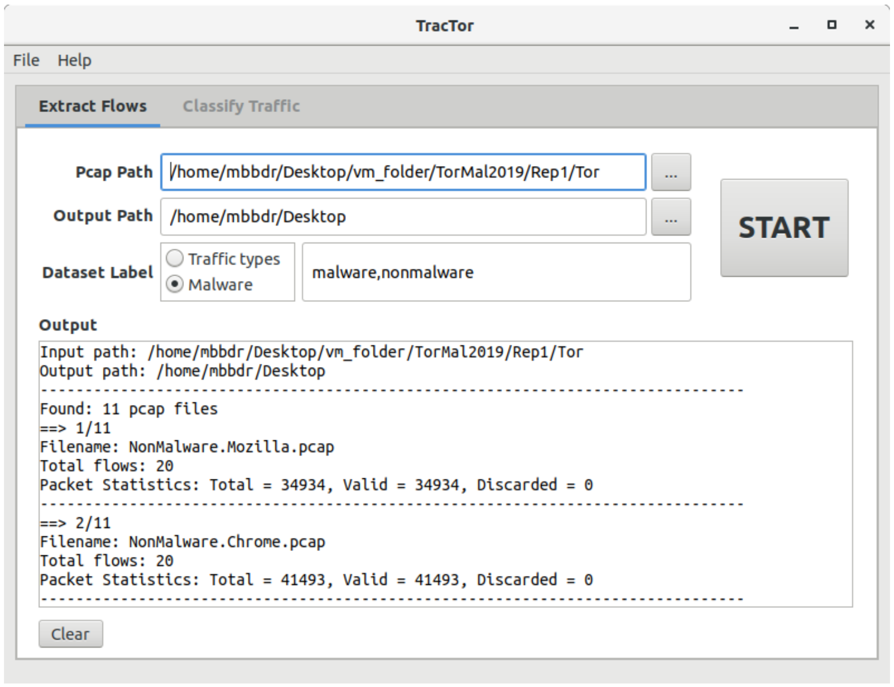
ACKNOWLEDGMENTS

The authors would like to thank VirusTotal for providing the malware samples for this study.

The authors would also like to thank Prof. Jaime M. Samaniego, Prof. Concepcion L. Khan, and the members of Systems Research Group (SRG) of our institute for their valuable comments and helpful suggestions.

REFERENCES

[1] [n. d.]. VirusTotal. https://www.virustotal.com/gui/home/.
[2] 2012. Whonix. https://www.whonix.org/. Whonix 15 64-bit.
[3] 2015. Onion Routing. https://www.onion-router.net/.
[4] 2017. CICFlowMeter. http://www.netflowmeter.ca/netflowmeter.html. Canadian Institute for Cybersecurity.
[5] Areej Al-Bataineh and Gregory White. 2012. Analysis and detection of malicious data exfiltration in web traffic. In *2012 7th International Conference on Malicious and Unwanted Software*. IEEE, 26–31.
[6] Blake Anderson, Subharthi Paul, and David McGrew. 2018. Deciphering malware's use of TLS (without decryption). *Journal of Computer Virology and Hacking Techniques* 14, 3 (2018), 195–211.
[7] Avast Threat Intelligence Team. 2006. A deep and technical look into the latest ransomware called Locky. https://blog.avast.com/a-closer-look-at-the-locky-ransomware.
[8] Ulrich Bayer, Imam Habibi, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. 2009. A View on Current Malware Behaviors. In *2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*.
[9] Dmitri Bekerman, Bracha Shapira, Lior Rokach, and Ariel Bar. 2015. Unknown malware detection using network traffic classification. In *2015 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 134–142.
[10] T Dübendorfer and Bernhard Plattner. 2005. Host behaviour based early detection of worm outbreaks in internet backbones. In *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)*. IEEE, 166–171.
[11] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. *Tor: The second-generation onion router.* Technical Report. Naval Research Lab Washington DC.
[12] Falko Dressler, Wolfgang Jaegers, and Reinhard German. 2007. Flow-based worm detection using correlated honeypot logs. In *Communication in Distributed Systems-15. ITG/GI Symposium*. VDE, 1–6.
[13] David B. Duncan. 1955. Multiple range and multiple F tests. *Biometrics* 11, 1 (1955), 1–42.
[14] Ronald A. Fisher. 1935. *The Design of Experiments.* Oliver and Boyd, Edinburgh. 252 pages.
[15] Guofei Gu, Junjie Zhang, and Wenke Lee. 2008. BotSniffer: Detecting botnet command and control channels in network traffic. (2008).
[16] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter* 11, 1 (2009), 10–18.
[17] Chihiro Hirotsu. 2017. *Advanced Analysis of Variance.* John Wiley and Sons. 415 pages.
[18] Jarilyn M Hernández Jiménez and Katerina Goseva-Popstojanova. 2018. The Effect on Network Flows-Based Features and Training Set Size on Malware Detection. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. IEEE, 1–9.
[19] Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. 2017. Characterization of Tor Traffic using Time based Features.. In *3rd International Conference on Information Systems Security and Privacy (ICISSP)*. 253–262.
[20] Zhen Ling, Junzhou Luo, Kui Wu, Wei Yu, and Xinwen Fu. 2014. TorWard: Discovery of malicious traffic over Tor. In *IEEE INFOCOM 2014-IEEE Conference*

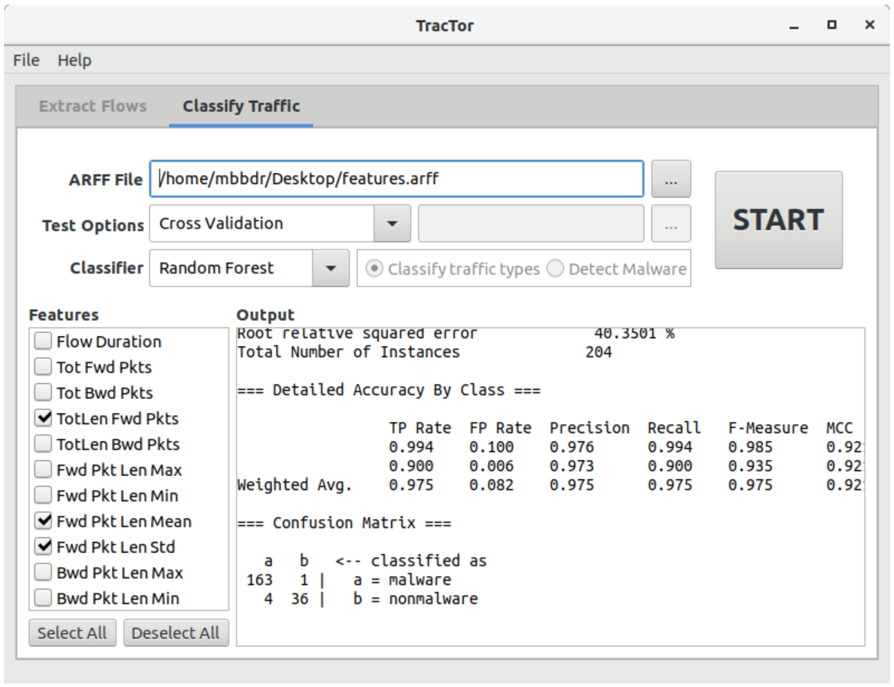


Pcap Path: /home/mbbdr/Desktop/vm_folder/TorMal2019/Rep1/Tor
Output Path: /home/mbbdr/Desktop
Dataset Label: Traffic types [✓ Malware] malware,nonmalware

Output

```
Input path: /home/mbbdr/Desktop/vm_folder/TorMal2019/Rep1/Tor
Output path: /home/mbbdr/Desktop
--------------------------------------------------------------------------------
Found: 11 pcap files
==> 1/11
Filename: NonMalware.Mozilla.pcap
Total flows: 20
Packet Statistics: Total = 34934, Valid = 34934, Discarded = 0
--------------------------------------------------------------------------------
==> 2/11
Filename: NonMalware.Chrome.pcap
Total flows: 20
Packet Statistics: Total = 41493, Valid = 41493, Discarded = 0
--------------------------------------------------------------------------------
```

Figure 8: A screenshot of TracTor used for extracting flow-based features.



ARFF File: /home/mbbdr/Desktop/features.arff
Test Options: Cross Validation
Classifier: Random Forest [✓ Classify traffic types] Detect Malware

Features
- [ ] Flow Duration
- [ ] Tot Fwd Pkts
- [ ] Tot Bwd Pkts
- [x] TotLen Fwd Pkts
- [ ] TotLen Bwd Pkts
- [ ] Fwd Pkt Len Max
- [ ] Fwd Pkt Len Min
- [x] Fwd Pkt Len Mean
- [x] Fwd Pkt Len Std
- [ ] Bwd Pkt Len Max
- [ ] Bwd Pkt Len Min

Output

```
Root relative squared error             40.3501 %
Total Number of Instances              204

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC
                 0.994    0.100    0.976      0.994    0.985      0.92
                 0.900    0.006    0.973      0.900    0.935      0.92
Weighted Avg.    0.975    0.082    0.975      0.975    0.975      0.92

=== Confusion Matrix ===

   a   b   <-- classified as
 163   1 |   a = malware
   4  36 |   b = nonmalware
```

Figure 9: A screenshot of TracTor used for classifying Tor traffic using the Random Forest classifier.

on Computer Communications. IEEE, 1402–1410.

[21] Zhen Ling, Junzhou Luo, Kui Wu, Wei Yu, and Xinwen Fu. 2015. Torward: Discovery, blocking, and traceback of malicious traffic over tor. *IEEE Transactions on Information Forensics and Security* 10, 12 (2015), 2515–2530.

[22] MalwareTech. 2017. Creating a Simple Free Malware Analysis Environment. https://www.malwaretech.com/2017/11/creating-a-simple-free-malware-analysis-environment.html.

[23] T. T.T. Nguyen and G. Armitage. 2008. A Survey of Techniques for Internet Traffic Classification Using Machine Learning. *Commun. Surveys Tuts.* 10, 4 (Oct. 2008), 56–76. https://doi.org/10.1109/SURV.2008.080406

[24] Vern Paxson. 1994. Empirically derived analytic models of wide-area TCP connections. *IEEE/ACM transactions on Networking* 2, 4 (1994), 316–336.

[25] Sabra Sultana, Muhammad Mutahir Iqbal, and Munir Akhtar. 2013. A visualization of Fisher's Least Significant Difference test. *Pakistan Journal of Commerce and Social Sciences* 17, 1 (2013), 100–106.