Network Traffic Classification in Encrypted Environment: A Case Study of Skype

A PROJECT REPORT

Submitted in partial fulfillment of the

requirements for the award of the degrees

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Utkarsh Saxena(130001038) **Harshil Shah(130002035)**

Guided by:

Dr. Neminath Hubballi

INDIAN INSTITUTE OF TECHNOLOGY INDORE

November 2016

CANDIDATE'S DECLARATION

We hereby declare that the project entitled **"Network Traffic Classification in Encrypted Environment: A Case Study of Skype"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Computer Science Engineering' completed under the supervision of **Dr. Neminath Hubballi, Assistant Professor,** IIT Indore is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

Signature and name of the student(s) with date

CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the students is correct to the best of my knowledge.

Signature of BTP Guide(s) with dates and their designation

Preface

This report on “Network Traffic Classification in Encrypted Environment: A Case Study of Skype" is prepared under the guidance of **Dr. Neminath Hubballi**.

Through this report we tried to implement a new approach for Skype traffic classification. We researched the existing approaches and analysed their shortcomings. The algorithm which we proposed is a novel work in the traffic classification research area and can find many real life applications.

We tried to the best of our abilities and knowledge to explain the content in a lucid manner using appropriate figures and tables.

Utkarsh Saxena

Harshil Shah

B.Tech. IV Year

Discipline of Computer Science and Engineering

IIT Indore

Acknowledgements

We wish to thank Dr. Neminath Hubballi for his kind support and valuable guidance throughout the duration of the project.

It is his help and support, due to which we were able to complete the design and technical report.

Without his support this report would not have been possible.

Utkarsh Saxena

Harshil Shah

B.Tech. IV Year

Discipline of Computer Science and Engineering

IIT Indore

Abstract

Skype is a very popular P2P VoIP application that has drawn the great attention of telecom operators and research communities. However, its source code, protocols and algorithms are not disclosed. Moreover, Skype uses strong encryption and random port numbers to disguise its presence making Skype traffic classification a challenging task. In this report, we present a unique sequence signatures based on payload lengths of TCP packets to implement host-level identification system for Skype traffic classification. We present experimental results using methods based on sequence signatures to identify Skype traffic and nodes. We show that Skype node and its related traffic can be easily identified by observing 5 minutes of Skype traffic flows with recall and precision better than 99%.

Contents

List of Figures

1

Introduction

Traffic classification [1] is the task of associating network traffic with the generating application. Traffic classification provides extremely valuable information. It can be used to identify trends in application usage for a better network design so that network administrators can know which application does packets belong to for providing different services to their clients. It is used for security operations like triggering alarms when unwanted traffic has been detected or filtering unwanted traffic. Quality of Service (QoS) is used to prioritize and treat traffic differently. To achieve QoS, first step is to classify traffic into different classes using traffic classification. It is used for lawful interception of illegal or critical traffic and anomaly detection for the identification of malicious use of network resources. Traffic classification is used by ISP also to perform lawful interception as required by some governments.

1.1 Characteristics of Classifier

The most important properties of traffic classifier that determines their applicability are:

- **Granularity:** Coarse grained classifiers are only able to classify large family of protocols(P2P vs non P2P, HTTP vs Streaming), whereas fine grained classifiers can classify specific protocols(eg. BitTorrent) or application(eg. Skype, WhatsApp).

- **Timeliness:** Early classification techniques need only few packets for classification and can classify traffic quickly, whereas late classification techniques take longer time for classification and may require collecting large amount of data.

- **Computational Cost:** For classifiers, operations which require heavy computation resources are mainly regular expression matching or packet payload access. Computational power needed for classification is an important factor while comparing different classifiers.

1.2 Types of Classification

Traffic classification is divided into following types:

Port based classification simply extracts transport-layer ports value from the packet header and then associates it with the corresponding application. This method has become highly unreliable as applications have started to use non-standard ports and hiding behind ports assigned to other protocols.

Payload Based classification inspects the content of packets looking for distinctive hints of an application protocol in packet payloads. *Deep Packet Inspection* (DPI) techniques try to search for manually derived patterns, keywords and regular expressions in the payload of non encrypted packets. Though extremely accurate, it has a very high computational cost as it involves several packet memory access. On the other hand, *Stochastic Packet Inspection* (SPI) automatically computes patterns distinctive to a protocol. It can deal with partially encrypted traffic by studying the randomness/ entropy of payload.

Statistical Classification uses various flow level features like frequency and lengths of packets exchanged to characterize the traffic. These classifiers then apply various data mining tools on these features to classify the traffic.

As these algorithms do not access the payload of packets, they are much more lightweight and can also be applied on encrypted and obfuscated traffic.

Behavioural Classification analyzes all the traffic received by a host in the network. These classifiers try to identify the application running on the target host by only examining the generated traffic patterns. They work on the principle that different applications will generate different traffic patterns. For example, the number of concurrent TCP connections made by the host and number of peers in the network. Behavioral classifiers are also lightweight as they avoid access to packet payload.

1.3 Challenges

The challenges faced by traffic classification have been increasing tremendously. First, modern classifiers must use extremely lightweight algorithms with little computational requirements since amount of traffic that needs to be identified and transmission rates are increasing drastically. Secondly, developers of network applications have identified clever ways to hide their applications traffic from network operators. The methods like encryption of network traffic and using encapsulation of network traffic behind some standard protocols like HTTP have been quite successful in disguising network traffic. Skype also disguises its traffic and avoid its inspection by using various measures such as implementing proprietary protocols, payload encryption and random port numbers for data transmission. These factors make traffic identification a challenging task. Thus, researchers need to propose novel efficient ways to solve the problem of traffic classification.

The rest of this report is organized as follows. In chapter 2, we review the literature related to Skype traffic classification. We provide an overview of Skype network architecture in Chapter 3. In Chapter 4, we discuss the proposed method for Skype node identification and Skype traffic

classification. In chapter 5, we discuss the design and implementation of our classifier. Experimental evaluations and their results are presented in Chapter 6 followed by the conclusion and future work in chapter 7.

2

Literature Survey

In this chapter, we discuss previous works in the field of traffic Classification of Skype.

Because of Skype's wide applications from the viewpoint of network optimal management, a lot of work has been done on Skype traffic classification [2, 3, 4, 5]. In [2], authors revealed unique sequence signatures of Skype UDP flows and implemented the system named SkyTracer for precise Skype traffic classification. The authors introduced for the first time the use of sequence signatures to carry out the precise Skype traffic Classification. As shown in Figure 1, the sequence signatures are shown as regular expressions of the third byte of payloads in a continuous sequence of packets that belong to a Skype UDP flow.

^\x02+[\x0d\x1d\x2d\x3d\x4d\x5d\x6d\x7d]*$
^\x02+[\x0f\x1f\x2f\x3f\x4f\x5f\x6f\x7f]+[\x0d\x1d\x2d\x3d\x4d\x5d\x6d\x7d]+$
^\x02+[\x05\x15\x25\x35\x45\x55\x65\x75]+[\x0d\x1d\x2d\x3d\x4d\x5d\x6d\x7d]+$

Figure 1: Sequence Signatures used by SkyTracer[2]

The authors also implemented SkyTracer for online Skype Identification. SkyTracer consists of two modules:

1. Skype Node Identification module: This module uses Skype Login Signal and IP address lookup of Skype Servers to identify the Skype node. Skype Login signals can be detected by presence of '0x02' at third position of all the packet payloads in this flow. For IP Address lookup, authors collected number of IP addresses of various Skype Servers providing various types of Skype service. To perform lookup, destination address of Skype flows is checked to identify the communication with Skype servers.
2. Fine Grained Classification module: This module uses Sequence Signatures matching to classify Skype UDP Flows from aggregated traffic.

In [3], authors presented a classification method based on two complementary techniques to reveal Skype traffic. In first approach, authors used Pearson's Chi-Square test to detect Skype's fingerprint in packet structure taking advantage of randomness present due to encryption. The test is designed to verify if the behavior of an object follows an expected behavior after observing it finite number of times. It can be done by computing the deviation of the observed output values with respect to the expected distribution of the output. In second approach, author used statistical properties of Skype traffic such as packet length and packet arrival rate to develop a decision process based on Naive Bayesian Classifier. However, classifier in this work needs full flow for examination, which is not suited for early classification mechanisms.

In [4], authors proposed a statistical method to rapidly identify the Skype traffic flows. They used different types of features for classifying the Skype traffic and also discussed effectiveness using different combinations of those features. The features used for classification are:

1. **Characteristic Packet Length**: Authors found that packets having length less than 80 bytes are restricted to a limited number of values. They occur frequently making them useful for Skype traffic classification.
2. **Interarrival Time**: Authors observed that skype client generates packet at intervals that are multiple of 16 milliseconds.
3. **Large Packet Statistics:** In this feature, authors used statistics across the whole time window under consideration on packet of length greater than or equal to 80 bytes. The statistics used were minimum, maximum, mean, standard deviation, etc.

However, this work focuses only on statistical properties of Skype traffic when using SVOPC codec.

In [5], authors analyzed robustness of different machine learning techniques for traffic classification of encrypted Skype traffic. For this purpose, five machine learning algorithms are tested - AdaBoost, Support Vector Machine, Naive Bayesian, RIPPER and C4.5. The features used by these algorithms are shown in Table 1.

Protocol	Duration of the Flow
#packets in forward direction	#Bytes in forward direction
#packets in backward direction	#Bytes in reverse direction
Min forward inter-arrival time	Min backward inter-arrival time
Std Deviation of forward inter-arrival times	Std Deviation of backward inter-arrival times
Mean forward inter-arrival time	Mean backward inter-arrival time

Max forward inter-arrival time	Max backward inter-arrival time
Max forward packet length	Max backward packet length
Min forward packet length	Min backward packet length
Mean backward packet length	Mean forward packet length

Table 1: Features [5] used for comparison between Machine Learning Algorithms.

Most of the works done for Skype traffic classification require complex calculations or regular expression matching [2, 3, 4, 5] however our method requires no such calculations which makes it computationally inexpensive. Most of the methods developed for Skype traffic classification relies mostly on UDP. So those methods do not work behind UDP restricted firewalls. The method we proposed relies only on TCP for the classification. Moreover, our method requires no access to payload bytes as compared to other payload based classification methods[2, 3] which requires several access to payload memory. Also, our method is able to differentiate between Skype traffic in IDLE period and VOICE traffic.

3

Skype Overview

In this chapter, we focus on discussing the Skype architecture by reviewing the works done in revealing the Skype architecture. The classification of traffic from VoIP applications like Skype has already been studied by several authors in the past few years. A few studies relied on understanding the protocol of Skype.

In [6], authors gave a detailed overview of Skype architecture and functionality such as login, NAT and firewall traversal, call establishment, media transfer, codecs and conferencing under different network setups. In Skype, only user's authentication is performed using Client-Server model. Once the user is successfully logged in, all further communication is done in P2P fashion.

Skype P2P network organizes all the nodes into two layers: Ordinary Nodes and Super Nodes as shown in Figure 2. Ordinary node is the Skype application that can be used to place calls and send messages. A Super

Node is a ordinary host's end point in Skype network. Super Nodes also function as ordinary nodes and are selected from the set of ordinary nodes only. Any ordinary node with public IP address and good computational resources and connectivity are eligible to become Super Node.

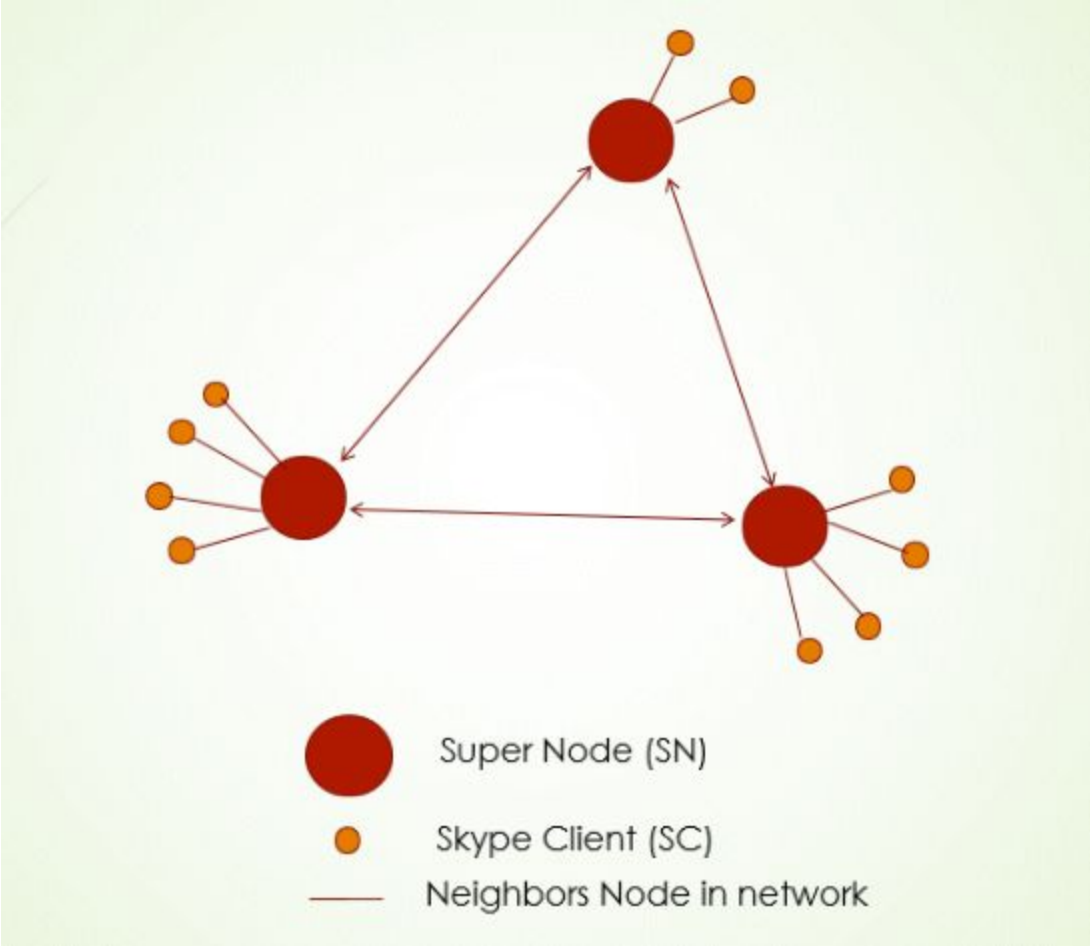

Figure 2: Network of Skype Nodes

Skype maintains an overlay P2P network among the Super Nodes and uses them for message relaying and handling metadata such as user profiles and presence information. Skype stores its user information in decentralized fashion among P2P nodes. In this way, Skype doesn't own most of its own architecture allowing it to easily scale to very large sizes while minimizing the need of large scale architecture. Skype only provides relatively small set of "bootstrap" Super Nodes hardcoded in Skype

executable, gateway to public switched networks and login servers for authentication. Bootstrap Super Nodes are contacted only when skype is newly installed and login servers need not be contacted at each login as clients may use Super Nodes to relay their login information.

Since Skype network is an overlay network, each Skype Client needs to maintain a list of reachable Super Nodes. This list is called Host Cache (HC) and it contains IP Address and port number of reachable Super Nodes. HC is stored in a XML file.

This work also describes the key components of the Skype architecture as follows:

1. Ports: Skype uses TCP listening port at 80 or 443 which are used for HTTP and HTTP-over-TLS requests.
2. Host Cache: HC is the list of Super Node IP address and port pairs that SC refreshes regularly. A SC for Linux stores HC file as XML file 'shared.xml' at $(HOMEDIR)/.Skype.
3. Codecs: Skype uses variety of Codecs like iLBC, iSAC, iPCM and SVOPC.
4. Buddy List: In Linux, skype stores buddy at $(HOMEDIR)/.Skype/<skype user id> and it is stored in unencrypted form.
5. Encryption: Skype uses AES (Advanced Encryption Standard) which is 256 bit encryption having total of 1.1×10^{77} possible keys to encrypt the data in each Skype call or instant message.
6. NAT and firewall: SC uses a variation of STUN to determine the type of NAT and firewall it is behind. This information is also stored in shared.xml file.

This work also discussed different functions of Skype like startup, login, user search, call establishment and tear down, media transfer, and presence messages by performing different experiments on Skype.

In [7], authors used both active and passive measurements to analyze the characteristics of traffic streams generated by voice and video communications, and the signaling traffic generated by Skype. They characterized the traffic generated by voice and video calls, by observing the time evolution in terms of bit rate, inter-packet gap and packet size. Bit rate(B) is the amount of bits generated at application layer in interval of 1 second. Inter-Packet gap(I) is time between two consecutive packets in the same flow. Payload Length(L) is the number of bytes transported in UDP or TCP payload. The authors also discussed different behaviour of traffic source based on adopted transfer layer protocol and changing network conditions.

In [8], authors deeply analyzed the Skype login and elucidated various functions, traffic characteristics and login process of Skype. The authors elaborated various secrets of Skype login in this work. According to this work, during the time of login, Skype client performs a fixed sequence of instructions which are as follows:

1. SC authenticates username and password with login server.
2. It establishes communication with 4 Super Nodes that are randomly selected from HC.
3. It establishes communication with MSN Echo server to determine the type of NAT and firewall it is behind.
4. It advertises its presence to other peers and buddies.
5. It checks for latest Skype version.
6. It keeps active communication with user feedback information server to collect user suggestions.
7. SC connects to facebook server to share friends on facebook.
8. If SC wants to communicate to another SC, it does this after connecting to Super Node.

4

Proposed Approach

In this chapter, we propose a signature based classification of Skype traffic which is based on the repetitive signalling that Skype Client does during idle period. During the idle phase, Skype repeats the fixed set of instructions in certain time intervals. These include:

1) Querying the DNS server for IP address of s.gateway.live.com (MSN Echo Server).
2) Skype client makes a communication with MSN Echo Server using TCP on port 443 (TLSv1).
3) Skype client communicates with Super Node using TCP, essentially performing certain signalling.
4) A stream of UDP packets is then exchanged between Skype client and other IP addresses.

We believe that some of the signalling communication between Skype client and Super Node should remain invariant in any instance of communication in step 3 of above procedure.

Thus, the lengths, flow direction and sequence of payloads corresponding to those signalling communication must also be constant in any case. If the Skype Client is behind the NAT, all those communication must be relayed through Super Node. Thus, while observing TCP communication between Skype Client and Super Node in Skype traces of IDLE period, we found that some of the *payload length sequences* were occurring frequently in a single flow. The payload length sequence of length N is defined as [P_1, P_2 … P_N] where P_i is the payload length in bytes of i^{th} packet in the sequence and all the packets from 1 to N must belong to a single flow in sequential order. Moreover, if the sign of P_i is positive then i^{th} packet is being sent and if P_i is negative then i^{th} packet is being received.

For example, sequence of [3, 10, -4, -2], [4, 104, -4, -12, 4, 272] and [-2, 3, 15, -13] payload lengths were occurring frequently in traces. The flow diagram of sequence [3, 10, -4, -2] is shown in Figure 3. Screenshot for pattern [3, 10, -4, -2] in wireshark is shown in Figure 4.

To confirm this observation, we conducted an experiment on different types of traces we collected using Wireshark [9] and Skype v4.3 on Linux. For every test sequence of payload length, we found the number of occurrences of it in each Skype trace. The results are mentioned in Table 2.

4.1 Identifying the Voice Traffic

For voice traffic, we found some new payload length sequences specific to voice data signalling viz. [3, -16, 4], [10, -3, -15], [-3, -7, 18], [-3, -15, 4] and [13, -18, 15]. So we conducted the similar experiment to observe total number of occurrences of these sequences in voice traffic. The results are mentioned in Table 2.

Dataset	Size(in MB)	P1	P2	P3	P4	P5	P6	P7	P8
IDLE	45	426	76	93	18	0	0	1	82

VOICE	450	72	17	7	1091	119	36	47	290

Pattern Name	**Payload Length Sequence**
P1	3, 10, -4, -2
P2	4, 104, -4, -12, 4, 272
P3	-2, 3, 15, -13
P4	-3, -16, 4
P5	10, -3, -15
P6	-3, -7, 18
P7	13, -18, -15
P8	-3, -15, 4

Table 2: Occurrences of Patterns

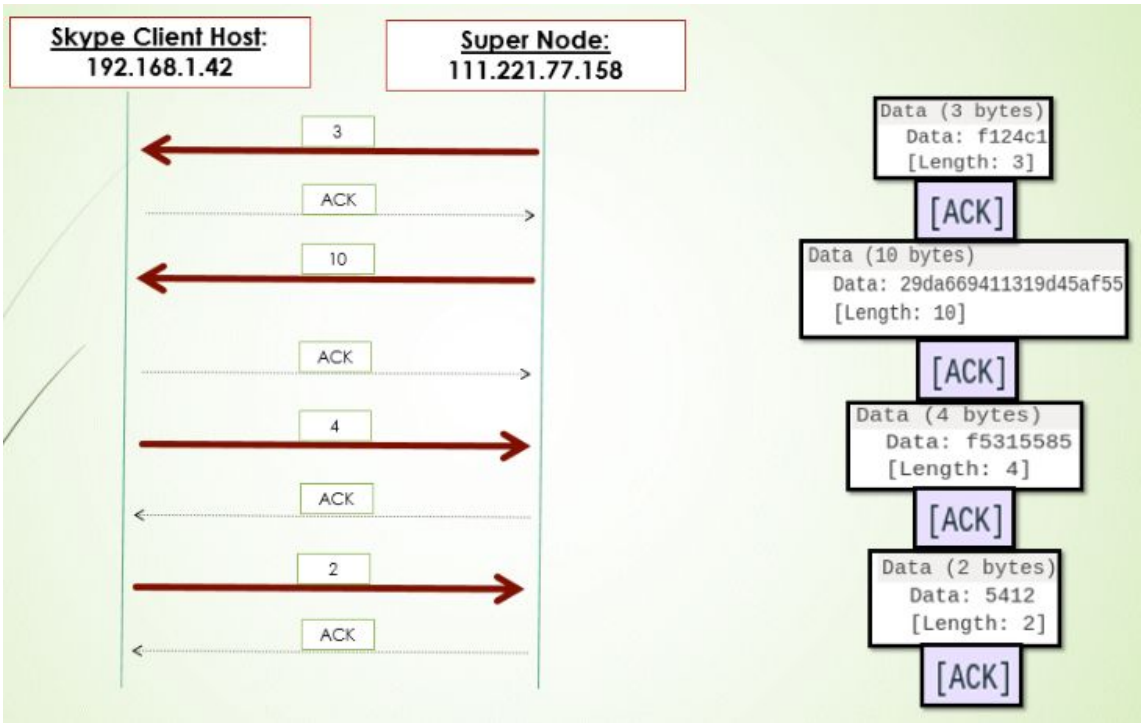

Figure 3: Flow for Pattern 3, 10, -4, -2.
(Negative sign for packet being sent and positive sign for packet being received)

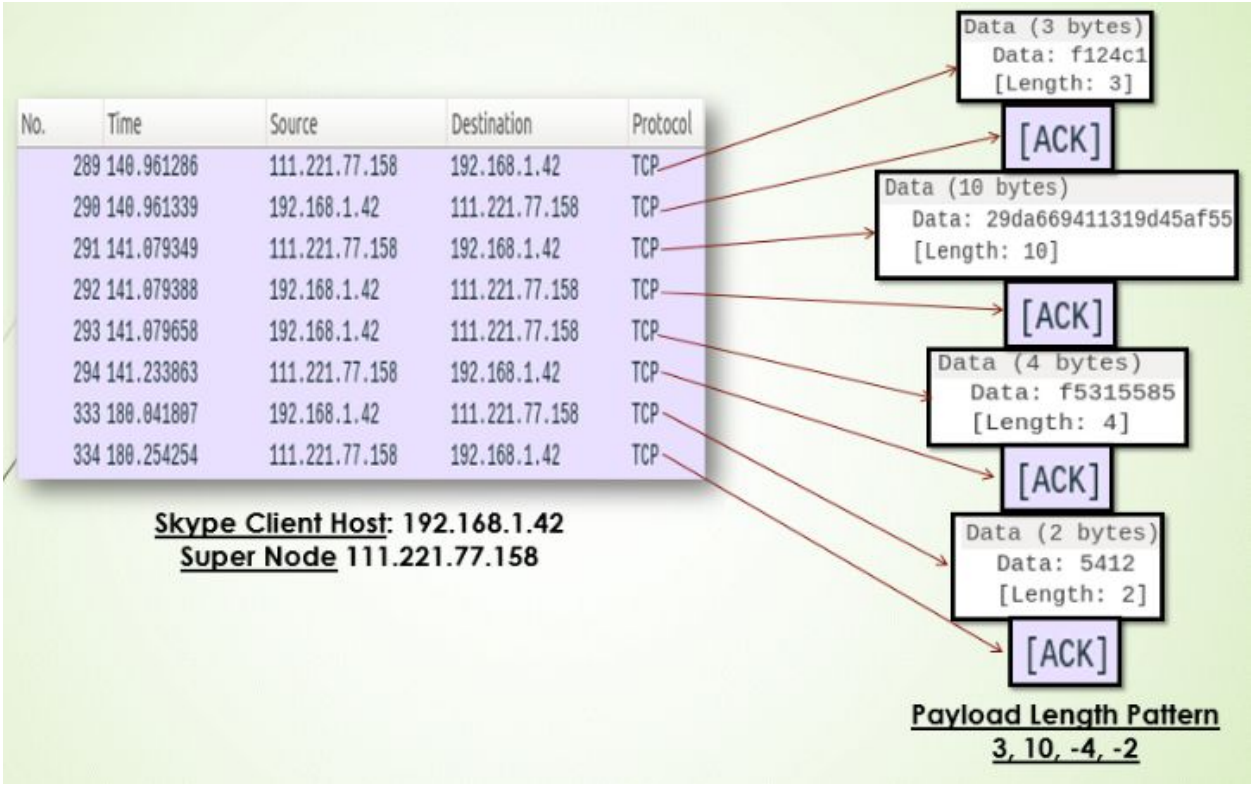

Figure 4: Pattern 3, 10, -4, -2 as seen in Skype Trace by Wireshark

One such instance of an occurrence of sequence [3, 10, -4, -2] as seen in wireshark application is shown in Figure 3 and 4. It shows communication between skype client (192.168.1.42 here) and Super Node(111.221.77.158 here).

Considering the observations presented above, we came up with a classifier that can decide whether the traffic under observation is generated by Skype host depending on the presence of expected sequences of payload. Currently our classifier labels a traffic as *IDLE* Skype traffic if it contains at least one of [3, 10, -4, -2], [4, 104, -4, -12, 4, 272] and [-2, 3, 15, -13] as a sequence of payload lengths belonging to a single flow. Moreover this classifier can be used to identify the Skype Client(SC) and corresponding Super Node(SN) as this sequence belongs to a single flow between SC and SN. Moreover, for identification of *VOICE* traffic generated

by Skype, our classifier classifies *VOICE* traffic if it contains at least one of [3, -16, 4], [10, -3, -15], [3, 7, -18], [-3, -15, 4] and [13, -18, 15].

5

Design and Implementation

We implemented our payload length based classification algorithm in java. We used JnetPcap[10] library to process our collected traffic and generate datasets from it.

5.1 Skype Classifier

Considering the observations of patterns of payload length, we came up with a classifier that can decide whether the traffic under observation is generated by Skype host depending on the presence of expected sequences of payload. Currently our classifier labels a traffic as Skype traffic if it contains at least one of the discussed patterns as a sequence of payload lengths belonging to a single flow. The pattern may not have contiguous occurrence in the dataset and may have extraneous packets in between. We have allowed at most **40** extraneous packets between the first and last packet belonging to a particular occurrence.

```
func classify( dataset )
        for each Expected Pattern : P
                if( detect(dataset, P) = Found)
                        return Skype
                endIf
        endfor
        return Non Skype
endfunc
```

Figure 5: Algorithm to Classify a Dataset

```
func detect( dataset, pattern)
        for  occurrence <P1, P2 ,.., PN> of pattern in dataset
                if( number of extraneous packets between PN and P1 <= 40
                   and all packets Pi's belong to the same flow
                   and direction of Pi and Pj differ iff sign of patterni and patternj differ)
                        return Found
                endIf
        endfor
        return Not Found
endfunc
```

Figure 6. Algorithm to Detect a Pattern in Dataset

Our classification algorithm is described in Figure 5 and 6. In order to classify our dataset, we search for our expected payload length patterns. For every test sequence of payload length S_1, S_2, .., S_N, we found an occurrence of it in each dataset satisfying the following conditions.

- All the packets belonging to a particular occurrence must belong to a single flow.
- The packets corresponding to negative payload must be in the opposite direction of packets corresponding to positive payload.
- Since two or more communications can generate interleaved sequences of packets, these occurrences may not contain contiguous packets one after another. But they must appear in particular order.

- Every occurrence P_1, P_2,.., P_N of the pattern is considered valid if number of extraneous packets in between P_1 and P_N is less than a threshold (40 in our experiment)

If the sequence satisfies all the given conditions, we classify the dataset as Skype traffic.

5.2 Implementation and Code Details

While processing the dataset, we only take TCP packets into consideration and remove all other packets from inspection.

```
public void payloadLenth_Patterns(int maximumGap, String packetFileName,
        boolean checkFlowDirection, int... pattern) {
```

Figure 7: Payload Pattern Finder

```
Pattern of Payload to search = 4, 104, -4, -12, 4, 272,
File name=  packets/idle 28-9-16 full nighht (us) data 3.pcapng
The pattern was found at following intervals
5061, 15650, 16113, 16391, 16555, 16886, 17143, 17234, 17572, 17675, 17871, 18185, 19047
Total Number of distinct Occurrences = 13
```

Figure 8: Sample Output of Pattern Finding Algorithm

To observe how many times and where our expected payload patterns are present in captured Pcap files, we implemented our pattern finding algorithm (similar to Algorithm in Figure 6). The method definition is described in Figure 7. It takes the maximum allowed extraneous packets as *maximumGap*, the captured Pcap file name and the expected pattern to search as *pattern*.

One sample output of this implementation is shown in Figure 8 showing the number and place of occurrences of pattern in the Pcap file.

```
boolean onPayloadLengthPattern(int threshold, List<PcapPacket> packets,
        int... pattern) {
```

Figure 9: Skype Classifier

The skype classifier as defined in Figure 9 is the heart of our algorithm which takes the dataset as a List of PcapPackets and classifies as Skype and Non Skype based on the occurrence of payload pattern *pattern* in the dataset.

```
*********************************
Classifying packets/Idle_24-9-2016 (small 50 min) data 5.pcapng
Total classified = 4
Total datasets = 4
*********************************
Classifying packets/idle 27-9-16 fullnight (us) data1.pcapng
Total classified = 47
Total datasets = 49
*********************************
Classifying packets/idle 27-9-2016 full night (teekaram) data 2.pcapng
Total classified = 66
Total datasets = 68
*********************************
Classifying packets/idle 28-9-16 full nighht (us) data 3.pcapng
Total classified = 53
Total datasets = 74
```

Figure 10: Sample Classification Results

Figure 10 describes a sample output of our classification algorithm on some of Idle Skype Trace. It shows how many datasets one file was divided into and how many datasets were correctly classified as Skype traffic by the algorithm.

```
Flow = 192.168.43.14:50164 -> 191.232.139.13:443 Tcp fw/rev/tot pkts=[15/12/27]
Forward: 0, 0, 62, 0, 0, 0, 0, 0, 267, 59, 0, 229, 37, 0, 0,
Reverse: 0, 1348, 112, 1348, 112, 408, 0, 59, 154, 0, 0, 0,

Flow = 192.168.43.14:50165 -> 191.232.139.13:443 Tcp fw/rev/tot pkts=[15/12/27]
Forward: 0, 0, 62, 0, 0, 0, 0, 0, 267, 59, 0, 229, 37, 0, 0,
Reverse: 0, 112, 1348, 1348, 112, 408, 0, 59, 154, 0, 0, 0,

Flow = 91.190.216.66:12350 -> 192.168.43.14:56137 Tcp fw/rev/tot pkts=[16/15/31]
Forward: 0, 0, 0, 5, 0, 0, 5, 0, 0, 5, 0, 0, 5, 0, 0, 5,
Reverse: 0, 5, 0, 0, 5, 0, 0, 5, 0, 0, 5, 0, 0, 5, 0,
```

Figure 11: Payload Lengths in Each Flow

In order to observe more payload length in the collected traffic, we divided all the packets in different flows and observed payload lengths in each flow separately. Sample output of the implemented program is shown in Figure 11. It shows each flow as tuple of <IP:port, IP:port> and the corresponding payload lengths of exchanged packets in each direction.

6

Experiments and Results

6.1 Data Collection

In order to verify and evaluate our payload based classification algorithm, we collected different types of Skype traces. All traffic was captured on Linux(Ubuntu) using wireshark[9]. A total of 45 hours of Skype traffic was captured during its IDLE period. It consisted of 45MB of Skype trace with about 100,000 TCP packets. It is comprised of various skype sessions. We also collected skype trace during login period to study the communication of Skype client during login session. We also collected Skype traffic during voice calls. A total of 15 hours of voice traffic was captured comprising of 450 MB data and 150,000 TCP packets. The voice traffic consisted 2 types of voice:

- Silence throughout the call with minor noises
- Voice transfer throughout the call with a song being played on repeat.

In order to verify that the patterns of payload are not specific to the user who has logged in as the SC, we collected traffic for different users as well. In order to verify that these patterns are not specific to TLS protocol, we collected traffic from various sites like Bank website/ transaction[11] and google hangout[12] who majorly uses TLS. Non-Skype traffic included traffic from sites like Facebook[13], Youtube[14], Google Search[15] and random browsing on the internet.

6.2 Dataset Generation

```
Set datasets to empty
For each captured pcap file: PcapF
        set currentDataset empty
        For each TCP packet in PcapF: P
                if( (earliest arrival time in currentDataset) -  P.timestamp <= WindowSize)
                        Add P to currentDataset
                else
                        Add currentDataset to datasets
                        Set currentDataset empty
                endif
        endfor
endfor
```

Figure 12: Algorithm for Dataset Generation

In order to generate multiple datasets from the captured traffic, we divided the captured packets into groups of datasets. The generated datasets did not have same number of packets but all of them contained packets for the same interval of time(referred as *WindowSize* in Figure 12). Since we needed to evaluate our algorithm on the time it takes to achieve high precision, we generated these datasets multiple times from the captured traffic but with different time slices like 2, 4, 5, 7, 10, 15 minutes.

6.3 Performance Parameters

The performance is based on the following factors:

- **Recall**[4] is the percentage of samples that belong to that class that were correctly identified as such.
- **Precision**[4] is the percentage of samples correctly classified as belonging to a particular class out of the total number classified as belonging to that class.

If we denote true-positive percentage by TP, true-negative percentage by TN, false-positive percentage by FP, and false-negative percentage by FN, then we define recall and precision as follows:

- $\text{Precision} = \frac{TP}{TP + FP}$

- $\text{Recall} = \frac{TP}{TP + FN}$

6.4 Results and Comparison

We executed our algorithm described in Figure 6 on all the datasets generated by algorithm described in Figure 12 for various time slices of 2, 4, 5, 7, 10, 15 minutes. As the amount of data and number of packets exchanged in IDLE period is quite less (but still substantial), it takes around 10 minutes to achieve a recall of greater than 97%. The results of classification for IDLE traffic is shown in Table 4. On the other hand, there is huge traffic during VOICE call and the expected patterns occur quite frequently thereby reducing time to get a recall greater than 99% to just 3-4 minutes. The results of classification for VOICE traffic is shown in Table 3. The Recall for IDLE and VOICE traffic is plotted in Figure 13 with window duration(in minutes) being on X axis.

TimeSlot(Minute)	TP	FP	FN	Recall%	Precision%
2	268	0	14	95.35	100
4	145	0	1	99.31	100
5	119	0	1	99.16	100
7	84	0	0	100	100
10	60	0	0	100	100
15	40	0	0	100	100

Table 3: Results for VOICE Traffic Classification

TimeSlot(Minute)	TP	FP	FN	Recall%	Precision%
2	269	0	380	38.76	100
4	322	0	118	73.18	100
5	340	0	91	78.8	100
7	303	0	27	91.81	100
10	235	0	7	97.10	100
15	159	0	2	98.75	100

Table 4: Results for IDLE Traffic Classification

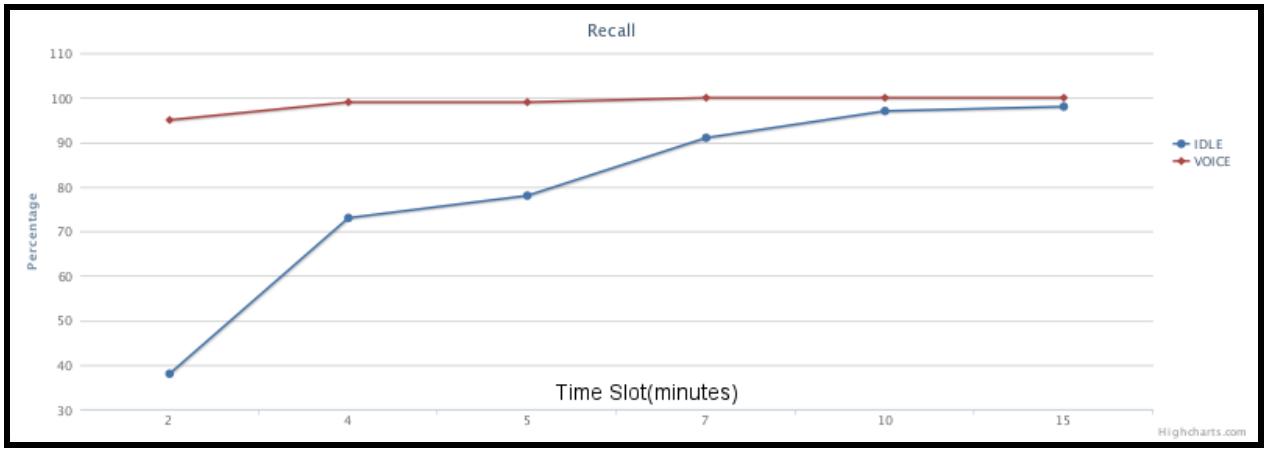

Figure 13: Comparison of Recall for IDLE and VOICE Traffic

This approach noticeably gives 100% precision denoting no false positive. It also verifies that the appearance of such patterns is neither due to specific protocols like TLS nor due to specific user activity in Skype client.

7

Conclusion and Future Work

In this report, we presented traffic classification method using unique sequence signatures based on payload lengths of TCP packets to implement host-level identification system for Skype traffic classification. We have used various sequences to build our classifier with recall and precision better than **97%** by observing about **15** minutes of Skype traffic. Higher recall of greater than **99%** is achieved in voice traffic in **4** minutes. We can also identify the type of session running on Skype Client (Idle or Voice) using our classification method. The classification method we proposed is lightweight and doesn't require access to payload.

We conjecture that such payload length sequences can be found for video traffic of Skype as well. Also a number of such payload length sequences can be found and used in classifier for better Recall. We believe that our work can also be extended for Skype's video traffic classification and more applications with similar architecture as Skype by finding their specific unique sequence signatures.

References

[1] S.Valenti, D.Rossi, A.Dainotti, A. Pescape, A. Finamore and M.Mellia, “Reviewing Traffic Classification,” in Data Traffic Monitoring and Analysis, pp. 123–147, Springer, 2013.

[2] X.Chen, D.Wang, Z.Yuan, C. Du and Y. Xue, “Skytracer: Towards Fine-grained Identification for Skype Traffic via Sequence Signatures,” International Conference on Computing, Networking and Communications(ICNC), 2014.

[3] M.Meo, D.Rossi, D.Bonfiglio, M.Mellia and P.Tofanelli, “Revealing Skype Traffic: When Randomness Plays with you,” in Special Interests Group on Data Communications(SIGCOMM), 2007.

[4] A. Heyde, P. Branch and G. Armitage, “Rapid Identification of Skype Traffic Flows,” Network and Operating Systems Support for Digital Audio and Video(NOSSDAV), 2009.

[5] R. Alshammari and A. Heywood, “Machine Learning Based Encrypted Traffic Classification: Identifying SSH and Skype,” Computational Intelligence for Security and Defense Applications(CISDA), 2009.

[6] S. A. Baset and H. Schulzrinne, “An Analysis of the Skype Peer-to- Peer Internet Telephony Protocol,” arXiv preprint cs/0412017, 2004.

[7] D.Bonfiglio and M.Dario, “Detailed Analysis of Skype Traffic,” IEEE Transactions on Multimedia, vol. 11, issue 1, pp. 117–127, 2009.

[8] Y. Ren, P. Qi, C. Du and Y. Xue, “The Secrets of Skype Login,” International Workshop on Cloud Computing and Information Security(CCIS), 2013.

[9] “https://www.wireshark.org.”

[10] “http://jnetpcap.com/.”

[11] “https://www.onlinesbi.com/.”

[12] “https://hangouts.google.com/.”

[13] “https://www.facebook.com/”

[14] “https://www.youtube.com/”

[15] “https://www.google.co.in”