INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2012; **25**:386–403
Published online 15 February 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/dac.1247

Skype-Hunter: A real-time system for the detection and classification of Skype traffic

Davide Adami, Christian Callegari*, †, Stefano Giordano, Michele Pagano and Teresa Pepe

Department of Information Engineering, University of Pisa, Pisa, Italy

SUMMARY

In the previous years, Skype has gained more and more popularity, since it is seen as the best VoIP software with good quality of sound, ease of use and one that works everywhere and with every OS. Because of its great diffusion, both the operators and the users are, for different reasons, interested in detecting Skype traffic.

In this paper we propose a real-time algorithm (named *Skype-Hunter*) to detect and classify Skype traffic. In more detail, this novel method, by means of both signature-based and statistical procedures, is able to correctly reveal and classify the signaling traffic as well as the data traffic (calls and file transfers). To assess the effectiveness of the algorithm, experimental tests have been performed with several traffic data sets, collected in different network scenarios. Our system outperforms the 'classical' statistical traffic classifiers as well as the state-of-the-art *ad hoc* Skype classifier. Copyright © 2011 John Wiley & Sons, Ltd.

Received 29 April 2010; Revised 29 September 2010; Accepted 25 December 2010

KEY WORDS: Skype; traffic classification; traffic analysis; real time classification

1. INTRODUCTION

In the recent years, the popularity of VoIP-telephony has progressively grown and the majority of network operators has started offering VoIP-based phone services. Skype [1] has rapidly become the most well-known example of this consolidated phenomenon: originally developed by the entrepreneurs who created the pioneering Web application Kazaa, Skype ended 2008 with 405 million user accounts (more than 42 million of real users [2]), a 47% increase from 2007 [3]. According to the TeleGeography Research [2], in 2008 Skype users spent 33 billion minutes talking to people in other countries, representing 8% of all international voice traffic. Moreover, Skype usage hit an all-time peak on 30 March 2009, when more than 17 million users were online at the same time [4].

Unlike other VoIP applications, which generally rely on the client–server architectural model, Skype operates on a Peer-to-Peer (P2P) overlay network. The communication among Skype users is established according to the end-to-end paradigm except when a relay node is used for symmetric NAT and firewall traversal. Skype offers several free services: voice and video communication, file transfer, chat, buddy list, and SkypeIn/SkypeOut to direct call towards the PSTN.

As a consequence of its success, Skype has attracted the attention of both network operators and researchers. The first ones require that efficient techniques for the identification of Skype traffic are

*Correspondence to: Christian Callegari, Department of Information Engineering, University of Pisa, Pisa, Italy.
†E-mail: christian.callegari@iet.unipi.it

developed so as to evaluate its impact on network performance, design effective security policies, and enforce traffic differentiation strategies.

From a research perspective, the identification and characterization of Skype traffic are hot topics. Indeed, Skype is based on proprietary protocols, which make extensive use of cryptography, obfuscation, and anti reverse-engineering procedures [5]. Moreover, Skype adopts *ad hoc* techniques to evade NATs and firewalls. As a result, classical statistical traffic classifiers are not suitable to correctly classify Skype traffic (as demonstrated in Section 5). Thus, innovative classification techniques should be introduced in order to identify Skype flows: this requires the definition of new algorithms, able to take into account not only the statistical properties of the traffic, but also all the Skype features that have been discovered by reverse-engineering procedures.

This paper proposes a joint signature-based and statistical approach that outperforms state-of-the-art methodologies in the identification of Skype traffic. The paper extends the preliminary work and results presented by the authors in [6].

In more detail, our primary contribution is the definition of a new algorithm (named *Skype-Hunter*) performing real-time classification, distinguishing among voice and video calls, data exchanges, and underlying control traffic. The algorithm exploits a detailed analysis of the signaling traffic, exchanged to establish a Skype call and to maintain updated the information concerning Skype peers. The description of the above-mentioned signaling phase is a further relevant contribution of this work.

The paper is organized as follows. Section 2 presents an overview of the state of the art concerning Skype protocols and classification. In order to clarify the fundamentals of our proposal, Section 3 analyzes the main characteristics of Skype traffic. Then Section 4 describes the proposed algorithm, while Section 5 presents the experimental results and the comparison with other approaches. Finally, Section 6 concludes the paper with some final remarks.

2. RELATED WORKS

At the time of writing, the most complete work concerning Skype traffic classification is [7], where the authors present a methodology working in real time. In more detail, they propose a framework based on two different and complementary techniques, for revealing Skype traffic from a traffic aggregate. The first approach, based on Pearson's Chi Square test, is used to detect Skype's fingerprints from the packet framing structure, but is agnostic to VoIP-related traffic characteristics. The second approach, instead, relies on a stochastic characterization of Skype in terms of packet arrival rate and packet length, which are used as features of a decision process based on Naive Bayesian Classifiers. As shown in the last section of the paper, our algorithm outperforms this work.

Apart from that, there are several other papers, where algorithms for detecting Skype traffic are presented, but none of these is able to correctly detect all the different kinds of Skype traffic (e.g. most of them do not take into account the TCP case). For example, in [8], the authors aim at identifying relayed, rather than direct traffic, and Skype is chosen as a case study to validate their approach, while in [9] the authors only focus on the UDP traffic. Finally, in [10] the authors just focus on a very particular case, detecting Skype by exploiting information derived from a 3G network.

Other works focus on understanding how the Skype protocol works. In [5] the authors provide an overview on Skype design and functionalities, exploring many operational aspects under different network scenarios. Users with public IP addresses as well as users behind a port-restricted NAT or UDP-restricted firewall are taken into account. In [11] the authors aim at studying Skype operations by means of traffic measurements over a five-month period. The authors analyze the user behavior for relayed, rather than direct, sessions. Results pertain the population of online clients and their usage pattern, the number of SNs and bandwidth usage.

Finally, in [12] the authors investigate Skype signaling mechanisms by means of passive measurements, providing insights into Skype signaling mechanisms and analyzing the cost and complexity of managing Skype P2P overlay network.

None of these papers present neither a complete description of the Skype protocol, nor an algorithm able to classify the Skype traffic. Hence, the contribution of this paper is two fold: on one side it completes the description of the Skype architecture, as provided in [5, 11], by adding

DOI: 10.1002/dac

more details both at the flow and at the packet levels (e.g. no previous work has addressed a detailed study of the packets payload). On the other hand the paper proposes a Skype detection algorithm, *Skype-Hunter*, able to detect and classify all the types of traffic generated by Skype.

3. SKYPE TRAFFIC

Before detailing the algorithm implemented to detect Skype traffic, we present a brief analysis of Skype features, only focusing on those aspects that are essential for the detection phase (for a more complete description of Skype architecture, please refer to [7]).

It is important to highlight that the Skype protocol has never significantly changed since its first versions, thus the following description, as well as the presented detection algorithm, is suitable for all the current versions of the protocol.

Skype is a P2P network composed of several nodes call Skype Clients (SCs). The SCs can start and receive a call, send instantaneous messages, and transfer files. Since it is a hybrid P2P network, there are some specialized nodes, called Super Nodes (SNs), which provide additionally functionalities. In more detail, they are responsible for the connection and disconnection of the SCs to Skype network. Each SC can be selected as an SN if it has a public IP address and 'enough' bandwidth. In that case, it must be able to reply to the requests from the SCs and other SNs and forward Skype requests towards the correct destination.

Moreover, Skype architecture presents a centralized element, named login server. This server is responsible for the authentication phase, since each SC must authenticate itself to the login server before accessing Skype network.

It is important to highlight that the behavior of the application depends on the characteristics of the network. For example, the presence of firewalls or NATs has an impact on the choice of the transport protocol to be used: if there are no restrictions on the use of UDP, the application usually sends the data traffic over UDP and the signaling traffic over TCP. On the contrary, if UDP cannot be used for some reasons (e.g. a firewall prevents the users from using such protocol), Skype sends both the signaling and the data traffic over TCP. Before deciding not to use UDP, Skype 'tries' to get through the NAT/firewall, by means of a technique called *UDP hole punching* [13], which is a variant of the well-known STUN [14] and TURN [15] protocols.

Let us analyze in more detail the procedure to establish a call, in the following different scenarios:

- no restrictions on the transport protocols:
  - *signaling traffic*: a direct TCP connection between the SCs is used
  - *data traffic*: a direct UDP flow between the SCs is used
- presence of a NAT:
  - *signaling traffic*: a TCP connection through an SN, acting as a relay node, is used
  - *data traffic*: a direct UDP flow between the SCs is used
- presence of a firewall which does not allow the use of UDP:
  - *signaling traffic*: a TCP connection through an SN, acting as a relay node, is used
  - *data traffic*: a TCP connection through an SN, acting as a relay node, is used.

In case the data traffic is not directly forwarded from the caller to the callee, we can have several scenarios. Indeed each user can 'decide' to use a relay node (or a network of relay nodes) independently from the other user. Thus, there can be both symmetric calls, where each user sends its data traffic through a relay node (see Figure 1(c)), and asymmetric calls where one user sends its traffic directly to the other, which instead uses a relay node to forward its own traffic (see Figure 1(b)).

Finally, we highlight that in case UDP is used, the messages present an unencrypted header, named Start of Message (SoM), which is used for realizing a reliable communication. This header contains, among others, the following fields:

- *session identifier* (2 bytes) identifies a given Skype session and
- *function field* (1 byte) used to distinguish the different message exchanges.

DOI: 10.1002/dac

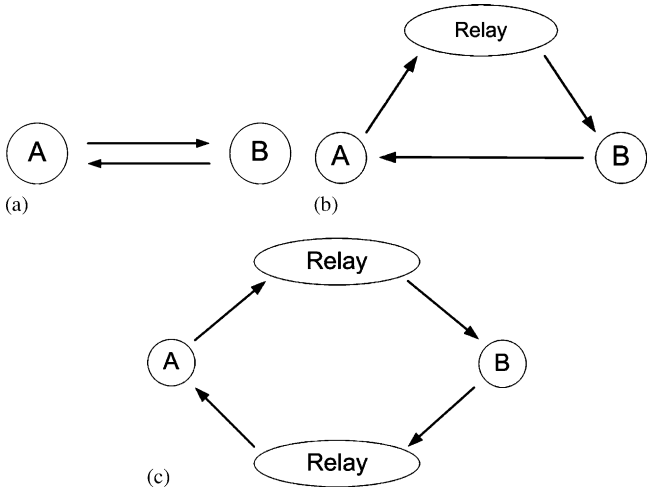

Figure 1. Call scenarios: (a) standard call; (b) call with one relay; and (c) call with two relays.

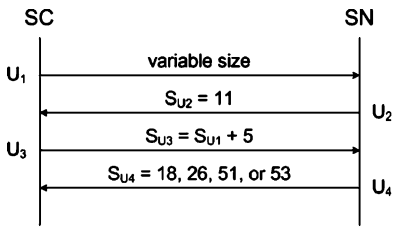

Figure 2. Long Skype UDP probe.

The structure of the SoM is different in case of SkypeOut calls. In that case we do not observe the previously described fields, but in most packets there is a field (first 4 bytes) that always assumes that the same value in a given call and varies from call to call. Because of that, we assume here that the field contains the Call Identifier (CID).

On the contrary, if TCP is used the message is completely encrypted.

In the following subsections, we detail some important features of the signaling phase, focusing on the description of the standard Skype behavior (e.g. the UDP Probe will not take place if a firewall is blocking UDP traffic).

3.1. Skype UDP Ping

The Skype UDP Ping is a message exchange, periodically carried out by all the SCs, which consists of two *keep-alive* messages and is used to advertise the presence of a node in the Skype network. These messages are characterized by the function field of the message equal to 0x02. Some UDP flows only consist of the exchange of such messages.

3.2. Skype UDP Probe

The Skype UDP Probe is a message exchange performed when Skype application is launched, to discover the SNs and the network characteristics (e.g. presence of NATs, firewalls, etc.). The Skype UDP Probe consists of four UDP messages (Long Skype UDP Probe): two request messages from the SC to the SN (U1 and U3) and two reply messages from the SN to the SC (U2 and U4). Figure 2 depicts the message exchange (here and in the following, s_x is the size of the payload of packet x).

DOI: 10.1002/dac

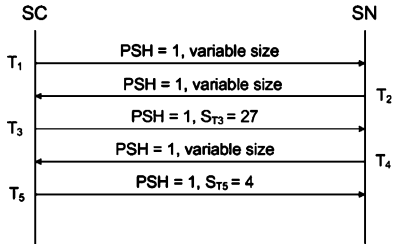

Figure 3. Skype TCP Handshake.

After the Skype UDP Probe has been executed, the SC performs the algorithm for the SN selection, which consists of the following steps:

if $s_{U4} = 18$ bytes **then**
  the SC opens a TCP connection (on the same port number used for the Skype UDP Probe) and the contacted node becomes its SN
else
  if $s_{U4} \in \{26, 51, 53\}$ bytes **then**
    the contacted node will not be taken into account any longer
    # it is likely that these messages can be interpreted as SN 'negative' replies
  end if
end if

A modified version of this message exchange, only involving U1 and U4 packets (Short Skype UDP Probe), is repeated until an SN is selected.

Moreover, the SC periodically repeats the Short Skype UDP Probe, to be sure to be always connected to an available SN.

3.3. Skype TCP Handshake

Once the SN has been selected, the SC has to establish a TCP connection with the SN, so as to be able to access Skype network. This phase is composed of the following steps:

- a TCP connection is opened towards the first node that has positively answered to the Skype UDP Probe (over the same port number used for the reception of the Skype UDP Probe messages)
- in case the TCP connection fails, a new connection is established with another SN that previously acknowledged a Skype UDP Probe
- once the connection has been established, a message exchange, named Skype TCP Handshake, is started.

It is worth noting that, although the whole payload is encrypted, these packets are characterized by the TCP PSH flag set and, some of them, by a fixed size packet payload. Figure 3 displays the message exchange.

From the figure, we note that five messages are exchanged: T1, T2, and T4 are variable size messages, while messages T3 and T5 are 27 and 4 bytes long, respectively.

Since this phase is based on the opening of a TCP connection over a randomly selected port, it is obvious that this packet exchange can only be realized if there are no restrictions on the usable ports. On the contrary, if the SC resides behind a NAT or a firewall that blocks the outgoing connections, the connection is established over TCP port 80 (Skype HTTP Handshake) or 443 (Skype HTTPS Handshake), which are usually left open by the firewalls, since they are respectively used by the HTTP and HTTPS protocols. Skype uses Transport Layer Security (TLS) protocol over port 443 and a proprietary protocol over port 80. The message exchange over both ports is depicted in Figure 4.

DOI: 10.1002/dac

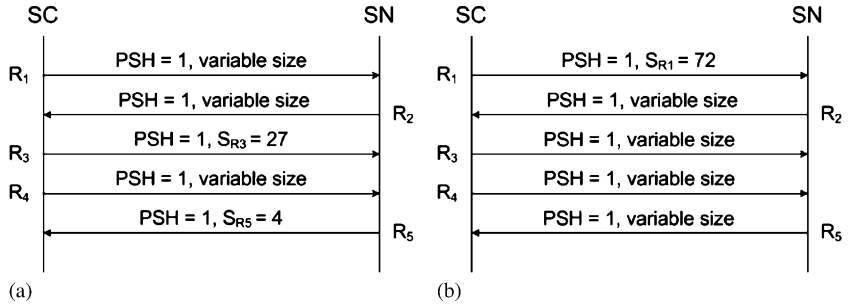

Figure 4. Skype HTTP/HTTPS Handshake: (a) HTTP and (b) HTTPS.

Let us analyze the two cases separately:

- connection over port number 443
  - the payload of the first packet is 72 bytes long and the first 56 bytes have a fixed value (named A_1 in the following), corresponding to the Client Hello of the TLS 1.0 protocol:

$$A_1 = 0x80\ 0x46\ 0x01\ 0x03\ 0x01\ 0x00\ 0x2d\ 0x00$$
$$0x00\ 0x00\ 0x10\ 0x00\ 0x00\ 0x05\ 0x00\ 0x00$$
$$0x04\ 0x00\ 0x00\ 0x0a\ 0x00\ 0x00\ 0x09\ 0x00$$
$$0x00\ 0x64\ 0x00\ 0x00\ 0x62\ 0x00\ 0x00\ 0x08$$
$$0x00\ 0x00\ 0x03\ 0x00\ 0x00\ 0x06\ 0x01\ 0x00$$
$$0x80\ 0x07\ 0x00\ 0xc0\ 0x03\ 0x00\ 0x80\ 0x06$$
$$0x00\ 0x40\ 0x02\ 0x00\ 0x80\ 0x04\ 0x00\ 0x80$$

  - the size of R2 is variable, but the first 79 bytes have a fixed value (named A_2 in the following), corresponding to the TLS Server Hello message:

$$A_2 = 0x16\ 0x03\ 0x01\ 0x00\ 0x4a\ 0x02\ 0x00\ 0x00$$
$$0x46\ 0x03\ 0x01\ 0x40\ 0x1b\ 0xe4\ 0x86\ 0x02$$
$$0xad\ 0xe0\ 0x29\ 0xe1\ 0x77\ 0x74\ 0xe5\ 0x44$$
$$0xb9\ 0xc9\ 0x9c\ 0xb4\ 0x31\ 0x31\ 0x5e\ 0x02$$
$$0xdd\ 0x77\ 0x9d\ 0x15\ 0x4a\ 0x96\ 0x09\ 0xba$$
$$0x5d\ 0xa8\ 0x70\ 0x20\ 0x1c\ 0xa0\ 0xe4\ 0xf6$$
$$0x4c\ 0x63\ 0x51\ 0xae\ 0x2f\ 0x8e\ 0x4e\ 0xe1$$
$$0xe6\ 0x76\ 0x6a\ 0x0a\ 0x88\ 0xd5\ 0xd8\ 0xc5$$
$$0x5c\ 0xae\ 0x98\ 0xc5\ 0xe4\ 0x81\ 0xf2\ 0x2a$$
$$0x69\ 0xbf\ 0x90\ 0x58\ 0x00\ 0x05\ 0x00$$

    Moreover, some of the fields of Skype SoM that should be variable, are instead fixed in these packets (i.e., *gmt_unix_time* field (bytes 12–15)—that should contain a time-stamp—and *random_bytes* field (bytes 16–43)—that should contain a random value).
- connection over port number 80
  - the payload size of R3 and R5 messages is 27 bytes and 4 bytes, respectively.

DOI: 10.1002/dac

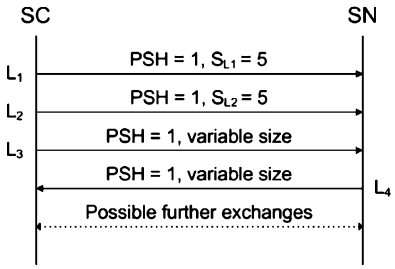

Figure 5. Skype Login Server Connection.

3.4. Skype TCP authentication

Once the connection has been established, the SC needs to authenticate itself to the server. In the TCP authentication phase, the SC opens a connection with the Login Server (that is the only centralized server of Skype architecture). Four packets are exchanged during the Skype Login Server Connection phase (L1, L2, L3, and L4), as depicted in Figure 5.

The main characteristics of these messages are the following:

- *L1*: The payload size is 5 bytes and it contains a fixed value (named A_3 in the following), that identifies a TLS Server Hello message

$$A_3 = 0x16\ 0x03\ 0x01\ 0x00\ 0x00$$

- *L2*: The payload size is 5 bytes and it contains a fixed value (named A_4 in the following)

$$A_4 = 0x17\ 0x03\ 0x01\ 0x00\ 0x00$$

- *L3*: The first 15 bytes have a fixed value (named A_5 in the following)

$$A_5 = 0x16\ 0x03\ 0x01\ 0x00\ 0xcd\ 0x41$$

$$0x03\ 0x00\ 0x09\ 0x80\ 0xc0\ 0x01$$

  while in a variable position we find A_4
- *L4*: The first 4 bytes have fixed value (named A_6 in the following)

$$A_6 = 0x17\ 0x03\ 0x01\ 0x00$$

It is to be noted that Skype application allows the user to save some authentication credentials. In that case, this message exchange does not take place when the application is launched.

3.5. Skype HTTP update

This message exchange is performed every time Skype is launched and it is aimed at retrieving the application updates. It consists of one unencrypted message, which varies depending on the Skype version, but presents either a fixed value (named A_7 in the following) in the first 29 bytes (Linux versions)

$$A_7 = \text{hexadecimal version of GET (http://ui.skype.com/ui/2/)}$$

or a fixed value (named A_8 in the following) in the bytes 95–124 (Windows versions)

$$A_8 = 0x48\ 0x54\ 0x54\ 0x50\ 0x2f\ 0x31\ 0x2e\ 0x31$$

$$0x0d\ 0x0a\ 0x55\ 0x73\ 0x65\ 0x72\ 0x2d\ 0x41$$

$$0x67\ 0x65\ 0x6e\ 0x74\ 0x3a\ 0x20\ 0x53\ 0x6b$$

$$0x79\ 0x70\ 0x65\ 0x99\ 0x20.$$

DOI: 10.1002/dac

4. SKYPE-HUNTER

In this section, we describe the algorithm introduced to perform a real-time classification of Skype traffic, detailing at first the approach used for UDP traffic (both signaling and data traffic) and then the one used for TCP traffic (both signaling and data traffic). Before discussing the algorithm, we list the parameters that are used in the following:

- D_i: payload size of packet i
- F: function field of Skype SoM
- t_i: arrival time of packet i
- $B_{i,j}$: bytes from i to j of the packet payload
- FLOW: UDP or TCP flow, identified by the 'classical' 5-tuple
- tentative: preliminary decision
- N_{0x0d}: number of packets with function field equal to 0x0d
- B_{sd}: number of bytes sent from the source to the destination
- B_{ds}: number of bytes sent from the destination to the source

By default, the algorithm produces two distinct output flows. The first is used to record all the events related to Skype control plane, while the second one is used to notify the detection of Skype calls or file transfers.

4.1. UDP traffic

The procedure developed for detecting and classifying Skype traffic over UDP can be split into two parts. Indeed, the detection of signaling traffic and data traffic is based on completely disjointed approaches.

4.1.1. Signaling traffic. Given a flow *FLOW*, the procedure aimed at detecting and classifying the signaling traffic over UDP consists of the following steps:

Packet#1: For being Skype signaling traffic, the function field of the first packet shall assume a predetermined value.

if first packet and $F == 0x02$ **then**
  $dimfirstpkt = D_1$
end if

Packet #2: Then the second packet dimension is checked for verifying whether the given $FLOW$ can be either a Skype UDP Ping or a Skype UDP Probe.

if second packet and $dimfirstpkt \neq 0$ **then**
  if $F \neq 0x02$ and $F \neq 0x07$ and $D_2 \notin \{11, 18, 26, 51, 53\}$ bytes **then**
    $dimfirstpkt = 0$
    $FLOW \neq Short\ Skype\ UDP\ Probe,\ Long\ Skype\ UDP\ Probe,\ UDP\ Skype\ Ping$
    EXIT
  end if
  if $F == 0x02$ and $D_2 \in \{18, 26, 51, 53\}$ bytes and $t_2 - t_1 < 10s$ **then**
    $FLOW = Short\ Skype\ UDP\ Probe$
    record the event into the log file 'skype_events.txt'
  else
    if $F == 0x02$ **then**
      $FLOW = Skype\ UDP\ Ping$
      record the event into the log file 'skype_events.txt'
    end if
  end if
end if

DOI: 10.1002/dac

Packet #3: The dimension of the third packet and its function field value are checked to eventually determine that the given $FLOW$ is not a Skype UDP ping or a Skype UDP Probe.

```
if third packet and dimfirstpkt ≠ 0 then
  if F ≠ 0x03 and D3 ≠ dimfirstpkt + 5 bytes then
    dimfirstpkt = 0
    FLOW ≠ Long Skype UDP Probe
    EXIT
  end if
  if F ≠ 0x02 then
    if FLOW == UDP Ping then
      FLOW ≠ Skype UDP Ping
      record the event into the log file 'skype_events.txt'
    else
      FLOW ≠ Skype UDP Ping
    end if
  end if
end if
```

Packet #4: Finally the dimension of the fourth packet and the value of the corresponding function field are used to classify the given $FLOW$ as either a Skype UDP Ping or a Skype UDP Probe.

```
if fourth packet and dimfirstpkt ≠ 0 then
  if F ≠ 0x02 then
    if FLOW == Skype UDP Ping then
      FLOW ≠ Skype UDP Ping
      record the event into the log file 'skype_events.txt'
    else
      FLOW ≠ Skype UDP Ping
    end if
  end if
  if F == 0x02 and D4 ∈ {18, 26, 51, 53} then
    FLOW = Long Skype UDP Probe
    record the event into the log file 'skype_events.txt'
  end if
end if
```

4.1.2. Data traffic. As far as the detection and classification of the data traffic are concerned, the system, given a flow, performs the following operations: for each received packet, if the interarrival time with respect to the previous one is less than 3 s, computes the quantities N_{0x0d}, B_{sd}, and B_{ds}. When N_{0x0d} exceeds a given threshold (experimental results have shown that a value of 95 packets allows us to obtain good performance with an acceptable detection time), the system computes ratio $= B_{ds}/B_{sd}$ and distinguishes between three events:

- direct call if ratio > 0.5
- file transfer if $0 <$ ratio < 0.5
- call with relay if ratio $= 0$.

One drawback of this procedure is that a call, where each user forwards the packets through a relay node (Figure 1(c)), will be classified as a direct call.

Moreover, the system is also able to correctly detect SkypeOut calls, by using a very similar procedure. The only difference is given by the fact that, after the first messages have been exchanged, all the packets present in the first 4 bytes have a fixed value, the CID. Moreover, a SkypeOut call

DOI: 10.1002/dac

is characterized by packets forwarded towards a gateway using UDP port number 12340. Thus, in more detail, the system counts the packets of a given flow that respect the following conditions:

- source or destination port equal to 12430
- payload size greater than 3 bytes
- interarrival time with respect to the previous one less than 3 s
- first 4 bytes have the fix value CID.

If the number of such packets exceeds a given threshold (95 packets, as in the previous case), the flow is decided to be a SkypeOut call.

4.2. TCP traffic

The procedure developed for detecting and classifying Skype traffic over TCP can be split into two parts. Indeed, as for UDP traffic, the detection of signaling traffic and data traffic is based on completely disjointed approaches.

4.2.1. Signaling traffic. Given a flow *FLOW*, the procedure aimed at detecting and classifying the signaling traffic over TCP consists of the following steps:

Packet #1: The system checks the dimension of the first packet, the destination port, and some portions of the payload, to verify whether the given $FLOW$ can be part of one of the following message exchanges: Skype HTTPS handshake, Skype Login Server Connection, or Skype HTTP Update.

if first packet **then**
  if destination port$==443$ and $D_1 > 55$ bytes and $B_{1,56} == A_1$ **then**
    $tentative = Skype\ HTTPS\ Handshake$
  end if
  if $payload == A_3$ **then**
    $tentative = Skype\ Login\ Server\ Connection$
  end if
  if destination port$==80$ and $D_2 \in \{167, 169, 175\}$ bytes and ($B_{1,29} == A_7$ or $B_{95,124} == A_8$) **then**
    $FLOW = Skype\ HTTP\ update$
    record the event into the log file 'skype_events.txt'
  end if
end if

Packet #2: The dimension of the second packet and some portions of the payload are checked to verify whether the given $FLOW$ can be either a Skype Login Server Connection or a Skype HTTPS Handshake.

if second packet **then**
  if $D_2 > 4$ bytes and $tentative == Skype\ Login\ Server\ Connection$ and $B_{1,5} == A_4$ **then**
    $tentative = 0$
  end if
  if $D_2 > 78$ bytes and $tentative == Skype\ HTTPS\ Handshake$ and $B_{1,80} \neq A_2$ **then**
    $FLOW = Skype\ HTTPS\ Handshake$
    record the event into the log file 'skype_events.txt'
    # the source is a SN
  end if
end if

DOI: 10.1002/dac

Packet #3: The dimension of the third packet and some portions of the payload are checked to verify if the given $FLOW$ can be either a Skype Login Server Connection or a Skype HTTP/TCP Handshake.

if third packet **then**
  if $D_3 > 4$ bytes and $tentative == Skype\ Login\ Server\ Connection$ and $B_{1,15} \neq A_5$ **then**
    $tentative = 0$
  end if
  if $D_3 == 27$ and destination port == 80 **then**
    $tentative = Skype\ HTTP\ Handshake$
  end if
  if $D_4 = 27$ bytes **then**
    $tentative = Skype\ TCP\ Handshake$
  end if
end if

Packet #4: The system checks the dimension of the fourth packet and some portions of the payload to verify if the given $FLOW$ can be a Skype Login Server Connection.

if fourth packet **then**
  if $D_4 > 3$ bytes and $tentative == Skype\ Login\ Server\ Connection$ and $B_{1,4} \neq A_6$ **then**
    $FLOW = Skype\ Login\ Server\ Connection$
    record the event into the log file 'skype_events.txt'
  end if
end if

Packet #5: The dimension of the fifth packet is checked to eventually classify the given $FLOW$ as either a Skype HTTP Handshake or a Skype TCP Handshake.

if fifth packet **then**
  if $D_5 == 4$ bytes and $tentative == Skype\ HTTP\ Handshake$ **then**
    $FLOW = Skype\ HTTP\ Handshake$
    record the event into the log file 'skype_events.txt'
    # the destination is a SN
  end if
  if $D_6 == 4$ bytes and $tentative == Skype\ TCP\ Handshake$ **then**
    $FLOW = Skype\ TCP\ Handshake$
    record the event into the log file 'skype_events.txt'
    # the destination is an SN
  end if
end if

4.2.2. Data traffic. In the case of TCP traffic, Skype application completely encrypts the packets payload, making it impossible to reveal a call by means of any signature. Our method is thus based on a statistical analysis of the traffic. In more detail, we are able to detect Skype traffic (in this case we cannot distinguish between calls and file transfers) in two distinct ways: revealing Skype TCP Ping traffic or the connection tear-down signaling traffic.

In the first case, the idea is to reveal Skype calls based on the fact that during a call Skype application periodically sends some messages with a payload size of 4 bytes and with an inter-time which is a multiple of 1 s. Thus, the system counts the number of 4 bytes long packets that satisfy the following conditions:

- $t_i - t_{i-1} \in [1, 10]$ s
- the difference between the 'microseconds part' of the two timestamps $\in [-15, 15]$.

DOI: 10.1002/dac

Table I. Skype score.

Skype score	Skype events
1	Simple TCP call
2	TCP call **and** (Skype TCP Handshake **or** Skype UDP Probe)
3	TCP call **and** Skype TCP Handshake **and** Skype UDP Probe
4	TCP call **and** (Skype Login Server Connection **or** Skype HTTP Update)
5	Simple UDP call
6	UDP call **and** (Skype TCP Handshake **or** Skype UDP Probe)
7	UDP call **and** Skype TCP Handshake **and** Skype UDP Probe
8	UDP call **and** (Skype Login Server Connection **or** Skype HTTP Update)

If such a number exceeds a given threshold and the mean dimension of the packets received prior to the first 19 bytes long packet $\in [10, 600]$ bytes, the traffic flow is decided to be a Skype flow.

Moreover, if the following conditions are fulfilled:

- number of packets greater than 4
- mean packet dimension $\in [100, 600]$ bytes
- total number of bytes belonging to the flow greater than 40 kbytes

then the flow is decided to be a call, otherwise it is considered as signaling traffic.

The other approach for detecting a Skype call is based on the detection of the traffic generated by the tear-down phase. In this case the system counts the number of 19 bytes long packets, for which $t_i - t_{i-1} < 3$ s. If such number exceeds a given threshold and the mean packet dimension $\in [10, 600]$ bytes, the traffic flow is decided to be a Skype flow.

Moreover, if the following conditions are respected:

- number of packets is greater than 6
- mean packet dimension $\in [100, 600]$ bytes
- total number of bytes belonging to the flow is greater than 40 kbytes

then the flow is decided to be a call, otherwise it is considered as signaling traffic. It is to be noted that, in this case, the reception of a packet, whose dimension is greater than 150 bytes, resets the counter. This is justified by the fact that we should not observe data packets in the call tear-down phase.

After having detected a Skype flow, the system gives a score to the flow, indicating how much the detection is reliable (see Table I).

It is to be noted that, in general, we always consider the detection of an UDP call more reliable than the detection of a TCP call. This is due to the fact that, while for UDP traffic we are able to verify some signatures, for TCP traffic the encryption mechanism does not allow us to check for any standard pattern in the packets. Moreover, we assume that we are sure of the detection results if the system detects either a Skype Login Server Connection or a HTTP Update, since in that case the involved user is surely using Skype.

5. EXPERIMENTAL RESULTS

To verify the effectiveness of our algorithm, we compare it with standard statistical classifiers and the state-of-the-art Skype classifier originally proposed in [7] (the latter has been downloaded from the authors' web site).

To this aim, we have tested the system both offline and online (real time). Regarding the offline testing, we have used three distinct data sets. The first one is represented by the DARPA data set [16], a collection of publicly available traffic traces, realized by the MIT Lincoln Laboratory on Information Systems Technology. This traffic is representative of a small LAN of the US force directly connected to the Internet. In this data set, realized in 1999, there obviously does not appear

DOI: 10.1002/dac

any Skype traffic. It is to be noted that we have used this data set, although being clearly old and simplistic, only because being publicly available it can be easily downloaded by everyone to verify the results. The second data set [17] is composed of the traffic traces used in [7] and only contains standard Skype calls and SkypeOut calls. Finally, the third data set (around 3 GB of traffic data—in libpcap format) has been realized partly over a laboratory LAN in our Dept., and partly over a small LAN connected to the Internet through an ADSL link. In this way, our data are representative of the types of traffic generated in a research lab as well as by home users. In more detail, the third data set contains, among the others, the following applications (2000 flows per application):

- eMule (both over TCP and UDP)
- Bittorrent (both over TCP and UDP)
- FTP
- HTTP/HTTPS
- MSN (both chats and calls)
- SNMP
- Skype (both over TCP and UDP).

Instead, for the online testing, we have installed the system over the gateway of our laboratory LAN.

As far as Skype traffic is concerned, we have considered traffic between two end hosts running SCs (including voice calls, chats and file transfers) as well as calls to traditional PSTN phones (SkypeOut calls). For the sake of generality, we used SCs running under Linux (versions 1.4, 2.0, and 2.1) as well as under Windows (versions 3.2, 3.6, 4.0, and 4.1).

The goal of our algorithm is to correctly identify Skype flows, hence as performance metrics we have considered the percentage of False Positives (FP) and False Negatives (FN). Since the lengths of the analyzed flows are quite different, these quantities have been estimated on a per-flow and on a per-byte basis, in order to have an insight not only into the number of flows incorrectly classified, but also into the involved amount of traffic.

As far as 'general purpose' statistical classifiers are concerned, we implemented the following algorithms [18] that are the most widely used ones:

- Naive Bayes Classifier (NB)
- Linear Discriminant Analysis (LDA)
- k-Nearest Neighbor (k-NN)
- Support Vector Machine (SVM).

The parameters of the classifiers have been tuned during an appropriate training phase that has been performed using one half of the traffic traces (the remaining ones have been used for the classification).

At first the TCP data have been processed by TStat [19] that calculates a set of significant features for each connection, while for the UDP traffic we have started by extracting all the 'reasonable' features. Then, in order to squeeze the most out of these classifiers, we have applied the Whitney method [18] to extract the 'best' features, and then we have normalized them, so as to obtain homogeneous values. Finally, by means of a cross-validation process we have limited the number of useful features to 31 for TCP traffic and 12 for UDP traffic (see Tables II and III, respectively).

Regarding the first data set (DARPA), the system has achieved a 'correct' behavior. Indeed it has not revealed any Skype flow. As far as the second data set is concerned the obtained results are presented in Table IV.

As we can see from the table, the system performs better over TCP traffic than over UDP traffic. In any case the percentage of non-detected bytes is always very low, even when the percentage of non-detected flow seems to be not negligible. This means that our system does not always correctly reveal some short flows, which correspond to signaling traffic, while the data traffic is almost always correctly detected.

DOI: 10.1002/dac

Table II. TCP features.

number of client packets
0 if there are not RST segments from client, 1 otherwise
number of ACKs sent by client
number of pure ACKs sent by client
number of bytes sent by client
number of SYNs sent by client
number of FINs sent by client
window scale option (client packet)
timestamps option (client packet)
number of SACKs sent by client
average client RTT (ms)
minimum client RTT (ms)
maximum client RTT (ms)
client RTT standard deviation
number of server packets
0 if there are not RST segments from server, 1 otherwise
number of ACKs sent by server
number of pure ACKs sent by server
number of bytes sent by server
number of SYNs sent by server
number of FINs sent by server
window scale option (server packet)
timestamps option (server packet)
number of SACKs sent by server
average server RTT (ms)
minimum server RTT (ms)
maximum server RTT (ms)
server RTT standard deviation
flow duration
time between the first flow segment and the first data segment
time between the first flow segment and the last data segment
time between the first flow segment and the first client packet
time between the first flow segment and the first server packet

Table III. UDP features.

number of packets per flow
flow packet size variance
client packet size average
client packet size variance
server packet size average
server packet size variance
ratio between the number of server and client packets
ratio between the number of server and client bytes
minimum packet dimension
maximum packet dimension
minimum inter-arrival time (client)
number of byte sent by the client

The results, obtained over the third data set, are summarized in Tables V and VI for UDP and TCP flows, respectively.

The previous results highlight that our algorithm outperforms both the general-purpose statistical approaches and Skype classifier [7]. The high value of FN at flow level for TCP traffic is due to the fact that up-to-date features characterizing such exchanges of data between SCs have not been identified yet. Nevertheless, it is important to highlight that, as in the previous case, the unclassified flows correspond to a very low quantity of traffic (bytes), which means that the system is not classifying the signaling traffic, while the calls are correctly classified.

DOI: 10.1002/dac

Table IV. Second data set.

	SkypeOut (UDP)	Skype and SkypeOut Calls (TCP)
FN (Flows) %	12.79	5.56
FN (Bytes) %	0.14	1.86

Table V. Third data set: UDP flows.

	Our algorithm	Skype classifier	NB	LDA	k-NN	SVM
FN (Flows) %	4.8	97.57	11.69	9.44	1.9	2.01
FN (Bytes) %	0.06	26.32	95.98	95.98	6.86	7.88
FP (Flows) %	0	2.4	11.83	13.74	10.95	14.94
FP (Bytes) %	0	16.85	6.12	7.27	5.33	17.47

Table VI. Third data set: TCP flows.

	Our algorithm	Skype classifier	NB	LDA	k-NN	SVM
FN (Flows) %	27.46	84.95	28.41	20.04	24.59	24.44
FN (Bytes) %	0.64	56.38	33.07	14.45	27.01	39.48
FP (Flows) %	0.01	9.68	4.49	6.95	5.08	3.37
FP (Bytes) %	0.001	94.58	7.27	2.48	0.56	0.11

Table VII. Real-time classification.

Skype event	Mean detection time
UDP Call	1.4 s
UDP Call with relay	1.5 s
TCP Call	End of the call
TCP Call with relay	18.7 s
UDP File Transfer	1.2 s
UDP File Transfer with relay	1.2 s
TCP File Transfer	End of the file transfer
TCP File Transfer with relay	19.3 s

It is also important to highlight that the same consideration also holds for the Skype classifier, indeed this classifier is not aimed at detecting the Skype signaling traffic. This consideration explains the big quantity of false alarms generated by the classifier.

Finally, concerning the real-time classification, since all the LAN users were monitored, we can state that the system has correctly classified all the Skype traffic (both signaling and data traffic), without any FP. Table VII shows the mean detection times for the online detection of the different Skype activities.

From the table we can easily note that the standard calls are detected quite immediately, while more time is necessary for detecting the TCP calls with relay node. Indeed, in this case, the system has to wait for a significant number of TCP Ping. Finally, for the direct TCP calls, since it is impossible to observe the TCP Ping messages, it is necessary to wait for the call tear-down phase.

As a general consideration, we can easily note that our method is able to achieve very good performance, outperforming all the classical classification algorithms as well as the 'Skype classifier'. Regarding the first ones, the main improvement is given by the fact that our algorithm combines statistical and signature-based approaches. Indeed, given that Skype behavior is 'statistically' analogous to other P2P systems, the use of the signatures allows to significantly improve the performance achieved by statistical classifiers. Moreover, in comparison with the latter, as already said, another great advantage of our approach is given by the fact that it is also able to detect the signaling traffic, that cannot be detected by the 'Skype classifier'.

DOI: 10.1002/dac

Finally, we have measured the memory and CPU consumption of the algorithm. The tests have been realized by using the proposed system over a standard PC (Intel Pentium 4 2.5 GHz, 1 GB RAM), connected to the mirroring port of the LAN switch. The results have shown that, even when the network becomes congested, the memory usage is less than 1%, while 60% of the CPU has been used at maximum (comprising the resources allocated to the OS).

6. CONCLUSIONS

In this paper we have proposed a real-time algorithm to detect and classify Skype traffic. In more detail, the presented method, by means of both signature-based and statistical procedures, is able to correctly reveal and classify the signaling traffic as well as the data traffic (calls and file transfers).

The performance analysis has shown that our algorithm achieves very good results over several types of traffic traces, representative of different access networks. Moreover, we have shown that our system outperforms the 'classical' statistical traffic classifiers as well as the state-of-the-art *ad hoc* Skype classifier.

ACKNOWLEDGEMENTS

The authors thank Federico Gentile and Fabio Strazzera for their contribution in support of the presented work. This work has been partially sponsored by the European Project FP7-ICT PRISM, contract number 215350.

REFERENCES

1. Skype web site. Available from: http://www.skype.com [10 April 2009].
2. Telegeography web site. Available from: http://www.telegeography.com/ [10 April 2009].
3. Skype by the numbers web site. Available from: http://apple20.blogs.fortune.cnn.com/2009/03/31/skype-by-the-numbers/ [10 April 2009].
4. Skype users online now web site. Available from: http://idisk.mac.com/hhbv-Public/OnlineNow.htm [10 April 2009].
5. Baset SA, Schulzrinne HG. An analysis of the Skype peer-to-peer internet telephony protocol. *INFOCOM' 06*: *Proceedings of the 25th IEEE International Conference on Computer Communications*, 2006; 1–11. Available from: http://dx.doi.org/10.1109/INFOCOM.2006.312.
6. Adami D, Callegari C, Giordano S, Pagano M, Pepe T. A Real-Time Algorithm for Skype traffic detection and classification. *Proceedings of NEW2AN*, St. Petersburg, Russia, 2009.
7. Bonfiglio D, Mellia M, Meo M, Rossi D, Tofanelli P. Revealing Skype traffic: when randomness plays with you. *SIGCOMM Computer Communication Review* 2007; **37**(4):37–48. Available from: doi:http://doi.acm.org/10.1145/1282427.1282386.
8. Suh K, Figueiredo DR, Kurose J, Towsley D. Characterizing and detecting Skype-relayed traffic. *Proceedings of IEEE INFOCOM*, Barcelona, Spain, 2006.
9. Lu L, Horton J, Safavi-Naini R, Susilo W. Transport layer identification of Skype traffic. *Proceedings of ICOIN*, Estoril, Portugal, 2007.
10. Svoboda P, Hyyti E, Ricciato F, Rupp M, Karner M. Detection and tracking of Skype by exploiting cross layer information in a live 3G network. *Proceedings of TMA*, Aachen, Germany, 2009.
11. Guha S, Daswani N, Jain R. An experimental study of the Skype peer-to-peer voip system. *IPTPS'06*: *The Fifth International Workshop on Peer-to-Peer Systems*, Microsoft Research, 2006. Available from: http://saikat.guha.cc/pub/iptps06-skype.pdf.
12. Rossi D, Mellia M, Meo M. Understanding Skype signalling. *Computer Networks* 2009; **53**(2):130–140. Available from: doi:http://dx.doi.org/10.1016/j.comnet.2008.10.013.
13. How Skype & co. get round firewalls web site. Available from: http://www.heise-online.co.uk/security/How-Skype-Co-get-round-firewalls–/features/82481 [10 April 2009].
14. Rosenberg J, Weinberger J, Huitema R. STUN—simple traversal of user datagram protocol (udp) through network address translators (nats). *RFC 3489*, RFC Editor, U.S.A., 2003.
15. Mahy R, Matthews P, Rosenberg J. Traversal using relays around NAT (TURN): relay extensions to session traversal utilities for NAT (STUN). *RFC 5766*, RFC Editor, U.S.A., 2010.
16. Ideval project web site. Available from: http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html [10 April 2009].
17. Tstat project data-set. Available from: http://tstat.tlc.polito.it/traces-skype.shtml [10 April 2009].
18. Duda RO, Hart PE, Stork DG. *Pattern Classification*. Wiley: New York, 2000.
19. Tstat—tcp statistic and analysis tool web site. Available from: http://tstat.tlc.polito.it/index.shtml [10 April 2009].

AUTHORS' BIOGRAPHIES

Davide Adami received a degree in Electronic Engineering from the Department of Information Engineering at the University of Pisa, Italy, in 1992.

From 1993 to August 1997 he worked at Consorzio Pisa Ricerche in the field of computer and Telecommunications networks, with particular focus on dial-up internetworking of TCP/IP networks, taking part in a lot of research projects funded by the Regione Toscana and the European Community. Within the project Internet, he was the Network Administrator of Towernet, one of the earliest ISPs in Italy, where he was concerned with policy routing and intradomain routing protocols (RIP, OSPF). In the framework of the MAESTRO ACTS project he was dealing with Quality of Service Networks in ATM networks and Resource Allocation Techniques (RSVP).

In September 1997 he joined the CNIT (National Consortium for Telecommunications), where he is a senior researcher in the field of telecommunications networks. His research interests mainly concern the provisoning of Quality of Service in IP networks and the TCP/IP enhancements for satellite networks. He is one of technicals responsible for the CNIT satellite network.

Currently, he is also an Assistant Professor at the Faculty of Engineering of the University of Pisa in the courses of 'Planning and Simulation of Telecommunications Networks' and 'Systems and Services of Telecommunications Networks'.

Christian Callegari received his Laurea degree in Telecommunication Engineering 'cum laude'on October 2004 from the University of Pisa, discussing a thesis titled 'Simulative analysis of RSVP-TE, and evaluation of end-to-end rerouting techniques in MPLS networks'. In 2005 he obtained the qualification to practice the profession of Engineer and he is a member of the IEEE Communication Society. On January 2005 he joined the department of Information Engineering at the University of Pisa as a PhD student, and in 2008 he obtained his PhD in Information Engineering. He is actually a Postdoctoral fellow in the Telecommunication Network research group in the department of Information Engineering of the University of Pisa. His research and professional areas of interest are Network Security, Traffic Anomaly Detection, and Traffic Analysis and Classification.

He has co-authored around 50 papers published in international journals and presented in leading international conferences.

Stefano Giordano received the Laurea degree 'cum laude' in Electronics Engineering and the PhD degree in Information Engineering from the University of Pisa in 1990 and 1994, respectively.

He worked with Consorzio Pisa Ricerche since 1990 in the field of Telecommunication Networks participating and coordinating several research activities. Since the end of 2001 he is associate professor at the Department of Information Engineering of the University of Pisa (Telecommunication Networks Group) where he gives lectures on 'Telecommunication Networks', and 'Design and Simulation of Telecommunication Networks'. His research and professional areas of interest are Broadband Communications, Telecommunication Networks Analysis and Design, Simulation of Communication Networks and Systems, Multimedia communications.

Michele Pagano received the laurea (cum laude) in Electronics Engineering in 1994 and a PhD in Electronics Engineering in 1998, both from the University of Pisa. Since 2007, he is an associate professor in the Dipartimento di Ingegneria dell'Informazione of the University of Pisa, where he is the official instructor of the courses of 'Telematics', 'Performance of Multimedia Networks' and 'Network Security'.

His research interests are related to statistical characterization of traffic flows and to network performance analysis, mainly in the framework of architectures able to support Quality of Service. Performance evaluation has been carried through analytical approaches as well as by means of discrete event simulation. Finally, a new research field is represented by network security issues, mainly in the framework of Intrusion Detection Systems.

He has co-authored around 100 papers published in international journals and presented in leading international conferences.

DOI: 10.1002/dac

Teresa Pepe received her Laurea degree in Telecommunication Engineering 'cum laude' on September 2008 from the University of Pisa, discussing a thesis titled 'Design and simulative analysis of a new algorithm for cost and performance optimization of Service Overlay Network'.

In 2009 she obtained the qualification to practice the profession of Engineer.

On November 2008 she won a grant funded by the MIUR for a PhD position in Information engineering in the Department of Information Engineering of the University of Pisa. She is actually a PhD Student in the Telecommunication Network research group of the Department of Information Engineering at the University of Pisa.

Her research and professional areas of interest are Network Security and Traffic Classification.

DOI: 10.1002/dac