Hiding Skype VoIP Calls from Parametric Identification

Mauro Migliardi
Centro Ingegneria Piattaforme Informatiche (CIPI)
University of Genoa and University of Padua
Via Opera Pia 13
16145 Genoa, Italy
+39 328 1003221
mauro.migliardi@unipd.it

Roberto Podesta'
DIST – University of Genoa
Via Opera Pia 13
16145 Genoa, Italy
+39 010 3532709
ropode@dist.unige.it

Matteo Tebaldi
DEI - University of Padua
Via Gradenigo 6
35131 Padua, Italy
tebaldi.matteo@gmail.com

Massimo Maresca
Centro Ingegneria Piattaforme Informatiche (CIPI)
University of Genoa and University of Padua
Via Opera Pia 13
16145 Genoa, Italy
Massimo.maresca@unipd.it

ABSTRACT
The proliferation of wideband connections and the use of more innovative technological platforms in both the business and the private market have caused a shift of the phone traffic from traditional circuit switched networks to packet switched IP based networks in the so-called Voice Over IP revolution. The flexibility of VoIP in general and of Voice Over Internet in particular comes at a price: the lack of a dedicated carrier infrastructure makes extremely hard to identify calls and to track them, thus lawful interception has turned extremely difficult as voice is mixed and randomly interleaved with generic data traffic. In past works it has been demonstrated the feasibility of call identification based on statistical parameters of data flow. However, we argue that it is extremely easy to "poison" the statistical characteristics of a VoIP stream making it stealthy without compromising its acoustical quality and we prove it first implementing a statistical detector showing a larger than 95% accuracy in detection and then a simple program capable of scrambling the statistical characteristics of Skype data streams.

Categories and Subject Descriptors
H.4.3 [**Communication Applications**]

General Terms
Management, Measurement, Security.

Keywords
VoIP, Skype, Call identification, Parametric Identification.

1. INTRODUCTION
The proliferation of wideband connections and the use of more innovative technological platforms have caused a shift of the phone traffic from traditional circuit switched networks to packet switched IP based networks in the so-called Voice Over IP revolution. VoIP, acronym for Voice over IP, is a large and composite set of technologies whose purpose is to carry speech through an IP network. This includes several processes, from voice digitalization, to packet deployment and transport [1]. So there isn't just one standard protocol but there are several ways to complete each step which are combined in many different stacks.

VoIP can offer a great benefit for the user privacy. In many implementations voice traffic is encrypted and travel from a peer to the other without passing any central commutation structure where a telephone tapping system can be installed. On the other hand, for technical or security reasons, network administrators and police enforcements require the chance of performing lawful interception. This task is not so difficult if the system monitored runs on dedicated server or if the product design offers this possibility. The problem arises when considering some very smart softphones running on a peer-to-peer network. Skype is probably the most diffused and controversial one for the lack of control on it from an administrator point of view. For its strong encryption algorithm and the ability of its protocol to traverse many network protections it can sometimes be considered a threat for networks security [2].

In a leaf node, it would be feasible to perform a content based analysis of each flowing packet to identify VoIP calls, however, such an approach presents several pitfalls:

- for privacy reasons, it is not possible to analyze the payload of every packet;
- the sheer number of packets flowing through a carrier switch makes computationally unfeasible to perform an effective analysis of each packet's payload;

E-FORENSICS 2008, January 21-23, Adelaide, Australia

DOI 10.4108/e-forensics.2008.2673

- some smart soft phones such as Skype encrypt both data and signaling packets thus make impossible to gather call setup information in real time.

Previous works [3][4] show that VoIP packet streams present some very distinctive characteristics that can be used to identify them even if some refinements are necessary in order to be able to identify stealthy soft phones such as Skype and to avoid false positives caused by media streams such as web radio. We adopt this approach and we define the statistical signatures of Skype data streams in order to allow fast identification of these streams at gateway level, then we show that this approach is prone to be fooled by stealthing techniques based on statistical "poisoning" of the audio data streams.

More in details, in this paper we describe how the call detection technique based on statistical parameters is to be used in the case of the voice streams generated by one of the most widely adopted soft phone, namely Skype. Thus, we perform the analysis of the data streams statistical characteristics, we identify the similarities with other similar media streams such as web radio and we design a model capable of preventing false positives. Finally we implement a software capable of demonstrating how it is possible to use this model to identify calls with a very low error rate. Finally, we show that the statistical peculiarities that allowed identifying Skype calls with such a good error rate can be masked and that the data streams can be easily morphed into ones that do not show those peculiarities without compromising the audio quality of the call. This proves the fragility of this kind of detection.

This paper is structured as follows, in section 2 we provide a brief introduction to Voice over IP in general and we describe the Skype soft phone architecture; in section 3 we describe how we designed our model, the empirical data onto which we based the parameters evaluation and we describe the proof-of-concept implementation of the detector; in section 4 we describe how it is possible to "poison" the statistical characteristics of a Skype audio stream to prevent its identification and we show a proof-of-concept implementation of a software capable of performing such a poisoning; finally, in section 5 we provide some concluding remarks.

2. VOICE OVER IP TELEPHONY AND SKYPE

Skype [5] is a peer-to-peer Internet telephony network, developed in 2003 by Niklas Zennström e Janus Friis. Nowadays Skype is the most used VoIP software to perform calls over the Internet: it was reported that nine million concurrent Skype users were online as of January 29, 2007. The code and the protocol used by Skype are not public but there have been several studies aimed at understanding how it works [6] [7] [8].

There are three main components in the Skype network:

• The Skype login server: is one of the few central elements of the network. When a user want to access the Skype network has to be authenticated through this server.

• Skype clients: is a normal user connected to the Skype network

• Super node: is a Skype client which routes request to appropriate destinations and servers, answer queries from other clients and forward login request in case the login server is not directly reachable from a client. Any client can become a super node.

It has been measured that the Skype codecs allow frequencies between 50 and 8000 Hz to pass through, a range characteristic of a wideband codec. Skype is supposed to use iLBC [9], iSAC [10] implemented by GlobalIPSound or a proprietary codec known as SVOPC. With the release of the new Skype 3.2 beta version there are some quality improvements with a better echo suppression.

Both signaling and media traffic are encrypted for any kind of Skype connection. As reported on the Skype website, Skype uses AES (Advanced Encryption Standard) and the strength of Skype encryption has been confirmed by several independent studies [11].

Most Voice over Internet Protocol (VoIP) solutions are designed for enterprise environments and many residential broadband users would be unable to make VoIP calls without complex reconfiguration of their routers and firewalls, often because the customer's network includes a restrictive firewall or a network address translation (NAT) gateway, neither of which are easily capable of carrying VoIP calls directly. Skype's P2P architecture solves this, allowing calls from users located behind a firewall or a NAT gateway to be transparently routed through the help of a peer that is un-firewalled. This means that anyone can use Skype to make VoIP calls without the need to reconfigure routers or firewalls.

3. SKYPE DETECTOR

Many models for VoIP traffic have been proposed like [12] and [13], but for their complexity they are not suitable for our kind of analysis: we need something lighter and more adaptable, thus we adopt the identification approach based on simple statistical parameters suggested in [3].

In order to ensure a good quality, VoIP traffic has to fulfill some strong realtime constraints. This is a first characteristic we could exploit, expecting a high transmission rate of packets between the two peers. This also suggest that the size of these packets will be quite small and regular. Skype uses no acknowledgement system and moreover we cannot assume we will catch packets used to establish or close the communication, so we should

Table 1. Statistics of a 1 minute call

	Peer A		Peer B	
	Outgoing	**Incoming**	**Outgoing**	**Incoming**
number of packets	1985	1993	2002	1990
duration [s]	66.721	68.466	72.699	72.555
IPD mean [s]	0.0336	0.0344	0.0363	0.0365
IPD variance [s]	0.0218	0.0221	0.0233	0.0252
IPD maximum [s]	6.594	6.3887	6.3887	6.5939
PS mean [bytes]	162	151.6	151.1	161.8
PS variance [bytes]	1.6241×10^3	1.589×10^3	1.6280×10^3	1.6305×10^3
PS maximum [bytes]	1.026	597	597	1.026

focus our attention onto the traffic produced in the middle of the conversation.

For these reasons, we focus our attention onto packet size and inter packet delay in a stream of packets characterized by the quadruple Source IP, Source Port, Destination IP, Destination Port. If we consider packets sent from a peer to another and enumerate them 1, 2, … n, the Inter Packet Delay (IPD) of the i-th packet is:

$$ipd(i) = timestamp(i) - timestamp(i-1)$$

The values of this parameter is easily measured in real-time as it requires very little computation. The parameters values have to be adapted to the codec (e.g. iLBC or iSAC), but the model is general and can be easily adapted to any configuration.

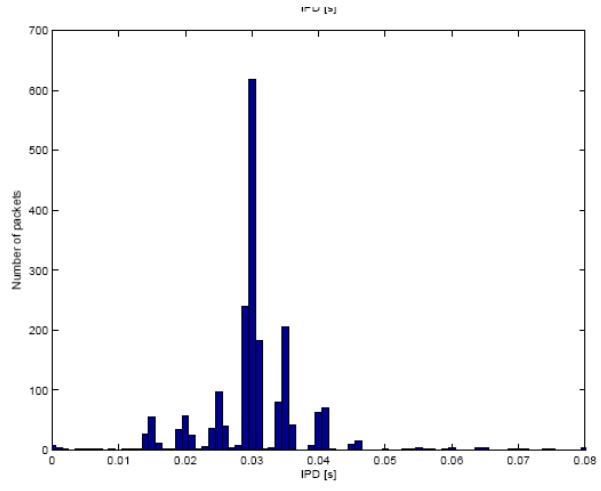

Figure 1 Histogram of Inter Packet Delay (IPD)

The test environment is formed by two PC (peer A and B) on the same LAN with the same hardware and software configuration.

We performed packet sniffing from each of the two peers with Wireshark [14]. Then, having timestamp, IPD and size of each packet, we calculated the effective duration of the observed call, the number of packets in the call, the mean, variance and maximum IPD, the mean, variance, minimum and maximum packet length over the whole call. In table 1 we show the statistical values we measured on a 1 minute call. In our experiments we also measured the parameter values for 5 and 30 minutes calls.

Once the communication starts, packet transmission appears to be very regular, settling the average IPD value at about 30 ms, regardless of the versions of the clients or of the call length. Average IPD variance is about 3×10^{-5} resulting in a standard deviation of 5.48×10^{-3} and even with some variation depending from the client versions and network traffic has always the same order of magnitude in every test. As it can be easily seen in figure 1, the IPD values have such marked characteristics that they definitely represents a primary feature on top of which it is possible to build the foundation of the call detection system.

As figure 2 clearly shows, the packet size has an, albeit slightly less marked, similarly distinctive statistic. Thus, we include this second parameter in our identification model.

In order to evaluate how a generic packets stream fits into our model we will use two functions, one for IPD and the other for packet size behaviour.

The similarity function used to monitor IPD likeness is ipdact, a sort of moving average defined as:

$$ipdact(n) = \alpha \times ipd(n) + (1 - \alpha) \times ipdact(n-1)$$

where ipd(n) is the IPD of the current packet and α is the weight of his IPD in the sum. ipdact is calculated for each packet and then compared with an upper threshold (ipdtup) and a lower threshold (ipdtdown).

An analogous similarity function, psact, is used for the packet size:

$$psact(n) = \beta \times ps(n) + (1 - \beta) \times psact(n-1)$$

where ps(n) is the size of the packet and β is his weight in the function. After it has been calculated for each packet, the value of this function too is compared with an upper threshold (pstup) and a lower threshold (pstdown). When

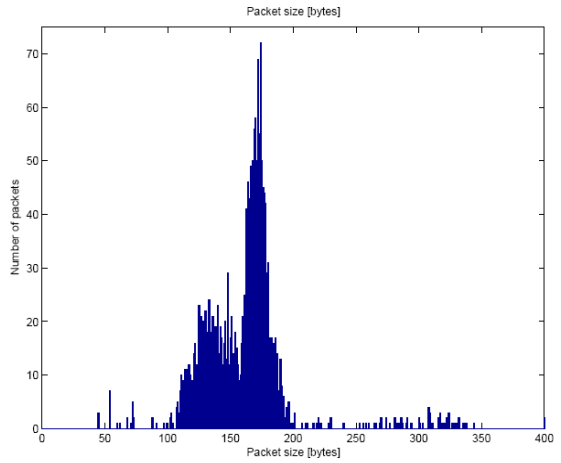

Figure 2 Histogram of Packet Size (PS)

the values of this functions are between the thresholds we can state there is a Skype communication.

To summarize, ipdact parameters are:α, ipdact initial value, ipdtup, ipdtdown. While psact parameters are: β, psact initial value, pstup, pstdown.

α and β obviously control how quickly the measure follows the new values and how sensitive it is to data fluctuations, their values are extremely important to avoid having a system too slow to identify changes or too unstable because of small fluctuations. Furthermore, a system too sensitive could easily identify as a Skype call a stream whose characteristics happen to "look like" a Skype call for a short period of time.

Several experiments allowed us to determine that with a value of α equal to 0.1 and a value of β equal to 0.05 we obtained an accuracy close to 98% and a delay in positive identification of less than 1 second. The optimal values for all the parameters are shown in table 2.

Even with accurately chosen values of α and β, we ran into a set of data streams whose characteristics consistently and

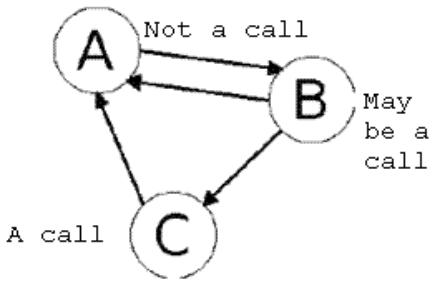

Figure 3 Finite State Machine of the detector

Table 2. Optimal values of the Similarity functions parameters

parameter	value
α	0.1
ipdtup	0.045 s
ipdtdown	0.020 s
β	0.05
pstup	170 bytes
pstdown	125 bytes

for prolonged periods of time coincided with the ones we considered to be the statistical signature of Skype calls. These streams were different forms of audio webcasts that happened to use one of the codecs adopted by Skype. In order to eliminate these problematic false positives, we observed that Skype calls always generate a couple of symmetric streams, while webcasts are always one way. Thus, we enhanced our model with a symmetry constraint, i.e. a stream that has been matched by our similarity functions is a Skype call if and only if there is another stream matched by our similarity functions going in the opposite direction. The addition of this constraint allowed us to completely eliminate the risk of false positives caused by web radio and any other kind of audio webcast.

We implemented our detector as a C application. We used the POSIX Threads library in order to deal with concurrent activities and we leveraged the pcap library [15] to capture network packets in real time. In our program a channel is an object defined by an ordered quadruple of IP addresses and port numbers. The first address-port couple represent the source and the second the destination of a packets stream. Channel also have other fields and methods used to determine if it fits our model. The main attributes are:

- ipdact, represent the IPD index of the channel;
- psact, represent the packet size index of the channel;
- call, represents the status of the channel (if it is a candidate call or not);
- tprev, the timestamp of the last packet received belonging to the channel;
- tstart, the timestamp of the packet which cause a transition of call from false to true.

Every time a packet belonging to a channel is captured it affects some of the attributes of the channel. If the values of the parameters ipdact and psact are compatible with the values determined when defining the model, we can suppose there is a Skype call going between the source and the destination. If these conditions remain valid for enough time then the presence is confirmed otherwise it is discarded. The decision is taken according to the FSM depicted in figure 3 where a channel in state A is NOT part

of a Skype call, a channel in state B MAY BE part of a Skype call and a channel in state C IS part of a Skype call. The transitions are driven by the following events:

- from A to B: Incoming packets made ipdact and psact match the statistical model of a Skype call;
- from B to A: ipdact and psact requirements are no more met.
- from B to C: the channel has been in state B for no less than 1 second.
- from C to A: ipdact and psact requirements are no more met or too long a time has passed since the last packet has arrived.

We tested the effectiveness of our software with several experiments that involved both calls with both endpoints inside the LAN and calls with endpoints connected through the Internet. Every call was detected although, in the case of calls over the Internet, we experienced loss of detection when the audio quality of the call was severely degraded by traffic congestion. The additional symmetry constraint made our software resilient to false positives and webcasts were always not identified as calls.

4. A SKYPE STEALTHER

As described in the previous section we have been able to develop a program that can identify a Skype call. However, this parametrical identification is fragile. The key part of the identification process is how much the statistical features of a channel match those of a specific statistical model, thus it is easy to put up a smoke screen that prevents the actual identification of a stream. In fact, we just need to re-shape the statistical feature of streams to disguise it completely.

It is quite obvious that we need to "poison the streams" without introducing excessive packet latency or significantly amplifying the packet jitter. In fact, such artifacts would not just prevent call identification but will also heavily degrade the call quality. Actually, there are two possible approaches:

1. We can perform some limited manipulations to the packet latency and jitter without changing the amount of bandwidth occupied so that the streams do not fit any-longer into the statistical model we designed but they still preserve an acceptable audio quality;
2. We can maintain the original streams quality in term of latency and jitter for the actual audio data but we can manipulate the overall stream characteristics inserting padding in some packets and adding ballast packets to the streams at the cost of an increased bandwidth.

The first approach exploits the fact that a limited latency in the communication, such as the 250 ms commonly experienced in satellite phone calls, may be annoying but does not prevent efficient communication. Let's assume, without loss of generality, to have an underlying audio stream coded with the iLPC codec. The nominal IPD of the original stream is equal to 20 or 30 ms. Let's assume, again with no loss of generality, that we are currently using the 20 ms mode. Thus it is possible to group packets in groups of three to five and send them together. This will scramble the IPD value making it significantly unstable and, in any case, with an average value that can vary from 60 ms to 100 ms, definitely out of the range the detector is looking for. This approach has, however, a significant drawback. In fact Internet based telephony is prone to call quality degradation due to traffic congestion; thus it is quite possible that the combination of the stealth induced quality reduction and the accidental reduction due to traffic congestion sum up to an unacceptable call quality. For this reason we decided to adopt the second approach to implement our solution.

To achieve a good level of stealth our action will be twofold: *i)* we will introduce padding in each packet to make packet size larger than expected and variable and *ii)* we will introduce in the stream ballast packet to make inter packet delay shorter and variable.

In table 1 we show that the measured average for a packet size is about 155 byte while the average inter packet delay is about 35 ms, these value identify an average bit rate of about 36 kbps. Thus, if we poison each packet with an amount of ballast varying from 30 to 60 bytes we obtain average packet sizes ranging from 185 to 215 bytes, well above the range that the detector identifies as Skype calls. At the same time, an average packet size of 200 bytes with an average inter packet delay identifies a bit rate of about 46 kbps. Thus our stream poisoning imposes a bit rate increase of less than 30%. Let's now consider the inter packet delay. As we do not want to introduce any kind of degradation of the call quality we cannot delay any packet, thus we can only inject additional ballast packets. If we inject very small packets we introduce a limited amount of additional bandwidth consumption while we can completely change the average inter packet delay. As an example, if, during the original 35 ms inter packet period we inject one or two ballast packets, we can obtain an average value of inter packet delay ranging from 12 to 17 ms. These values are completely out of the range the detector is capable of recognizing as produced by a Skype call, thus our poisoning is effective. As it is not possible to send IP packets shorter than 40 bytes, we now calculate to what extent our stream poisoning increases the stream bit rate. In a 35 ms period we will send 155 bytes of actual data and 80 bytes (two 40 bytes packets) of ballast. These values sum-up to 235 bytes in a 35 ms period. Thus we generate a 54 kbps packet stream. Compared to the original 36 kbps data stream we have increased the required bandwidth by 50%. Actually, the injection of small ballast packet ha also the effect of changing the average packet

size. As a matter of fact, if we consider to send groups of three packets one averaging 155 bytes and the other two of 40 bytes we obtain an average packet size of about 78 bytes, a value that is by far lower than the threshold the detector uses to identify Skype calls. If we consider switching the poisoning technique several times during a Skype call we obtain a stream with statistical features that do not match the model of a Skype call.

To implement the stealther we encountered a major problem: Skype is a closed source program thus we had no way to apply any change to its code. Without a way to manipulate the code Skype uses to send a packet, it was not possible to integrate and test the pad-based poisoning, however, we found a way to test the actual efficacy of ballast packet injection. To do this, we had to introduce a change in the way the detector implements the concept of channels. In fact, we removed the capability of the detector to distinguish the source and destination ports of the stream and we performed the tests considering all the traffic between two IP addresses as a single channel. Thus our stealther ended up as a simple packet injector with configurable frequency and patterns. When the only traffic between the two end points was the skype call, the detector was capable of recognizing it with the same precision described in section 3. When we started injecting packets with the rate and pattern described in the previous section the Skype call was immediately discarded by the detector as other traffic.

5. CONCLUDING REMARKS

Voice over IP is rapidly becoming a major actor in the scenario of telecommunication. This has the potential to generate large cost reduction for telecom operators and industries without reducing but, on the contrary, enhancing the services available. Nonetheless, this technological shift has some risk, as a matter of fact network managers and law enforcement agencies can have an extremely hard time at trying to consistently monitor VoIP connections.

Past works show that it is possible to identify VoIP streams by carefully analyzing their statistical characteristics; in this paper we have demonstrated that a rigorously designed statistical model allows achieving a very high level of reliability in the recognition of the calls generated by one of the most widely adopted VoIP clients, namely Skype. Our proof of concept software implementation was able of identifying Skype calls in seconds with larger than 95% accuracy while posing a very limited computational burden on the hosting machine. Nonetheless, the call identification methodology based on the statistical characteristics of the stream is extremely fragile, in fact it is quite easy to "poison the streams" so that their statistical behavior is significantly distant from the expected one. In the last part of this paper we studied how it is possible to mask the statistical features of a Skype stream without compromising the perceived voice quality and without requiring a large increase in bandwidth availability. Our proof of concept software demonstrated the feasibility of this approach although the closed source nature of the Skype client prevented us from actually integrating this stealth module into it. Such integration may happen in future works.

6. REFERENCES

[1] B. Goode. Voice Over Internet Protocol (VoIP). Proceedings of IEEE, 90(9):1495⊥ 1517, September 2002.

[2] S.L. Garfinkel. VoIP and Skype security, January 2005.

[3] T. Okabe, T. Kitamura, and T. Shizuno. Statistical traffic identification method based on flow-level behavior for fair VoIP service. In Proceedings of the 1st IEEE Workshop onVoIP Management and Security, 2006, pages 35⊥ 40. IEEE Press, April 2006.

[4] K. Suh, D. R. Figueiredo, J. Kurose, and D. Towsley. Characterizing and detecting skype-relayed traffic, Proc. of IEEE InfoCom 2006, Barcelona, April 23-29.

[5] Skype. http://www.skype.com.

[6] S. A. Baset and H. Schulzrinne. An analysis of the Skype peer-to-peer Internet telephony protocol. Technical Report CUCS-039-04, Columbia University, September 2004.

[7] S. Ehlert and S. Petgang. Analysis and signature of Skype VoIP session traffic, Fraunhofer FOKUS Technical Report NGNI-SKYPE-06b July 2006.

[8] S. Guha, N. Daswani, and R. Jain. An experimental study of the Skype peer-to-peer VoIP system, Proc. of IPTPS 2006.

[9] iLBC codec. http://www.globalipsound.com/pdf/gips_iLBC.pdf.

[10] iSAC codec. http://www.globalipsound.com/pdf/gips_iSAC.pdf.

[11] T. Berson. Skype security evaluation. Technical Report ALR-2005-031, Anagram Laboratories, October 2005.

[12] A. Biernacki. VoIP Source Model based on the Hyperexponential Distribution. In Transactions on Engineering, Computing and Technology, volume 11, pages 202⊥ 206. World Enformatika Society, February 2006.

[13] A. Estepa, R. Estepa, and J. Vozmediano. A new approach for VoIP traffic characterization. IEEE Communications Letters, 8(10):644⊥ 646, October 2004.

[14] Wireshark. http://www.wireshark.org.

[15] Libpcap. http://sourceforge.net/projects/libpcap.