International Journal of Advanced Smart Convergence Vol.9 No.4 209-214 (2020)
http://dx.doi.org/10.7236/IJASC.2020.9.4.209

IJASC 20-4-28

Implementation of Extracting Specific Information by Sniffing Voice Packet in VoIP

Dong-Geon Lee[1], WoongChul Choi[2]

[1]*B.S. Student, Dept. of Computer Science, Kwangwoon University, Korea*
[2]*Professor, Dept. of Computer Science, Kwangwoon University, Korea*
wchoi@kw.ac.kr

Abstract

VoIP technology has been widely used for exchanging voice or image data through IP networks. VoIP technology, often called Internet Telephony, sends and receives voice data over the RTP protocol during the session. However, there is an exposition risk in the voice data in VoIP using the RTP protocol, where the RTP protocol does not have a specification for encryption of the original data. We implement programs that can extract meaningful information from the user's dialogue. The meaningful information means the information that the program user wants to obtain. In order to do that, our implementation has two parts. One is the client part, which inputs the keyword of the information that the user wants to obtain, and the other is the server part, which sniffs and performs the speech recognition process. We use the Google Speech API from Google Cloud, which uses machine learning in the speech recognition process. Finally, we discuss the usability and the limitations of the implementation with the example.

Keywords: *Audio, VoIP, RTP, Sniffing, STT, Speech Recognition*

1. Introduction

Most of the users' audio data of Voice-over-IP(VoIP) comes from voice calls, i.e. Internet telephony, and audio data usually contain sensitive information of users. While many VoIP services provide service with security guarantee by a user authentication process and an encryption process, there are still many VoIP services that are using the Real-time Transport Protocol(RTP), which does not have a specification for encryption of the original data, therefore, an attacker simply snatches a packet using several sniffing tools and consequently, the VoIP phone user's call content may be exposed to the attacker. In [1], we implemented a VoIP-specific sniffing tool to sniff the VoIP packets and showed the security issues in VoIP. In this paper, we implement tools for extracting meaningful information from the user's dialogue. The meaningful information means the information that the program user wants to obtain.

In order to do that, our implementation has two parts. One is the client part, which inputs the keyword of the information that the user wants to obtain, and the other is the server part, which sniffs and performs the

Manuscript Received: December 2, 2020 / Revised: December. 6, 2020 / Accepted: December. 7, 2020
Corresponding Author: wchoi@kw.ac.kr
Tel: +82-2-940-5582, Fax: +82-2-943-5602
Professor, Department of Computer Science, KwangWoon University, Korea

speech recognition process. We use the Google Speech API from Google Cloud, which uses machine learning in the speech recognition process [2].

We organize the paper as follows. First, we briefly overview the packet format and the operation of VoIP using RTP and then how to save the payloads of the VoIP packets in a file for further processing to extract the meaningful information. Finally, our implementation is explained in detail.

2. Backgrounds

In order to obtain data from the packets obtained by sniffing using our implemented sniffing tool, it is necessary to know what information is contained in the packet. The following is the packet format of a RTP. As mentioned earlier, the RTP protocol is mostly used in the UDP protocol and has the structure shown in Figure 1.

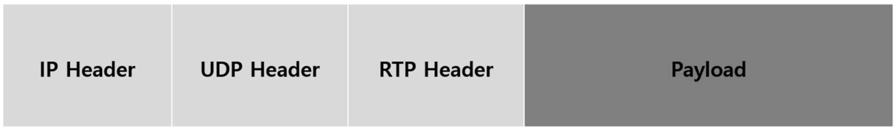
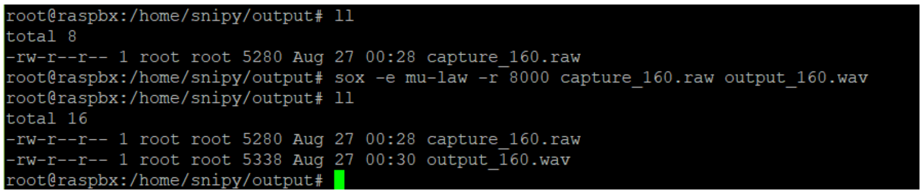


Figure 1. RTP Packet Format

As shown in the figure, the RTP packet has an IP header, a UDP header, and an RTP header, and there is a substantial audio data sample in the payload at the end of the packet. However, since the audio samples in the payload are transmitted in byte format, it is impossible to convert them into voice data simply by using the samples. Therefore, we want to analyze the structure of RTP header because we need additional information in the header. Figure 1 shows the structure of the RTP header.

The data in the Payload field are saved in a raw file. Then, the generated raw file can be converted into several audio conversion programs. We converted the raw file into a wav file using a program called Sound eXchange (Sox) [3-5].

The Figure 2 shows the process of converting a RAW file into a WAVE file using the command 'sox -e mu-law -r 8000 capture_160.raw output_160.wav'. This command creates the output_160.wav file by converting the capture_160.raw file to the mu-law (PCMU) method and the 8000 sampling rate [5-8].



```
root@raspbx:/home/snipy/output# ll
total 8
-rw-r--r-- 1 root root 5280 Aug 27 00:28 capture_160.raw
root@raspbx:/home/snipy/output# sox -e mu-law -r 8000 capture_160.raw output_160.wav
root@raspbx:/home/snipy/output# ll
total 16
-rw-r--r-- 1 root root 5280 Aug 27 00:28 capture_160.raw
-rw-r--r-- 1 root root 5338 Aug 27 00:30 output_160.wav
root@raspbx:/home/snipy/output#
```

Figure 2. Converting the RAW file stored in payload to WAV file using Sox

3. Tool for Speech Recognition

3.1 Speech Recognition Server & Client

Now, we show the implementation of a program that recognizes speech from packets obtained using the implemented sniffing tool and extracts information by inputting specific keywords. The sniffing tool implemented so far will be a server program that sniffs the voice of clients using SIP phones. The sniffing server receives the keyword from the client and transmits the information matching the keyword back to the client. Conversely, the client program receives the keyword directly from the user and sends it to the server and waits for the matching information to be returned from the server. When the information is returned, the client program plays the returned audio data and outputs a transcript. The following Figure 3 shows the interaction between the server and the client.

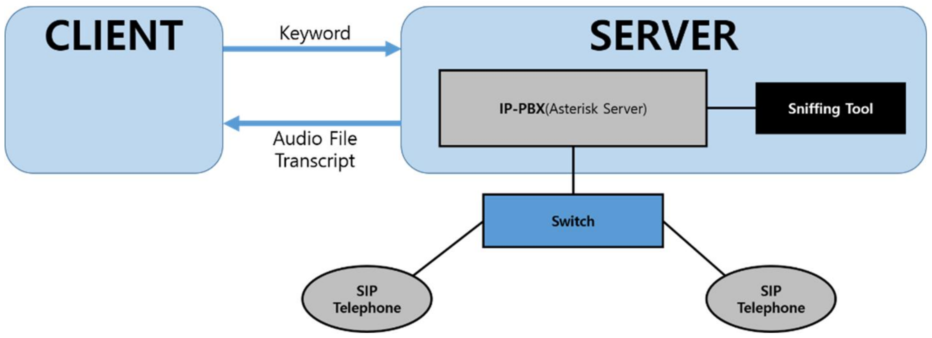
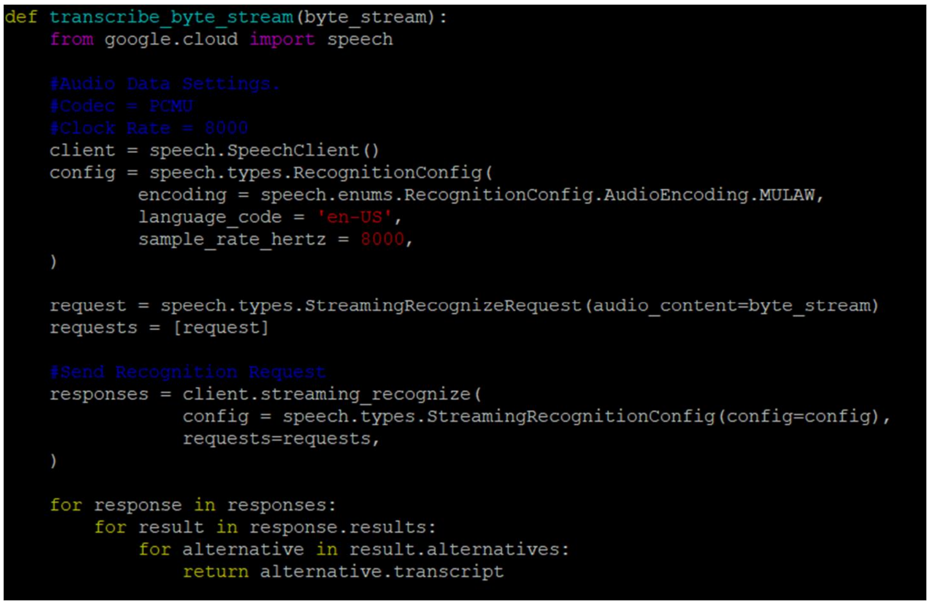


Figure 3. The Topology of Recognition Server & Client

3.2 Recognizing Audio at Server

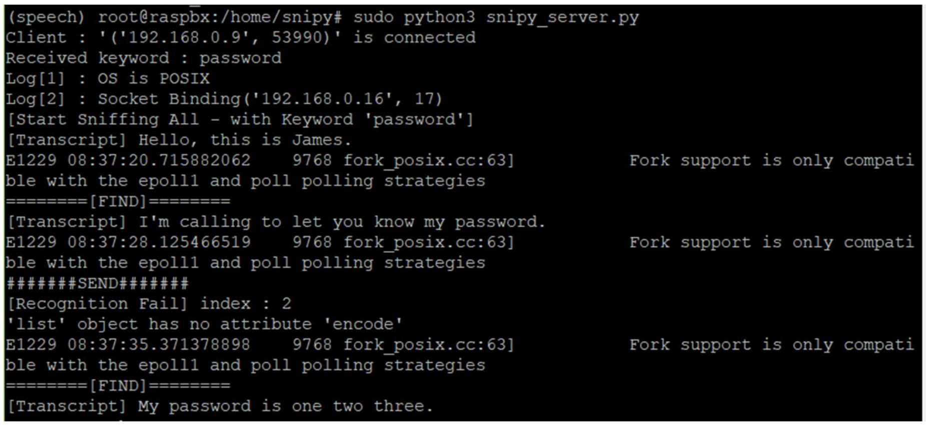
First, we want to show the process of speech recognition in the server program. We use the Google Speech API to do speech recognition. The Google Speech API is an Speech-To-Text (STT) library that uses machine learning to convert speech into text [10]. We will recognize the voice using this library and compare it with the input keyword to extract specific information. The Figure 4 shows the implemented code [9, 11].
Since the user's voice is inputted in real time, the voice recognition should also be performed in real time. Therefore, our program decided to request speech recognition every 5 seconds. That is, the voice data that we recognize at one time is the length corresponding to 250 packets since it is the data encoded by the PCMU method. So, 250 payload values are combined into one byte string and then the recognition is requested. When the recognition is done and the transcript is returned as the result value, it is compared with the inputted keyword. If a transcript matching the keyword is found, the RAW file is created by storing the payload in a total of 750 packets including the corresponding interval and the RTP packet corresponding to 5 seconds before and after.



```
def transcribe_byte_stream(byte_stream):
    from google.cloud import speech

    #Audio Data Settings.
    #Codec = PCMU
    #Clock Rate = 8000
    client = speech.SpeechClient()
    config = speech.types.RecognitionConfig(
            encoding = speech.enums.RecognitionConfig.AudioEncoding.MULAW,
            language_code = 'en-US',
            sample_rate_hertz = 8000,
    )

    request = speech.types.StreamingRecognizeRequest(audio_content=byte_stream)
    requests = [request]

    #Send Recognition Request
    responses = client.streaming_recognize(
                config = speech.types.StreamingRecognitionConfig(config=config),
                requests=requests,
    )

    for response in responses:
        for result in response.results:
            for alternative in result.alternatives:
                return alternative.transcript
```

Figure 4. The code that requests Speech Recognition

The generated RAW file is converted into a WAVE file using a Sox program [11]. The server program then sends the generated WAVE file and transcript to the client program. The Figure 5 show the result of speech recognition on server side.



```
(speech) root@raspbx:/home/snipy# sudo python3 snipy_server.py
Client : '('192.168.0.9', 53990)' is connected
Received keyword : password
Log[1] : OS is POSIX
Log[2] : Socket Binding('192.168.0.16', 17)
[Start Sniffing All - with Keyword 'password']
[Transcript] Hello, this is James.
E1229 08:37:20.715882062    9768 fork_posix.cc:63]          Fork support is only compati
ble with the epoll1 and poll polling strategies
========[FIND]========
[Transcript] I'm calling to let you know my password.
E1229 08:37:28.125466519    9768 fork_posix.cc:63]          Fork support is only compati
ble with the epoll1 and poll polling strategies
#######SEND#######
[Recognition Fail] index : 2
'list' object has no attribute 'encode'
E1229 08:37:35.371378898    9768 fork_posix.cc:63]          Fork support is only compati
ble with the epoll1 and poll polling strategies
========[FIND]========
[Transcript] My password is one two three.
```

Figure 5. Result of Speech Recognition and Comparing with keywords

3.3 Receiving Matched Data at Client

The client program accesses a server program that performs sniffing, transmits keywords related to desired information, and waits until receiving the corresponding information. The client program was implemented using JAVA and was created using eclipse on Window 10. The Figure 6 shows Client program window.

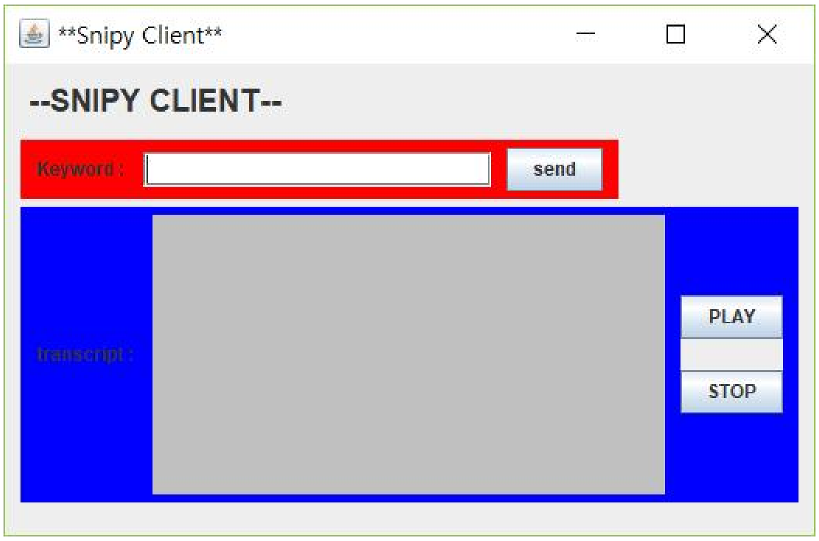


Figure 6. Client program window that receives keywords

4. Closing Remarks

After implementing the sniffing tool, we also implemented a program that sniffed users' VoIP phones and extracted specific information from the sniffed voice data using the Google Speech API. Our program has the advantage of extracting the requested information only among the meaningful information in the conversation. Taking the advantages of our program, a user, such as bank, can extract the customer's information from the conversation when the customer speaks with the agent and retrieve customer information to provide faster service. Also, when using ARS service over the phone, member authentication may be required. In this case, the keypad of the telephone is pressed to input information from the user. However, using our program can reduce the hassle of the certification process by extracting customer information from the conversation. In fact, “Lotte Home Shopping”, the company in Korea to adopt the voice recognition method in ARS for the first time in Korea in 2018, showed an average of 60 seconds shortened. This is not only advantageous for the customers who use the service, but also for the service provider. Similarly, when a variety of information, such as membership, needs to be inputted, applying our program allows the user to dial the relevant information without having to enter multiple fields, so the necessary information is automatically entered.

On the other hand, there are limitations in implementation as well. Since our program requires speech recognition, if the quality of the speech is poor due to the mixed noise, the speech may not be recognized or may be misrecognized and the incorrect information may be extracted. Therefore, if the speech recognition function is not completely guaranteed, even if the information is extracted through the speech recognition, the person who uses the extracted information needs to be careful to confirm the information again. Such limitations need to investigate further in the following research to improve the current implementations.

Acknowledgement

The present research has been conducted by the Research Grant of KwangWoon University in 2018.

References

[1] Dong-Geun Lee and WoongChul Choi, "Security Exposure of RTP packet in VoIP", International Journal of Internet, Broadcasting and Communications, Vol. 11, No.3, pp. 59-63, August 2019
DOI: http://dx.doi.org/10.7236/IJIBC.2019.11.3.59

[2] Google Speech API, https://cloud.google.com/speech-to-text

[3] H. Schulzrinne and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control.", RFC3551, The Internet Society, 2003

[4] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications.", RFC3550, The Internet Society, 2003

[5] Malcolm Davenport, "RTP Packetization", https://wiki.asterisk.org/wiki/display/AST/RTP+Packetization

[6] Patrick Park, "Voice over IP Security", Cisco Press, 2009

[7] Himanshu Dwivedi, "Hacking VoIP", No Starch Press, 2009.

[8] Laura Chappell, " Wireshark Network Analysis: The Official Wireshark Certified ", Protocol Analysis Institute, 2010

[9] Google Speech API, Python Client for Cloud Speech API, https://googleapis.github.io/google-cloud-python/latest/speech/index.html

[10] Arvind Ravulavaru, "Google Cloud AI Service Quick Start Guide: Build intelligent applications with Google Cloud AI Service", Packt Publishing Ltd, May 2018

[11] Giuseppe Ciaburro, Kishore Ayyadevara, and Alexis Perrier, "Hands-On Machine Learning on Google Cloud Platform: Implementing smart and efficient analytics using Cloud ML Engine", Packt Publishing Ltd, April 2018