International Journal of Internet, Broadcasting and Communication Vol.11 No.3 59-63 (2019)
http://dx.doi.org/10.7236/IJIBC.2019.11.3.59

IJIBC 19-3-8

Security Exposure of RTP packet in VoIP

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

[1,2] *Dept. of Computer Science, KwangWoon University, Korea*
[1]*matth1996@kw.ac.kr,* [2]*wchoi@kw.ac.kr*

Abstract

VoIP technology is a technology for exchanging voice or video data through IP network. Various protocols are used for this technique, in particular, RTP(Real-time Transport Protocol)[1] protocol is used to exchange voice data. In recent years, with the development of communication technology, there has been an increasing tendency of services such as "Kakao Voice Talk" to exchange voice and video data through IP network[2]. Most of these services provide a service with security guarantee by a user authentication process and an encryption process. However, RTP protocol does not require encryption when transmitting data. Therefore, there is an exposition risk in the voice data using RTP protocol. We will present the risk of the situation where packets are sniffed in VoIP(Voice over IP)[3] communication using RTP protocol. To this end, we configured a VoIP telephone network, applied our own sniffing tool, and analyzed the sniffed packets to show the risk that users' data could be exposed unprotected.

Keywords: *RTP, sniffing, Packet analysis, VoIP*

1. Introduction

VoIP technology has been widely used for exchanging voice or image data through IP network. VoIP telephony, often called Internet telephony, sends and receives voice data over the RTP protocol during users' conversation. In addition, with the recent development of communication technology, the service using VoIP such as "Kakao Voice Talk" is increasing. VoIP is a technology for transmitting audio data of users through the Internet.

Most of the users' audio data comes from voice calls (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 RTP protocol and the RTP protocol 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 fact, experts have developed SRTP(Secure Real-time Transport Protocol)[4] protocols that recognize this problem and apply encryption to the Payload field to solve the problem.

Manuscript Received: June. 9, 2019 / Revised: June. 16, 2019 / Accepted: June. 20, 2019
*Corresponding Author: wchoi@kw.ac.kr
Tel: +82-2-940-5582, Fax: +82-2-940-5582
Dept. of Computer Science, KwangWoon University, Korea

However, many developers using RTP do not apply SRTP in most cases because of the performance problems of VoIP phones, therefore, there is an exposition risk in the voice data using the RTP protocol, which can be checked when sniffing packets that are being transmitted are performed.

To show this, we will design and implement a VoIP where audio data are exchanged. We also create a VoIP-specific sniffing tool to show this risk of exposition in audio data transmission. That is, we implement a program that sniffs users' VoIP phones and extracts specific information from the sniffed voice data. Our developed tool will be applied to the audio transmission network that is also implemented. In addition, we will reconstruct the audio data through captured packet analysis and diagnose security vulnerabilities and seek possible solutions.

2. VoIP Network Configuration

We designed and implemented a VoIP network to sniff voice packets as shown in Figure 1. The key part of our implementation is the equipment called IP-PBX. A PBX is a device such as a telephone network exchanger, and is a device that acts as an exchange agent through the Internet. We chose the hardware to be the IP-PBX as 'Raspberry Pi'. In addition, a SIP(Session Initiation Protocol) server is required to function as an IP-PBX. We installed SIP server in 'Raspberry Pi' using open source Asterisk Server and used FreePBX to provide Asterisk's web-based GUI to manage server more efficiently. The version information of the open source that we use is as follows.
- Raspbian Stretch LITE (Raspberry Pi OS)
- Asterisk 13.20.0
- FreePBX 14.0.3.2

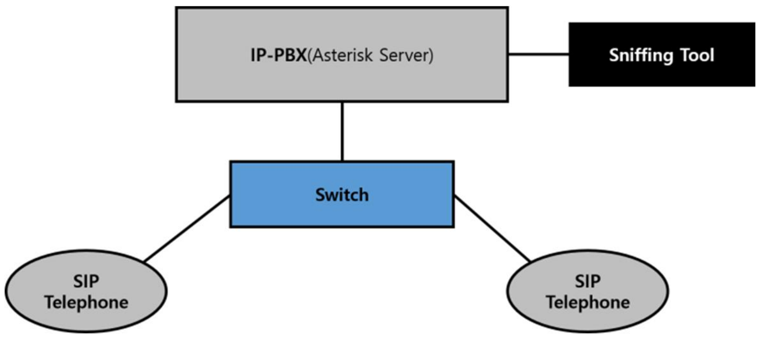


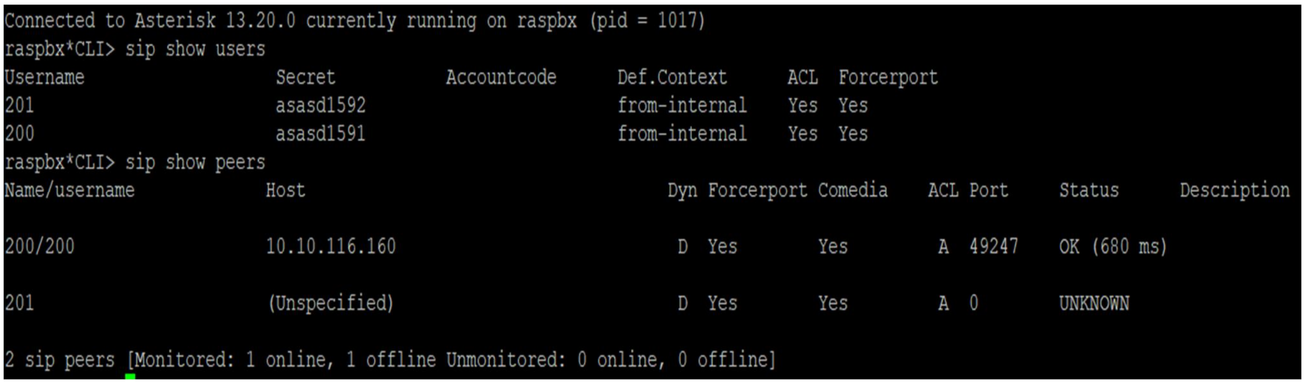
Figure 1. Network diagram to implement.

Once you have finished implementing the server, you will need to register two telephones to be used as clients in the server. To register a phone, you must first access the server page. If the page is successfully connected, log in to the admin account. Then, go to [Applications] → [Extensions] tab. When you enter the [Extension] tab, you will be presented with a window to register users. At this time, you can create users for each channel driver. The channel driver is used to convert the incoming signal to the core of the server when communicating with a device (SIP Telephone) outside the Asterisk server[5]. Since our research is not a channel driver-driven study, we have created a user using the Chan_sip driver, which is the earliest basic stack. Table 1 shows the information of the created user.

Table 1. User Information registered in SIP Server.

	Extension Number	Display Name	Channel Driver	Port
USER 1	200	Tester1	Chan_sip	5060
USER 2	201	Tester2	Chan_sip	5060

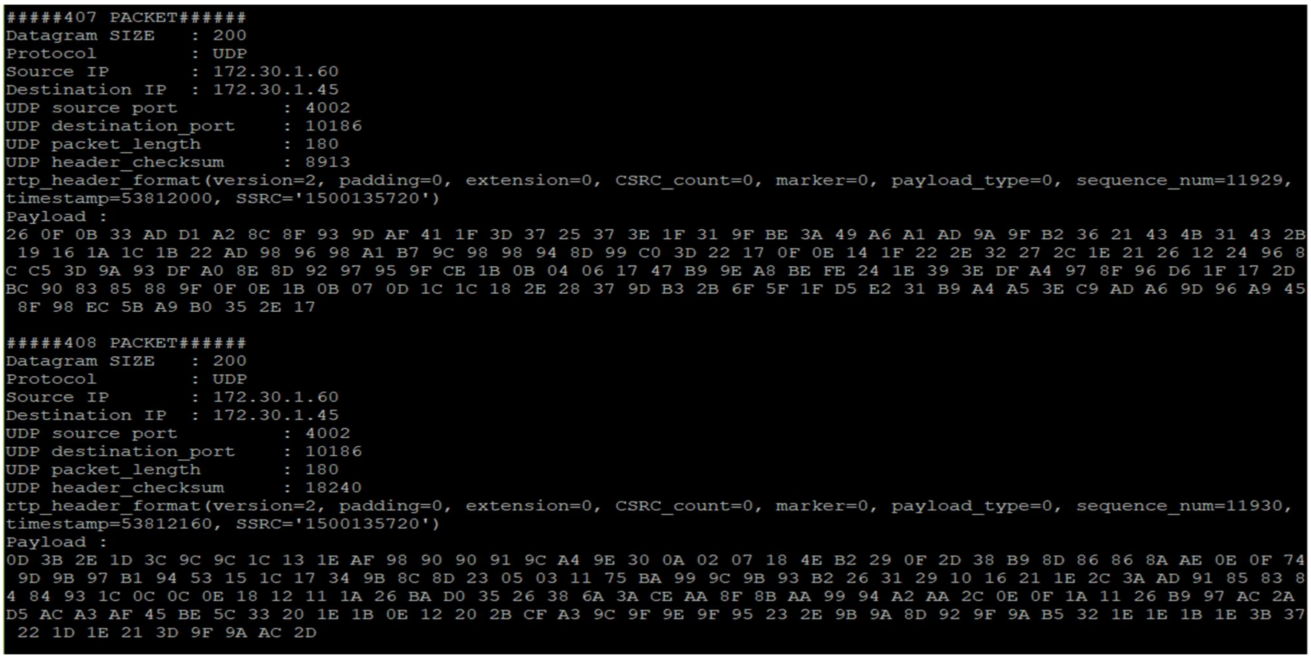
After you have registered users on the server, you must also configure them on the Client side to connect to the server next. In our study, Client corresponds to SIP Telephone. We decided to install 'CSipSimple', an open source Android application made by Regis Montoya, on two Android smartphones to play the role of SIP Telephone. After installing the application, set up the client using the information registered in the server.



```
Connected to Asterisk 13.20.0 currently running on raspbx (pid = 1017)
raspbx*CLI> sip show users
Username                  Secret          Accountcode      Def.Context      ACL  Forcerport
201                       asasd1592                        from-internal    Yes  Yes
200                       asasd1591                        from-internal    Yes  Yes
raspbx*CLI> sip show peers
Name/username             Host                                    Dyn Forcerport Comedia    ACL Port     Status      Description

200/200                   10.10.116.160                            D  Yes        Yes         A  49247    OK (680 ms)

201                       (Unspecified)                            D  Yes        Yes         A  0        UNKNOWN

2 sip peers [Monitored: 1 online, 1 offline Unmonitored: 0 online, 0 offline]
```

Figure 2. Registered user information and Server connection status.

3. RTP Packet Analyzation



```
#####407 PACKET######
Datagram SIZE   : 200
Protocol        : UDP
Source IP       : 172.30.1.60
Destination IP  : 172.30.1.45
UDP source port          : 4002
UDP destination_port     : 10186
UDP packet_length        : 180
UDP header_checksum      : 8913
rtp_header_format(version=2, padding=0, extension=0, CSRC_count=0, marker=0, payload_type=0, sequence_num=11929,
timestamp=53812000, SSRC='1500135720')
Payload :
26 0F 0B 33 AD D1 A2 8C 8F 93 9D AF 41 1F 3D 37 25 37 3E 1F 31 9F BE 3A 49 A6 A1 AD 9A 9F B2 36 21 43 4B 31 43 2B
 19 16 1A 1C 1B 22 AD 98 96 98 A1 B7 9C 98 98 94 8D 99 C0 3D 22 17 0F 0E 14 1F 22 2E 32 27 2C 1E 21 26 12 24 96 8
C C5 3D 9A 93 DF A0 8E 8D 92 97 95 9F CE 1B 0B 04 06 17 47 B9 9E A8 BE FE 24 1E 39 3E DF A4 97 8F 96 D6 1F 17 2D
BC 90 83 85 88 9F 0F 0E 1B 0B 07 0D 1C 1C 18 2E 28 37 9D B3 2B 6F 5F 1F D5 E2 31 B9 A4 A5 3E C9 AD A6 9D 96 A9 45
 8F 98 EC 5B A9 B0 35 2E 17

#####408 PACKET######
Datagram SIZE   : 200
Protocol        : UDP
Source IP       : 172.30.1.60
Destination IP  : 172.30.1.45
UDP source port          : 4002
UDP destination_port     : 10186
UDP packet_length        : 180
UDP header_checksum      : 18240
rtp_header_format(version=2, padding=0, extension=0, CSRC_count=0, marker=0, payload_type=0, sequence_num=11930,
timestamp=53812160, SSRC='1500135720')
Payload :
0D 3B 2E 1D 3C 9C 9C 1C 13 1E AF 98 90 90 91 9C A4 9E 30 0A 02 07 18 4E B2 29 0F 2D 38 B9 8D 86 86 8A AE 0E 0F 74
 9D 9B 97 B1 94 53 15 1C 17 34 9B 8C 8D 23 05 03 11 75 BA 99 9C 9B 93 B2 26 31 29 10 16 21 1E 2C 3A AD 91 85 83 8
4 84 93 1C 0C 0C 0E 18 12 11 1A 26 BA D0 35 26 38 6A 3A CE AA 8F 8B AA 99 94 A2 AA 2C 0E 0F 1A 11 26 B9 97 AC 2A
D5 AC A3 AF 45 BE 5C 33 20 1E 1B 0E 12 20 2B CF A3 9C 9F 9E 9F 95 23 2E 9B 9A 8D 92 9F 9A B5 32 1E 1E 1B 1E 3B 37
 22 1D 1E 21 3D 9F 9A AC 2D
```

Figure 3. Captured RTP packet.

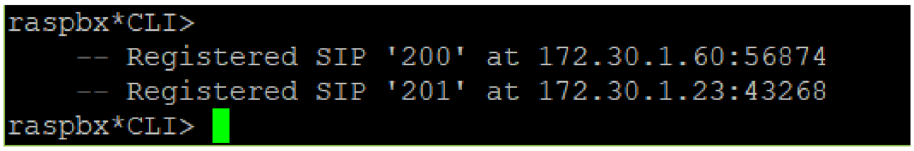
When a caller makes a call, the SIP message is delivered through the server and the other party's phone rings. When the other party receives the call, the SIP message is transmitted again to the caller, and the RTP session is started and the audio data is transmitted in real time. When the call is terminated, a SIP message is also transmitted and the caller informs the other party that the call has ended. Various protocols are used for such a call.

We wrote our own program to capture RTP packets to sniff packets. The language used in the program was Python version 3.5.3 and written using Vim on Raspbian Stretch LITE.

Figure 3 shows some of the screens capturing RTP packets using our program since the start of RTP Session. The contents of the above captured packets are as follows.

- Source IP :172.30.1.60, Destination IP:172.30.1.45

When a voice or video is sampled and generated as a packet on the user's telephone, the packet is transmitted to the other telephone through the server. When the phone connects to the server, the asterisk CLI window will show the information shown in Figure 4.



```
raspbx*CLI>
    -- Registered SIP '200' at 172.30.1.60:56874
    -- Registered SIP '201' at 172.30.1.23:43268
raspbx*CLI>
```

Figure 4. The log of the client connecting to the server.

We can see from this that data is entered from a user with the number 200 and passed to the server.

- Payload_type=0

The value in this field is of 0 means that using the audio coding method of PCMU, defined in ITU-T Recommendation G.711. PCMU has a transmission speed of DS-O class (64kbps) in a way that uses μ-law (mu-law) among two encoding methods of G711. Also, the PCMU method has a sampling data of 20ms per packet and a sampling rate (clock rate) value of 8000. Therefore, this method transmits 50 packets of data per second and transmits 8000 sampled data (sampling data). Therefore, it can be seen that one packet contains 160 sampled data in the payload field, and it can be seen that the sampled data (= 1 octet) is generated by sampling the audio every 125 μsec. In other words, the value of this field is important information for conversion into Audio file[6] [7].

- Sequence_number=11929, 11930

This field has no meaning when viewed in a single packet and is meaningful when compared to the Sequence number field value of another RTP packet before and after. Since this value increases by 1 in the order of the packets, the relative temporal position of the packet can be detected. Since the previous packet has the value of 11929 and the subsequent packet has the value of 11930, it can be seen that the two packets are forward continuous packets.

- Timestamp=53812000, 53812160

The value of this field also shows the temporal relationship when compared to the preceding and following packets. This value is affected by the value of the Payload Type field because the amount of increase depends on the sampling period of each codec. In our case, since we use the PCMU method, the sampling period is 160, and the value of the Timestamp is also increased by 160 times.

- Payload

As mentioned above, there are 160 sampling data in the payload of one packet. In fact, we can see that there are 160 data in the picture. The data present in this field are actual audio or video data. However, since these data are transmitted unencrypted, there is a risk that information will be exposed.

4. Conclusion

In our study, we attempted sniffing in an implemented VoIP network to capture RTP packets. We could get the user's voice data easily by ONLY receiving the packet and getting the payload through header analysis. Even if an attacker simply snatches a packet using a method such as spoofing, the VoIP phone user's call content may be exposed to the attacker. This problem is because there is no authentication process in the RTP protocol and packets are transmitted and received without applying the packet encryption algorithm. In fact, experts have developed SRTP protocols that recognize this problem and apply encryption to the Payload field to solve the problem. However, many developers using RTP do not apply SRTP in most cases because of the performance problems of VoIP phones. The Asterisk Server we used in our research is also in a situation where users' audio data can be easily exposed to an attacker by using an unencrypted RTP protocol. Of course, VoIP also uses other protocols such as SIP, so it is also important to enhance the security of SIP communication with TLS method, but it is necessary to protect the actual audio data by encrypting it[8] [9] [10][11].

Acknowledgement

The present Research has been conducted by the Research Grant of KwangWoon University in 2017.

References

[1] RTP: A Transport Protocol for Real-Time Applications, https://datatracker.ietf.org/doc/rfc3550/

[2] KakaoTalk, accessed on Aug 26, 2018, https://www.kakaocorp.com/service/KakaoTalk

[3] VoIP Voice over IP, Voice over Internet Protocol, Internet Telephony , http://www.ktword.co.kr/abbr_view.php?m_temp1=1120

[4] The Secure Real-time Transport Protocol (SRTP), https://datatracker.ietf.org/doc/rfc3711/

[5] Malcolm Davenport, "Channel Driver Modules", Asterisk Wiki, last modified by Rusty Newton on Jul 10, 2014, accessed on Aug 26, 2018, https://wiki.asterisk.org/wiki/display/AST/Channe l+Driver+Modules

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

[7] Malcolm Davenport , "RTP Packetization", Asterisk Wiki, last modified by Matt Jordan on Aug 08, 2012, accessed on Aug 26, 2018, https://wiki.asterisk.org/wiki/display/AST/RTP+P acketization

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

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

[10] Laura Chappell, "Wireshark Network Analysis: The Official Wireshark Certified", Protocol Analysis Institu te, 2010

[11] Theo Zourzouvillys, Eric Rescorla, "An Introduction to Standards-Based VoIP: SIP, RTP, and Friends", IE EE Internet Computing, Vol. 14, Issue 2, pp. 69-73, March-April 2010. DOI: https://doi.org/10.1109/MIC.2 010.31