디지털산업정보학회 논문지 제14권 제4호-2018년 12월
http://dx.doi.org/10.17662/ksdim.2018.14.4.117

VoIP 스니핑을 통한 특정정보 탈취 위험성에 관한 연구

이 동 건* · 최 웅 철**

A study on the risk of taking out specific information by VoIP sniffing technique

Lee Donggeon · Choi Woongchul

〈Abstract〉

Recently, VoIP technology is widely used in our daily life. Even VoIP has become a technology that can be easily accessed from services such as home phone as well as KakaoTalk.[1] Most of these Internet telephones use the RTP protocol. However, there is a vulnerability that the audio data of users can be intercepted through packet sniffing in the RTP protocol. So we want to create a tool to check the security level of a VoIP network using the RTP protocol. To do so, we capture data packet from and to these VoIP networks. For this purpose, we first configure a virtual VoIP network using Raspberry Pi and show the security vulnerability by applying our developed sniffing tool to the VoIP network. We will then analyze the captured packets and extract meaningful information from the analyzed data using the Google Speech API. Finally, we will address the causes of these vulnerabilities and possible solutions to address them.

Key Words : VoIP, RTP, SIP, Security, Sniffing

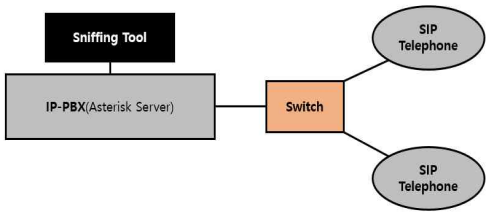
Ⅰ. 서론

VoIP기술은 사용자들의 음성데이터를 IP망 을 통하여 전달하는 기술이다. 오디오데이터는 대부분 사용자간의 음성통화(인터넷 전화)를 통해서 전달된다. 때문에 오디오데이터들은 사용자들의 민감한 정보들이 포함할 가능성이 매우 높다. 우리는 이 연구에서 오디오데이터들이 오갈 VoIP망을 설계 및 구현할 것이다. 또한 VoIP에 특화된 스니핑 도구를 제작하여 구현한 망에 적용시킬 것이다. 또한 우리는 캡처 된 패킷을 분석하여 오디오로 재구성하여 원하는 특정 정보를 추출해냄으로써 보안 취약점을 진단할 것이다. 그리고 가능한 해결책을 제시할 것이다.

* 광운대학교 소프트웨어학과
** 광운대학교 소프트웨어학과 교수(교신저자)

Ⅱ. 구현 환경 설정

우리는 VoIP망의 데이터들을 스니핑하기 위해서 상용망이 아닌 자체적인 VoIP망을 구축할 것이다. 망을 구현하기 위해 우리는 먼저 그림1과 같은 망을 설계하였다.



<그림 1> 구현할 망 설계도

2.1 서버 셋팅

우리 구현의 핵심이 될 부분은 IP-PBX라는 장비이다. PBX는 전화망의 교환기 같은 장비이다. 따라서 IP-PBX는 인터넷을 통하여 교환기역할을 수행하는 장비라고 생각할 수 있다. 우리는 IP-PBX가 될 하드웨어를 'Raspberry Pi'로 정하였다. 또한 IP-PBX로써의 역할을 수행하는 SIP서버로 오픈소스인 Asterisk Server를 사용하기로 하였다. SIP서버는 'Raspberry Pi'에서 구현하였다. 또한 서버를 더욱 효율적으로 관리하기 위하여 Asterisk의 웹 기반 GUI를 제공하는 FreePBX를 사용하였다. 우리가 사용하는 오픈소스들의 버전 정보는 다음과 같다.

- Raspbian Stretch LITE (Raspberry Pi OS)
- Asterisk 13.20.0
- FreePBX 14.0.3.2

서버를 구현을 마쳤다면 다음으로는 분석에 사용할 전화기 2대를 서버에 등록을 해야 한다. 전화기를 등록하기 위해선 먼저 서버페이지에 접속을 하고 admin계정으로 로그인을 한다. 그 후, [Applications]탭의 [Extensions]탭으로 들어간다. 이렇게 [Extension]탭으로 들어가면 사용자를 등록할 수 있는 창이 나온다. 이 때 Channel Driver별로 사용자를 생성할 수 있다. Channel Driver란 Asterisk서버의 외부에 있는 디바이스(SIP Telephone)와 통신을 할 때 들어오는 신호를 서버의 코어에게 변환시켜 전달해주는 역할을 한다. 다른 말로 VoIP Protocol 이라고도 한다[2]. Asterisk서버에서 사용할 수 있는 Channel Driver(VoIP Protocol)에는 여러 가지가 존재하는데 대표적으로 IAX2, DAHDI, Chan_sip, Chan_pjsip가 있다. 초기의 Channel Driver의 역할은 Chan_sip에 의해서 모두 수행이 되었지만 VoIP기술의 발달로 새로운 Channel Driver의 설계가 필요해짐에 따라 Chan_pjsip와 같은 새로운 스택의 Channel Driver들이 설계 되면서 여러 종류의 드라이버들이 존재하게 되었다. 하지만 우리의 연구는 Channel Driver에 영향을 받는 연구가 아니기 때문에 가장 초기의 기본 스택인 Chan_sip 드라이버를 사용하여 사용자를 생성하도록 하였다. 생성한 사용자의 정보는 표1과 같다.

<표 1> SIP서버에 등록한 사용자 정보

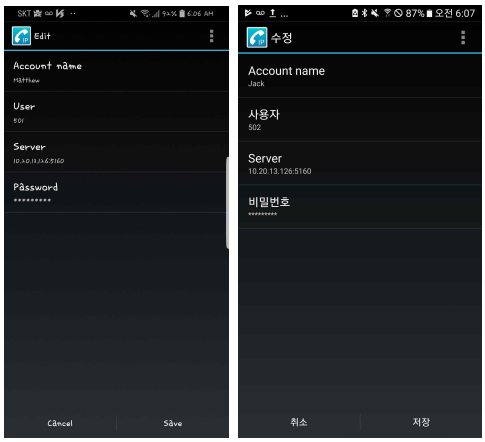
	Extension Number	Display Name	Channel Driver	Port
USER1	501	Matthew	Chan_sip	5060
USER2	502	Jack	Chan_sip	5060

2.2 클라이언트 셋팅

서버에 사용자등록을 마친 후, Client가 서버에 접속하기 위해서는 Client쪽에서도 설정을 해주어야한다. 우리 연구에서 Client는 SIP Telephone에 해당한다. 우리는 안드로이드 스마트폰 2대를 통해 SIP Telephone의 역할을 수행하도록 Regis Montoya에서 만든 오픈소스 안드로이드 어플리케이션인 'CSipSimple'을 사용하기로 하였다. 어플리케이션을 설치 한 후 서버에 등록한 정보를 이용하여 그림2과 같이 Client설정을 완료한다.

설정을 완료하면 서버 CLI창에서 "sip show users"라는 명령어로 등록된 사용자와 접속 여부를 확인할 수 있다. 이렇게 두 사용자가 모두 서버에 접

속하면 상대방에게 전화를 걸 수 있는 상태가 된다.



<그림 2> Client 등록 설정

Ⅲ. VoIP 프로토콜

VoIP전화는 발신자가 전화를 걸면 서버를 통해 SIP를 이용한 메시지가 전달되면서 상대방전화가 울린다. 그리고 상대방이 전화를 받으면 다시 SIP메시지가 발신자 측으로 전달되며 RTP Session이 시작된다. 이 때 오디오데이터들이 실시간으로 전송되면서 통화가 진행된다. 또한 RTP세션이 진행되는 동안 RTCP 프로토콜로 QoS(Quality of Service)정보를 교환하면서 품질을 모니터링 한다. 통화가 종료될 경우 역시 SIP메시지가 전달되며 서버를 통해 상대방에게 통화가 종료되었음을 알리게 된다. 이처럼 통화가 한번 이루어지는 데에는 여러 가지 프로토콜이 사용된다.[8]

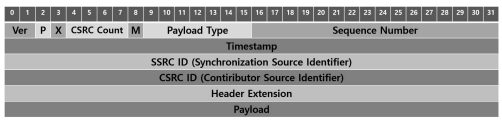
3.1 사용자측 설정

우리 연구는 RTP패킷을 스니핑하여 오디오데이터로 변환시킨 뒤 원하는 정보를 얻어내는 것을 목표로 한다. 따라서 오디오데이터가 들어있는 RTP프로토콜에 대해 연구하고자 한다.

RTP란 Real-time Transport Protocol로 오디오데이터나 영상데이터와 같이 실시간 전송되는 데이터들을 위해 설계된 프로토콜이다. RTP 프로토콜은 대부분의 경우 UDP프로토콜에서 사용된다[3].

3.2 RTP 구조

UDP상에서 사용되는 RTP패킷은 IP header, UDP header, RTP header가 붙어 있으며 가장 마지막에 오는 payload에 실질적인 오디오 데이터샘플이 존재한다. 하지만 payload에 존재하는 오디오샘플은 직렬화된 형태로 전달되어 오기에 단순히 샘플만으로는 오디오데이터로의 변환이 불가능하다. 따라서 header에 존재하는 부가적인 정보들이 필요하기에 패킷구조분석을 하려한다[3].



<그림 3> RTP 헤더 구조

그림3에 나온 RTP 헤더의 내용은 아래와 같다.

- Version (2bits): RTP프로토콜의 버전을 나타내는 비트이다. 현재의 RTP의 버전에 따라 2로 세팅되어 있다.
- CSRC Count (4bits): RTP의 헤더는 기본적으로 최소 12바이트의 크기를 가지는데 CSRC는 기본 크기의 헤더 뒤에 가변 크기로 존재하는 헤더필드이다. CSRC Count는 뒤따라오는 CSRC의 개수를 의미하며 0부터 15까지 세팅이 가능하다.
- Marker (1bit): 이 필드는 RTP 패킷 스트림 내에서 프레임의 경계같은 중요한 이벤트를 나타내기 위해 세팅된다.

- Payload Type (7bits): 이 필드는 Payload의 코덱을 나타낸다. 필드의 값에 따라 어플리케이션에서 어떤 코덱으로 payload를 해석할지 매핑되어 있다. 이 필드에서 나타내는 코덱에 따라 인코딩방법과 sampling rate가 결정되기에 패킷을 오디오 파일로 변환시키는 데에 아주 중요한 정보라고 할 수 있다.

<표 2> Payload type별 코덱

Payload Type	Codec (Encoding Name)	Clock Rate (Sampling Rate)
0	PCMU	8000
3	GSM	8000
4	G723	8000
6	DVI4	16000
7	LPC	8000
8	PCMA	8000
9	G722	8000

표2는 Payload type필드의 값에 따라 코덱들이 매핑된 것을 보여준다[4-5].

- Sequence Number (16bits): 이 필드의 값은 패킷의 순서를 나타내는 값이다. 시작점은 보안상의 이유로 랜덤한 값에서 시작하며 패킷의 순서에 따라 1씩 증가한다. 이 필드의 값으로 어플리케이션은 패킷 손실을 탐지할 수 있으며 패킷시퀀스를 복구하는데 이용할 수 있다.

- Timestamp (32bits): 이 필드의 값은 패킷의 첫번째 바이트(first octet)가 표본화(sampling)된 시간을 의미한다. 이 필드의 시작 값 역시 보안상의 이유로 랜덤한 값을 가지며 패킷당 일정한 간격으로 증가한다. 증가하는 간격은 payload type에 따른 코덱의 sampling rate에 따라 달라진다.

- SSRC (32bits): 패킷의 발신지에 따른 고유하고 랜덤한 ID값을 가지는 필드이다. 하나의 RTP Session에서는 중복된 SSRC 값을 가지면 안되기 때문에 RTP를 구현하는 과정에서 중복을 탐지하고 해결하는 방법이 존재하여야 한다.

- CSRC (variable): 이 필드의 값은 CSRC Count필드의 값에 따라 길이가 달라지며 count당 32bits의 크기를 가진다. 여러 개의 오디오데이터가 믹서에 의해 통합되어 전송될 경우 발신지의 개수가 CSRC Count 값으로 지정이 되고 발신지들의 모든 SSRC값이 이 필드의 값으로 지정된다.

Ⅳ. 구현 및 프로토콜 분석

4.1 패킷 스니핑

우리는 패킷을 스니핑하기 위해서 RTP패킷을 캡처하는 프로그램을 직접 작성하였다. 프로그램에 사용된 언어는 Python으로 3.5.3버전을 사용하였고 Raspbian Stretch LITE상에서 Vim을 이용하여 작성하였다.

우리 연구는 RTP패킷의 payload필드의 값이 필요하기 때문에 소켓을 생성하고 받아 온 직렬화된 패킷을 IP Header, UDP Header, RTP Header순서로 분리한 후 역직렬화시킨다. 그 후, 역직렬화된 패킷의 헤더에 담긴 정보를 활용하여 RTP 패킷만을 갖고 작업하도록 필터링한다. 이렇게 필터링을 거치고 나면 실질적인 오디오나 비디오 데이터가 담긴 RTP Payload필드가 남는다. 이 필드 역시 바이트 형태로 존재하는데 우리는 이것을 파일로 저장을 하여 Raw Audio파일을 생성하고 Google Cloud Speech Api를 활용하여 원하는 특정정보만을 오디오 파일로 저장함을 보일 것이다.

4.2 RTP 패킷 분석

그림4는 우리가 만든 프로그램을 이용하여 RTP

```
#####422 PACKET######
Datagram SIZE   : 200
Protocol        : UDP
Source IP       : 10.20.5.55
Destination IP  : 10.20.13.126
UDP source port         : 4000
UDP destination_port    : 12442
UDP packet_length       : 180
UDP header_checksum     : 53861
rtp_header_format(version=2, padding=0, extension=0, CSRC_count=0, marker=0, payload_type=0,
sequence_num=14180, timestamp=322880, SSRC='786745240')
Payload :
D5 EB 76 F8 78 57 F2 5E 50 E2 75 F6 D5 E5 D6 6D FF EC 79 F1 62 54 5F 63 67 F3 E8 DD E0 F0 D8
E6 69 71 63 E9 6B 65 5D 7A 71 78 76 74 70 DE DC F3 E5 62 59 6E 5F DB E8 D3 F0 E1 DD 70 52 56
68 67 75 FC DF 65 6A 59 70 61 E3 CF D6 E6 6A D8 EC 67 F6 76 5A 6C F5 E8 7C ED E1 62 61 7E 5A
72 70 6F 73 DD F6 6D 61 DE ED DF 6F 5E 6E 7C 51 EF DF E6 D1 DE 5F 76 FE 62 63 D2 DD 64 62 63
EA E0 6C 5D 7A 76 DC DB 4F 59 7D E1 EE 7B FC E7 E7 66 FE 6C FE EE F0 6F F8 DA E5 5E 63 6F E1
59 68 EE 63 5F

#####423 PACKET######
Datagram SIZE   : 200
Protocol        : UDP
Source IP       : 10.20.5.55
Destination IP  : 10.20.13.126
UDP source port         : 4000
UDP destination_port    : 12442
UDP packet_length       : 180
UDP header_checksum     : 64607
rtp_header_format(version=2, padding=0, extension=0, CSRC_count=0, marker=0, payload_type=0,
sequence_num=14181, timestamp=323040, SSRC='786745240')
Payload :
EF DF 61 FE E4 F2 7D 7D 67 DA E7 F5 72 6E EF 7D F6 FB 51 60 EF 6A DF DF E7 7C E5 E5 F3 EF 5F
72 73 6D DA E6 75 6C 63 69 6A 6A 64 EC 79 7A 71 E5 DF FF 76 68 62 67 E6 75 73 DF 79 6E E5 7D
73 FA E6 E1 78 FD E9 6E 63 6A F7 EF 74 F5 71 E9 ED 68 5F F4 FD ED 75 6C 67 EA FE 7E 6E F7 7A
7A E9 75 77 ED ED FB 64 6D EE 7A F7 F8 EF E2 FD 71 F9 FC F3 FB 73 78 78 7E 78 FE 74 FF 6B 7D
F9 77 F0 6D 6B 76 FA 7A F8 79 71 EC F5 75 74 E4 EE 7B 7A 76 EF FE 68 6E FF FB FE 77 FA F9 76
EE F1 7C F9 F6
```

<그림 4> 캡처된 RTP 패킷

Session이 개시된 이후 RTP패킷을 캡처한 화면 중 일부이다. 위의 캡처된 패킷의 정보로 알 수 있는 내용은 다음과 같다.

- Payload_type=0

이 필드의 값이 0이라는 것은 ITU-T Recommendation G.711에 정의되어 있는 PCMU라는 음성부호화방식을 사용한다는 것을 의미한다. PCMU는 G711의 두 가지 인코딩방식 중 μ-law(mu-law)를 사용하는 방식이다. 또한 PCMU방식은 한 패킷당 20ms구간의 표본화데이터를 가진다. Sampling Rate(Clock Rate)는 8000의 값을 갖는다. 따라서 이 방식은 1초당 50패킷의 데이터를 전송하며 하나의 패킷은 160개의 표본화데이터를 Payload필드에 담고 있다는 것을 알 수 있으며 오디오를 125㎲마다 표본화시켜 하나의 표본화데이터(=1 octet)를 생성한다는 것을 알 수 있다. 즉 이 필드의 값은 Audio파일로 변환시키는 데에 중요한 정보라고 할 수 있다[4-5].

- Sequence_number=14180, 14181

이 필드는 하나의 패킷만 봐서는 의미가 없으며 앞뒤의 다른 RTP패킷의 Sequence_number필드 값과 비교했을 때 의미가 있다. 이 값은 패킷의 순서에 따라 1씩 증가하기 때문에 패킷의 상대적인 시간적 위치를 파악할 수 있다. 앞의 패킷이 14180의 값을 가지고 뒤의 패킷이 14181의 값을 가지므로 순방향의 연속적인 패킷임을 알 수 있다.

- Timestamp=322880, 323040

이 필드의 값 역시 앞뒤의 패킷들과 비교했을 때 시간적인 관계를 알 수 있다. 이 값은 각 코덱의 표본화주기에 따라 증가량이 달라진다. 때문에 Payload Type필드의 값에 영향을 받는다고 할 수 있다. 우리의 경우 Payload Type의 값이 0인 PCMU의 방식을

사용한다. 따라서 표본화주기는 160이 되어 Timestamp의 값도 160씩 증가하는 것을 확인할 수 있다.

- Payload

위에서 언급했듯이 하나의 패킷의 Payload에는 160개의 표본화데이터가 존재한다. 실제로 그림에서 160개의 데이터가 존재하는 것을 확인할 수 있다. 이 필드에 존재하는 데이터는 실질적인 오디오 혹은 비디오 데이터들이다. 우리는 이 필드의 값을 Raw파일로 저장하여 원하는 정보를 추출하는 작업을 할 것이다.

4.3 오디오 파일 변환

우리는 우선 Payload필드에 있던 데이터들을 Raw파일로 저장을 했다. 그리고 Sox(Sound eXchange)라는 프로그램을 이용하여 저장된 Raw파일을 wav파일로 변환시켰다.

명령어 'sox –e mu-law –r 8000 capture_160.raw output_160.wav'는 capture_160.raw라는 RAW파일을 mu-law(PCMU)방식과 8000의 Sampling rate로 변환시켜 output_160.wav라는 WAV파일을 만드는 명령이다.

그림5는 wav파일로 변환시킨 결과이다. 변환된 파일을 재생한 결과, 단순히 패킷을 저장하여 변환시켰을 뿐인데 실제 사용자가 입력한 음성 그대로 재생이 되었다.

이렇게 암호화 되지 않은 패킷은 Google Speech Api를 활용하면 특정 키워드가 언급되는 것을 감지할 수 있다. 따라서 원하는 정보와 관련된 키워드를 등록해놓는 다면 사용자들의 대화를 실시간으로 필터링하여 정보를 얻어내거나 Raw파일을 저장해두고 원하는 정보만을 검색하여 얻어내는 것이 가능해진다. 만약 공격자가 "계좌"라는 키워드를 등록하고 제3자들의 VoIP통화를 스니핑한다면 공격자는 더 쉽고 빠르게 정보탈취가 가능해진다. 다음의 그림 6은 "계좌"라는 단어가 포함된 음성파일을 인식하는 모습이다.

```
(speech) root@raspbx:/home/snipy# python3 transcribe_mulaw_8000_ko_KR.py /home/snipy/output/c
apture_MyAccount2.raw
/usr/lib/python3.5/site-packages/google/auth/_default.py:66: UserWarning: Your application ha
s authenticated using end user credentials from Google Cloud SDK. We recommend that most serv
er applications use service accounts instead. If your application continues to use end user c
redentials from Cloud SDK, you might receive a "quota exceeded" or "API not enabled" error. F
or more information about service accounts, see https://cloud.google.com/docs/authentication/
  warnings.warn(_CLOUD_SDK_CREDENTIALS_WARNING)
Transcript: 이것은 테스트 용도로 녹음하는 것입니다 제 계좌번호는 5 6 7 8 9입니다
(speech) root@raspbx:/home/snipy#
```

<그림 4> 캡처된 RTP 패킷

```
root@raspbx:/home/snipy/output# ll
total 72
-rw-r--r-- 1 root root 71664 Nov 23 07:08 capture_test.raw
root@raspbx:/home/snipy/output# sox -e mu-law -r 8000 capture_test.raw capture_test.wav
root@raspbx:/home/snipy/output# ll
total 144
-rw-r--r-- 1 root root 71664 Nov 23 07:08 capture_test.raw
-rw-r--r-- 1 root root 71722 Nov 23 07:09 capture_test.wav
```

<그림 5> Sox를 이용하여 payload가 저장된 RAW파일을 WAV파일로 변환한 결과

이처럼 암호화 되지 않는 패킷은 공격자의 입장에서는 특정정보만을 추출해내거나 텍스트로 변환시키는 등 활용도가 매우 다양해 사용자의 정보가 유출되는 피해가 발생하기 쉽다.

Ⅴ. 결론

우리의 연구는 같은 사설IP범위 안에서 스니핑을 시도하여 RTP패킷을 캡처하는 시도를 하였다. 단순히 패킷을 받아 헤더 분석을 통해 Payload를 변환하는 작업을 진행했을 뿐인데 사용자의 음성통화 내용이 그대로 오디오파일로 재생되었다. 심지어 Google Speech Api와 같은 STT(Speech-To-Text)라이브러리를 활용한다면 더욱 손쉽게 정보의 탈취가 가능하다는 것을 보였다. 만약 공격자가 스푸핑과 같은 방법을 사용하여 패킷을 단순히 탈취하기만 한다고 해도 VoIP전화기 사용자의 통화내용은 모두 공격자에게 노출될 수 있다는 것이다. 이러한 문제는 RTP프로토콜에 별도의 인증과정이 없을 뿐 아니라 패킷 암호화 알고리즘이 적용되지 않은 채 패킷이 송수신되기 때문이다. 실제로 전문가들은 이러한 문제를 인식하고 문제를 해결하기 위해 Payload필드에 암호화를 적용하는 SRTP프로토콜을 개발하였다. SRTP프로토콜은 실질적인 민감한 데이터가 들어가 있는 Payload필드를 AES알고리즘으로 암호화하는 방식으로 데이터의 기밀성을 유지한다. 이때 패킷의 헤더는 암호화시키는 않는데 이것은 라우터나 스위치에서 전송 경로를 파악하는 데에 헤더의 정보가 필요하기 때문이다. 그래서 SRTP프로토콜은 HMAC-SHA1함수로 인증과 무결성 체크 기능을 제공한다. 하지만 RTP를 활용한 여러 개발사들은 VoIP전화의 성능의 문제로 SRTP를 대부분 적용하고 있지 않아 우리가 연구에 사용한 Asterisk Server역시 암호화 되지 않은 RTP프로토콜을 사용하였다. 다른 말로 사용자들의 오디오데이터가 공격자에게 쉽게 노출될 수 있는 상황에 있다는 것이다. 물론 VoIP에는 SIP와 같은 다른 여러 프로토콜도 쓰이기에 SIP통신을 TLS방식으로 보안성을 높이는 것도 중요하다. 실제 AES로 Payload를 암호화하는 암호화키는 SIP패킷의 교환 시 이루어진다. 따라서 TLS로 SIP패킷을 암호화하지 않은 상태에서 SRTP프로토콜을 사용하는 것은 보안성을 향상시켰다고 보기 어렵다. 따라서 RTP만을 사용하는 VoIP통신을 지양하고 SRTP프로토콜을 사용하고 SIP를 TLS로 암호화하는 방식으로 사용자의 정보를 보호하는 것이 요구된다.[6-10]

참고문헌

[1] KakaoTalk, https://www.kakaocorp.com/service/KakaoTalk

[2] Malcolmd, “Channel Driver Modules,” https://wiki.asterisk.org/wiki/display/AST/Channel+Driver+Modules

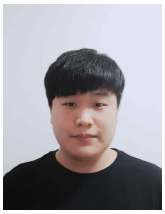
[3] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications.,” RFC3550, The Internet Society, 2003.

[4] H. Schulzrinne, S. Casner, “RTP Profile for Audio and Video Conferences with Minimal Control.,” RFC3551, 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, 와이어샤크 네트워크 완전 분석 (이재광, 전태일 역), 에이콘, 2012
[9] 김창식, 김태경, “텍스트마이닝을 이용한 정보보호 연구동향 분석,” (사)디지털산업정보학회논문지, 제14권, No.2 , 2018, pp. 19-25.
[10] 최희식, 조양현, “모바일 앱 서비스 이용 증가로 인한 보안 위협 분석,” (사)디지털산업정보학회논문지, 제14권, No.1 , 2018, pp. 45-55.

■ 저자소개 ■



이 동 건
(Lee Donggeon)

2015년 3월~현재
광운대학교 컴퓨터소프트웨어학과

관심분야 : 데이터네트워크 보안
E-mail : secmatth1996@gmail.com

최 웅 철
(Choi Woongchul)

2002년 9월~현재
광운대학교 컴퓨터소프트웨어학과
교수
2001년 5월 Univ. of Illinois, Ph.D.,
Computer Science
1991년 2월 서울대학교
컴퓨터공학과(공학석사)
1989년 2월 서울대학교 컴퓨터공학과(공학사)

관심분야 : 데이터네트워크, 보안
E-mail : wchoi@kw.ac.kr

논문접수일 : 2018년 11월 26일
수 정 일 : 2018년 12월 10일
게재확정일 : 2018년 12월 14일