Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.6, Dec. 2016 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2016.26.6.1505

안드로이드 앱을 이용한 실시간 유료 방송 취약점 분석*

최 현 재,† 김 형 식‡
성균관대학교

A Vulnerability Analysis of Paid Live Streaming Services Using Their Android Applications*

Hyunjae Choi,† Hyoungshick Kim‡
Sungkyunkwan University

요 약

동영상 실시간 스트리밍 (Live Streaming)이란 텔레비전 생방송처럼 촬영한 정보를 실시간으로 사용자의 동영상 플레이어로 보내 재생하도록 하는 방식을 말한다. 비디오와 오디오를 실시간으로 인코딩하여 많은 사용자들이 동시에 시청하기 위해서는 RTMP (Real Time Messaging Protocol), HLS (HTTP Live Streaming) 등 실시간 스트리밍을 지원하는 프로토콜이 필요하다. 본 논문에서는 실시간 스트리밍을 제공하는 국내 6개 OTT (Over the Top) 업체의 어플리케이션을 대상으로 패킷 캡처를 통해 어플리케이션들의 패킷을 분석하였다. 채널 목록을 암호화하지 않거나, 유료채널에 적합하지 않은 프로토콜을 사용함으로써 유료채널을 무료로 시청할 수 있다는 취약점을 밝히고자 한다.

ABSTRACT

Live streaming is a method to provide media service by sending recoded media to a user's video player. In order to provide video and audio contents in real-time for a large number of users simultaneously, live streaming compatible protocols such as RTMP (Real Time Messaging Protocol), HLS (Http Live Streaming), are required. In this paper, we analyzed vulnerability of paid live streaming services with the captured packets from the applications used by six major OTT (over-the-top) companies in Korea supporting live streaming services. We found that streaming channels were not encrypted and access control mechanisms were not properly used. Thus, guest users can freely use paid live streaming services.

Keywords: Live Streaming, HLS, OTT, Premium channel, vulnerability

I. 서 론

시간과 장소에 제약 없이 실시간 방송을 TV, 스마트폰, PC 등 다양한 기기를 통해 이용할 수 있는 'N 스크린' 서비스를 이용하는 사람들이 많아지면서 OTT (Over the Top) 서비스 시장 또한 급증하고 있다[1]. 그 결과 케이블 방송 및 위성방송, IPTV 등 고정형 TV 중심의 유료 방송의 가입해지를 하는 이용자들의 비율이 증가하고 있다. 시장조사업체 스트라베이스에 따르면 국내 OTT 시장 규모는 2013년 1490억 원에서 2016년 3069억 원, 2019년 6345억 원, 2020년 7801억 원으로 성장할 것으로 예상하고 있다[2]. 현재 국내에는 6개의 OTT 업체에서 실시간 방송 서비스를 제공하고 있다.

국내 6개의 OTT 업체 모두 모바일 어플리케이션을 통해 사용자에게 실시간 방송 서비스를 제공한다. PC 환경과 다른 모바일 환경에서는 어떤 프로토콜

Received(09. 22. 2016), Modified(12. 12. 2016), Accepted(12. 12. 2016)
* 본 논문은 한국연구재단의 신진연구지원사업(2014R1A1A1003707)에 의하여 연구되었음.
* 본 논문은 2016년도 하계학술대회에 발표한 우수논문을 개선 및 확장한 것임.
† 주저자, nowc@skku.edu
‡ 교신저자, hyoung@skku.edu(Corresponding author)

을 사용하여 스트리밍을 하는지 어플리케이션 패킷을 캡처하여 분석을 진행하였다. 그 결과 몇몇의 OTT 업체는 전체 채널 목록을 암호화 하지 않았고, 인가되지 않은 사용자가 미디어 서버에 접근 할 수 있는 문제가 있었으며, 유료 채널의 미리 보기 기능만으로 유료 회원이 사용하는 스트리밍 주소가 노출되는 문제점을 찾을 수 있었다.

본 논문에서는 이러한 문제점을 이용하여 유료 채널을 무료로 시청할 수 있는 취약점을 살펴보고자 한다.

II. 실시간 스트리밍 전송 기술

2.1 Real Time Messaging Protocol

RTMP (Real Time Messaging Protocol)는 Adobe 사의 독점 프로토콜로 오디오, 비디오 및 기타 데이터를 플래시 플레이어를 이용하여 스트리밍할 때 사용된다[3][4]. TCP 기반에 암호화되지 않은 프로토콜로, 1935번 포트를 기본적으로 사용하며 실패할 경우 433번 포트 (RTMPS)나 80 포트 (RTMPT)로 재시도한다.

RTMP는 비디오 및 기타 데이터를 여러 조각들 (fragments)로 나누기도 한다. 이 조각들의 크기는 클라이언트와 서버 간에 유동적으로 결정되며, 비디오 및 기타 데이터에 대한 스트림 조각들의 기본 크기는 128 바이트이다.

RTMP의 암호화 방법으로는 표준 TLS/SSL[8] 메커니즘을 사용하여 스트리밍 주소를 암호화 시키는 방법과 RTMPE (128비트로 암호화된 RTMP)를 사용하여 스트리밍 주소를 암호화 시키는 방법이 있다.

2.2 HTTP Live Streaming

HLS (HTTP Live Streaming)는 Apple에서 iOS 3.0과 QuickTime X를 위해 개발한 프로토콜로 IETF를 통한 표준화 작업을 통해 Apple 디바이스뿐만 아니라 Android를 포함한 다양한 디바이스 운영체제에서도 지원하기 시작하였다[5].

Fig. 1과 같이 인코딩 된 스트리밍 데이터를 MPEG-2 전송 스트림 (Transport Stream)에 담아 전송하면 스트림 세그멘터 (Stream Segmenter)는 일정한 시간 간격마다 입력받은 스트리밍 데이터를 분할해 파일 (ts)을 만들고, 그 분

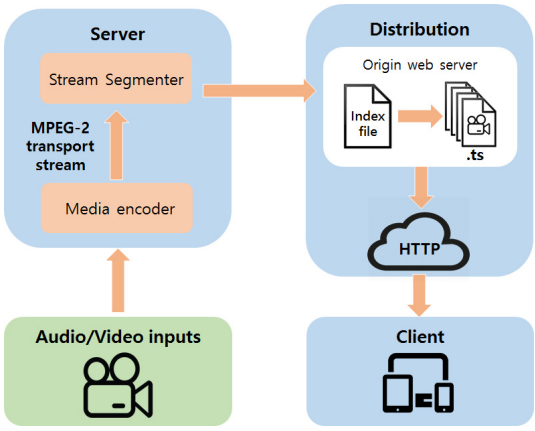

Fig. 1. Composition of HLS

할한 파일에 접근할 수 있는 재생 목록 (m3u8)을 생성하는 일을 한다[6]. HTTP는 양방향 방식이 아니기 때문에, 클라이언트에서 반드시 서버에 요청을 해야 그에 맞는 응답을 받을 수 있다. 즉, 잘게 쪼갠 동영상과 다음에 볼 동영상 정보를 함께 클라이언트에 전달하고 동영상이 끊임없이 때에 맞춰 다음 동영상 정보를 요청한다.

HLS는 콘텐츠 보호를 위해 분할된 파일 (ts)을 개별적으로 암호화하는 방식을 사용한다. 암호화가 적용되면 암호화에 사용한 키 (key)를 #EXT-X-KEY 지시어로 플레이어에 제공한다. 플레이어는 이 키 파일에 기록된 키를 이용하여 ts 파일을 해독하여 재생한다.

2.3 스트리밍 엔진

앞서 소개한 RTMP, HLS 프로토콜로 실시간 방송을 수많은 디바이스에 스트리밍하기 위해서는 스트리밍 엔진이 필요하다. 대표적인 스트리밍 엔진으로는 MediaSystems 사의 Wowza 스트리밍 엔진이 있다.

Wowza 스트리밍 엔진은 토큰 기반 사용자 인증을 통해 실시간 방송 서비스를 제공한다[7]. Fig.2와 같이 웹 서버와 미디어 서버를 갖고 있으며, (1) 사용자가 원하는 방송의 채널, 화질 등을 선택하여 웹 서버에게 스트리밍 주소를 요청하면 (2)웹 서버는 사용자의 정보와 암호화 알고리즘을 이용하여 토큰을 생성한다. (3)웹 서버는 생성된 토큰을 미디어 서버에 전달함과 동시에 미디어 서버 주소와 토큰을 담은 웹 페이지를 생성하여 사용자에게 전달한다. 사

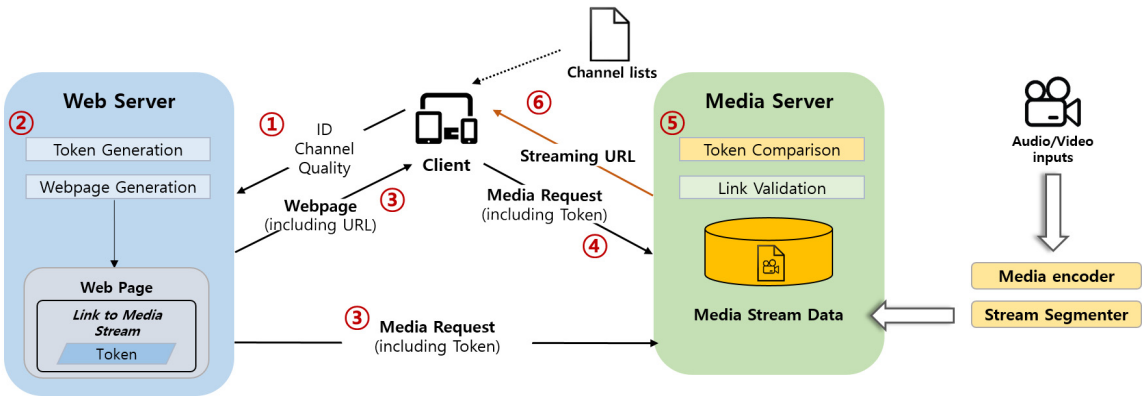

Fig. 2. Composition of streaming engine

용자는 (4)웹 페이지에 담긴 미디어 서버 주소와 토큰을 이용하여 미디어 서버에게 스트리밍 주소를 요청하고, (5)미디어 서버는 스트리밍 주소를 요청한 사용자의 토큰이 웹 서버에게 전달 받은 토큰과 일치하는지 확인 후 (6)스트리밍 주소를 사용자에게 알려준다.

국내 6개 OTT 업체 모두 이러한 스트리밍 엔진을 사용하여 다양한 디바이스에 실시간 방송 서비스를 제공한다.

III. 어플리케이션 환경에서의 취약점

3.1 점검 대상 및 분석 환경

본 논문은 Google Play Store에서 유료 채널을 제공하는 국내 6개 OTT 업체의 안드로이드 어플리케이션을 대상으로 한다.

SHV-E300K 기기를 루팅하여 bitShark[9] 어플리케이션을 통해 OTT 업체의 어플리케이션들의 패킷을 캡처하였고, 캡처한 .pcap 파일은 Windows 10 환경에서 Wireshark 2.2.0[10] 프로그램으로 분석하였다. TLS/SSL로 암호화 된 패킷은 임의의 인증서를 SHV-E300K에 설치하여 Fiddler 4[11] 프로그램으로 패킷을 동적 분석하였다.

RTMP 형식의 스트리밍은 안드로이드 환경, PC 환경 모두 RTMP를 지원하는 별도의 동영상 플레이어가 필요하고, HLS 형식의 스트리밍은 안드로이드 환경의 경우 기본 브라우저로 실행 가능하며, PC 환경의 경우 멀티미디어 재생 목록 (m3u8)을 지원하는 플레이어에서 실시간 방송을 재생할 수 있다.

3.2 암호화 되지 않은 채널 목록

국내 6개 OTT 업체의 어플리케이션들 중에서 RTMP 프로토콜을 사용하는 어플리케이션은 A사 1개가 있었으며, 위 환경에서 패킷을 캡처하여 분석하였다.

어플리케이션 초기 실행 단계에서 암호화 없이 모든 채널 목록을 xml 형식으로 불러오는 것을 확인하였으며, Fig. 3과 같이 채널 목록으로부터 모든 유료 채널들의 'id'를 알 수 있었다. A사는 웹 서버를 통해 유료 회원이 확인되면 유료 채널 id를 사용하여 미디어 서버에게 스트리밍 주소를 요청하였다. 무료 채널 또한 같은 방식으로 스트리밍 주소를 요청하는 것을 확인 하였다. Fig. 4와 같이 무료 스트리밍 주소를 요청 할 때 'channel' 항목만 유료 채널의 id로 바꾸면 유료 채널의 스트리밍 주소를 받아올 수 있었다.

위 방법을 통해 A사에서 제공하는 약 250개의 유, 무료 채널 모두 안드로이드와 PC 환경에서 시청

Fig. 3. Premium channel information for company A

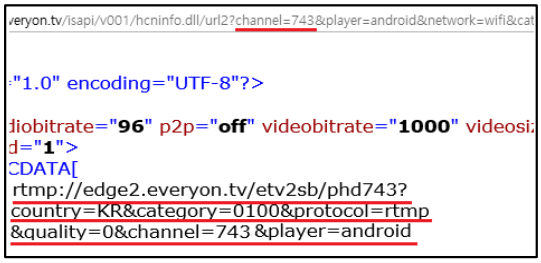

Fig. 4. Premium channel streaming url for company A

할 수 있었다.

3.3 토큰 기반 사용자 인증 우회

국내 6개의 OTT 업체의 어플리케이션 중 5개의 어플리케이션에서는 HLS 프로토콜을 사용하고 패킷을 암호화하지 않는 어플리케이션은 2개가 있었으며, 그 중 B사의 어플리케이션을 3.2와 동일한 방법으로 분석하였다.

B사의 경우 채널 목록이 json 형식으로 구성이 되어 있었으며, Fig. 5와 같은 정보가 확인되었다. 다른 OTT 업체 어플리케이션들과 다르게 B사는 유일하게 지상파를 지원하였고 채널 목록에서 확인된 'quality' 항목에 따라 다른 요금을 부과한다. 유료 채널인 경우 비회원에게는 약 2분의 미리 보기를 제공한다.

B사는 토큰 기반 사용자 인증을 통해 실시간 방송 서비스를 제공한다. 유료 회원과 비회원 각각의 계정으로 같은 화질의 유료 채널을 선택하게 되면 어플리케이션은 웹 서버에게 스트리밍 주소를 요청하게 된다.

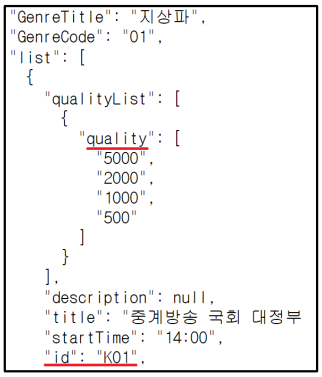

Fig. 5. Premium channel information for company B

Fig. 6과 같이 유료 회원이 유료 채널을 선택하면 'credential' 항목에 계정의 정보가 암호화되어 웹 서버에게 요청을 보내고 비회원이 유료 채널을 선택하면 Fig. 7과 같이 none 값을 넣어 웹 서버에게 요청을 보내는 것을 확인하였다.

웹 서버가 유료 회원과 비회원에게 전달해준 웹 페이지에는 Fig. 8과 Fig. 9와 같이 토큰을 포함한 'url' 미디어 서버 주소가 담겨있다. 비회원 웹 페이지에는 'adUrl' 광고 항목이 추가되었을 뿐 유료 회원과 미디어 서버 주소가 같다는 것을 알 수 있었다. 또한, 미디어 주소 뒤에 붙은 토큰 값은 새로운 디바이스에서 같은 토큰 값으로 미디어 서버에 접근 할 때 이전에 접속된 디바이스의 시청을 차단하는 기능만을 하였다.

미디어 서버 주소에서 받아온 실제 스트리밍 주소는 Fig. 10과 같이 유료 회원과 비회원 모두 같았으며, 비회원의 경우 일정 시간 후 시청이 제한되는 것은 어플리케이션 내에서 작동한 기능임을 알 수 있었다.

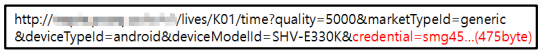

Fig. 6. Streaming request url of premium members for company B

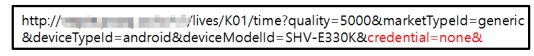

Fig. 7. Streaming request url of guest users for company B

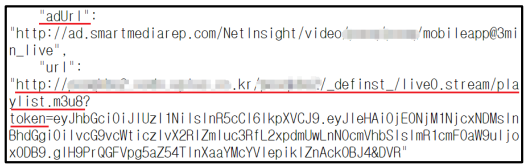

Fig. 8. Web page recived by premium members of company B

Fig. 9. Web page recived by guest users of company B

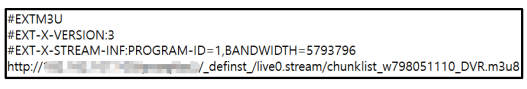

Fig. 10. Streaming url for company B

결국 비회원 계정만으로 미디어 서버에서 유료 채널의 스트리밍 주소를 받아올 수 있었고, B사에서 제공하는 약 70개의 유, 무료 채널 모두 안드로이드와 PC 환경에서 시청할 수 있었다.

3.4 미리 보기 기능으로 인한 스트리밍 주소 노출

국내 6개의 OTT 업체의 어플리케이션 중 HLS 프로토콜을 사용하며, TLS/SSL 로 패킷을 암호화하는 어플리케이션 3개가 있었으며, 그 중 C사의 어플리케이션을 3.3과 동일한 방법으로 분석하였다.

C사의 경우 채널 목록, 스트리밍 요청 주소, 전달받은 웹 페이지까지 TLS v1.0 으로 암호화하는 것을 확인하였다.

하지만 Fig. 11과 같이 사용자와 미디어 서버 사이에는 암호화 되어 있지 않았고 미디어 서버 주소와 토큰 그리고 OTP 값을 이용하여 미디어 서버에게 스트리밍 주소 요청하는 것을 확인할 수 있었다.

Fig. 12와 같이 비회원 계정을 통해 유료 채널 미리 보기만으로 유료 회원과 같은 스트리밍 주소를 받을 수 있었으며, 일정 시간 내에 다시 재생을 한다면 PC 뿐만 아니라 다양한 디바이스에서 시간 제약 없이 유료 채널을 시청할 수 있었다.

```
GET /live/TVNHD.daml/playlist.m3u8?
device_id=AF2A9103A48549079C53DCA6051361F7CD5D8E7D3&token=bkMOwSaB6VInMa
YunOmz7n0jGQsdwDz85JTut1501%252F2kIIFIZ8F97j36q
%252F4NNBXJf3AP&masterId=M7140521568x-auth-
info=DB9AEBA6870BC0C5D03AE96552A928F703F10D75561A193431980842F7DF9342&x-
device-info=SHV-E330K%2F0.0.0&x-os-info=android%2F4.4.2&x-service-
info=mobiletv
%2F3.1.0&OTP_VALUE=80733F870BD39D51E64601F1B66FDA424C805F6751AF8519D9D73
F33E42F9DF4209CB8EA4AE187EBCA5323006112753266FC715752ED4072C69C486899038
FC7&x-link=wifi HTTP/1.1
Host: onair.hanafostv.com:8080
User-Agent: stagefright/1.1 (Linux;Android 4.4.2)
Connection: Keep-Alive
```

Fig. 11. Midea Server access url for company C

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=2000000
http://1.226.200.57:8080/live/TVN_HD/chunklist.m3u8?
digicapsessionid=704120491
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1000000
http://1.226.200.57:8080/live/TVN_SD/chunklist.m3u8?
digicapsessionid=704120491
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=500000
http://1.226.200.57:8080/live/TVN_LD/chunklist.m3u8?
digicapsessionid=704120491
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=56000
http://1.226.200.57:8080/live/TVN_AO/chunklist.m3u8?
digicapsessionid=704120491
```

Fig. 12. Streaming url for company C

3.5 분석 결과

국내 6개 OTT 업체 모두 사용자 인증을 하는 웹 서버와 스트리밍 주소를 알려주는 미디어 서버가 분리되어 있다. 어플리케이션 초기 실행 단계에 채널 목록을 웹 서버에서 가져오는데 채널 목록이 암호화 되어 있더라도 유료 채널 미리 보기가 가능하다면 미디어 서버에 누구나 접근할 수 있다는 것을 확인하였다. 미리 보기 스트리밍 주소는 유료 회원이 사용하는 스트리밍 주소와 모두 동일하였고 미리 보기의 시간제한은 어플리케이션 내에서 작동하는 기능이기 때문에 스트리밍 주소만 알게 된다면, 시간제한 없이 다양한 디바이스에서 동시 시청이 가능하였다. 오직 E사의 어플리케이션만 미리 보기를 지원하지 않았기 때문에 비회원 계정으로 실시간 유료 방송을 시청할 수 없었다.

이러한 취약점이 존재하는 이유는 크게 두 가지가 있다. 첫 번째로는 유료 실시간 방송을 비회원에게 미리 보기로 제공하려면 미디어 서버에서 시간을 제한 시켜야 하는데 어플리케이션에서만 시간을 제한하고 있다는 것이다. 두 번째로는 RTMP 나 HLS 프로토콜 상에도 암호화 기능이 있지만 암호화와 복호화 과정에서 오버헤드가 생기면 실시간 방송의 경쟁력이 떨어지기 때문에 사용하지 않고 있다는 것이다.

Table 1. Security check result of six OTT companies (RTMP: A, HLS: B, C, D, E, F)

	Channel list Encryption	Preview	Packet Encryption	Premium Access
A	X	X	X	O
B	X	O	X	O
C	O	O	O	O
D	O	O	O	O
E	O	X	O	X
F	X	O	O	O

IV. 대응 방안

국내 6개 OTT 업체가 사용하는 스트리밍 엔진은 콘텐츠 보호를 위해 분할된 파일 (ts)들을 각각 다른 암호로 암호화하고 플레이어가 해독하여 재생하는 기능을 갖추고 있다. 하지만 대부분의 OTT 업체들은 실시간 방송 특성상 복호화 과정에서 시간차가 발생할 수 있는 문제와 모바일 디바이스의 성능상의 문

제로, 분할된 파일 (ts)들을 각각 다른 암호로 암호화하고 복호화 하는 기능은 사용하지 않고 있다. 대부분의 OTT 업체들은 미디어 서버에게 스트리밍 주소를 요청할 때만 토큰 등을 이용한 사용자 인증을 할 뿐 그 이상의 인증 수단은 보이지 않았다.

앞서 분석한 취약점을 보완하기 위해 세 가지 대응 방안을 제시한다. 첫 번째로 E사와 같이 비회원에게 제공하는 유료 채널 미리 보기 기능을 삭제하는 것이다. 미리 보기 기능을 삭제 한다면 미디어 서버의 주소를 알더라도 토큰을 발급 받을 수 없기 때문에 로그인 없이는 유료 채널에 접근 할 수 없을 것이다. 두 번째로 미리 보기만을 제공하는 미디어 서버를 새로 구축하는 것이다. 일정 시간 후 시청이 제한되는 미리 보기 기능은 미디어 서버가 아닌 어플리케이션에서 작동하는 기능이기 때문에 분석 결과와 같은 취약점이 발생한다. 따라서 미리 보기용 미디어 서버를 구축하고 비회원에게는 일정 시간 동안만 재생할 수 있는 재생 목록 (m3u8)을 제공한다면 미리 보기로 인한 취약점은 사라질 것이다. 마지막으로 웹 서버에서 토큰 생성 시 유효기간을 정하고 주기적인 재발급과 검증을 하는 것이다. 미디어 서버는 사용자가 유료 채널의 스트리밍 주소를 요청할 때 최초로 한번만 토큰을 확인하고 토큰 값이 담긴 재생 목록 (m3u8)을 제공하기 때문에, 다른 디바이스에서 재사용 할 수 있는 취약점이 발생한다. 비회원뿐만 아니라 유료 회원에게도 일정 시간이 지나면 발급 받았던 토큰을 폐기하고 새로운 토큰을 생성하여 재생 목록 (m3u8)에 갱신한다면 스트리밍 주소를 재사용할 수 있는 취약점은 사라질 것이다.

V. 결론 및 향후 계획

본 논문에서는 모바일 환경에서의 실시간 유료 방송의 취약점에 대하여 분석하였다. 국내 6개 OTT 업체의 어플리케이션 중 5개의 어플리케이션에서 유료 채널을 무료로 볼 수 있는 취약점을 발견하였다.

5개의 어플리케이션에서 제공하는 채널들은 어플리케이션에서 제공할 수 있는 국내 대부분의 채널을 포함하기 때문에 보안적으로 안전한 서비스를 제공하는 OTT 업체에게는 막대한 피해가 있을 것이다.

OTT 서비스 중 실시간 방송은 앞으로도 큰 비중을 차지할 것이며, OTT 업체들은 스트리밍 엔진과 선택한 프로토콜에 대해 점검이 필요하다.

향후 연구에서는 스트리밍을 이용한 유료 VOD 서비스 취약점과 실시간 스트리밍 콘텐츠의 효과적인 보호 방법을 연구할 것이다.

References

[1] Gi-man Seo, "OTT Understanding and prospects of service," Broadcasting and Media Magazine, 16(1), pp. 91-101, Mar. 2011

[2] Etoday News, "http://www.etoday.co.kr/news/section/newsview.php?idxno=1321277," Apr. 2016

[3] Parmar H and M. Thornburgh, "Adobe's Real Time Messaging Protocol," Copyright Adobe Systems Incorporated, Dec. 2012.

[4] Modi Darshan, "Quality Control in Video Streaming," International Research Journal of Engineering and Technology, vol. 2, pp. 1228-1231, Sep. 2015.

[5] Pantos, R and W. May, "HTTP Live Streaming draft-pantos-http-live-streaming-05," Published by the Internet Engineering Task Force, Nov. 2010.

[6] Ma, Kevin J and Radim Bartos, "HTTP live streaming bandwidth management using intelligent segment selection," Global Telecommunications Conference, pp. 1-5, Jan. 2011.

[7] Riegel, Brian M and James S. Sherry, "Token-based security for links to media streams," U.S. Patent No. 8,640,229, Jan. 2014.

[8] Dierks, Tim, and Christopher Allen. "The TLS protocol version 1.0," Published by the Internet Engineering Task Force, Jan. 1999.

[9] bitShark, "https://play.google.com/store/apps/details?id=blake.hamilton.bitshark"

[10] WireShark, "https://www.wireshark.org"

[11] Fiddler, "http://www.telerik.com/fiddler"

〈저자소개〉

최 현 재 (Hyunjae Choi) 학생회원
2016년 3월~현재: 성균관대학교 소프트웨어대학 석사과정
〈관심분야〉 정보보호, 네트워크 보안, 역공학

김 형 식 (Hyoungshick Kim) 종신회원
1999년 2월: 성균관대학교 정보공학과 졸업
2001년 2월: KAIST 전산학과 석사
2012년 2월: University of Cambridge 컴퓨터공학과 박사
2013년 3월~현재: 성균관대학교 컴퓨터공학과 조교수
〈관심분야〉 정보보호