배포 가능, 수정 불가

Torrent Forensics

Torrent Forensics A-Z

2013-09-22

An investment in knowledge still yields the best returns. By Benjamin Franklin

지식에 투자하는 것이 여전히 최고의 수익을 낳는다. By 벤자민 프랭클린

Keyword : BitTorrent 의 원리, 망 내에서의 최초 공유자 식별 방법, BitTorrent Client Artifacts

1. 서론

현재 국내 파일 공유는 웹 하드와 토렌트 프로그램이 주를 이루고 있다. 그러나, 웹하드는 저작권 문제로 인해 많은 자료들이 배포 금지되고 있어, 사람들이 원하는 모든 자료를 충족하여 주지 못하고 있다. 이와 같은 이유로 최근 파일 공유 추세는 토렌트를 더 선호하고 있다.
본 문서에서는 토렌트 프로토콜인 BitTorrent 의 동작 원리와, 디지털 포렌식 관점에서의 토렌트 분석 방법 및 망 내에서의 최초 공유자 식별 방법, Windows 에서의 BitTorrent 클라이언트의 Artifacts 에 대해 정리 해보고자 한다.

2. BitTorrent 란?

BitTorrent 는 비공식적 통계로 전 세계 인터넷 트래픽 중 상당 부분을 차지하고 있을 정도로 사용량이 매우 높다. 이는 최초에 BitTorrent 프로토콜과 동명의 클라이언트 프로그램으로 구현되었으며, 현재는 프로토콜과 호환 되는 다양한 토렌트 클라이언트 프로그램으로 구성되어 있다. BitTorrent 프로토콜은 P2P(Peer to Peer) 방식이며, 프로그래머 브렘코헨이 설계 및 구현하였고, 현재 BitTorrent 프로토콜과 클라이언트 프로그램은 브렘코헨이 설립한 BitTorrent.inc 회사에서 유지 보수 중이다.

2.1 국내 법 관점에서의 BitTorrent

국내 법에서는 아직까지 BitTorrent 의 불법성을 정의한 사례는 없다. 다만, BitTorrent 의 특성을 고려해 수시기관에서 저작권 물에 대한 감시를 진행하고 있을 뿐이다. BitTorrent 는 '아동·청소년 성 보호에 관한 법률 위반(이하 아청법)' 때문에 사회적으로 많은 언급이 되었으며, 수사기관에서는 업로드 사용자뿐만 아니라 다운로드 사용자에 대해서도 검거를 하려 하고 있다. 현재는 수사기관에서 아청법과 관련한 저작물 만을 감시하고 있지만, 감시물의 대상이 넓어진다면 BitTorrent 사용자 대부분은 경찰서에 출두해야 하는 사태가 벌어질지도 모른다. 그렇다면 "왜 수사기관은 다운로드 사용자도 검거하려는 것일까?" 간단히 언급하자면 BitTorrent 는 다운로드와 동시에 공유가 진행되는 동작 원리이기 때문이다. 이는 사실이긴 하지만, 수사기관에서는 BitTorrent 연구를 통해 업로드 사용자와 다운로드 사용자의 기준을 좀 더 명확히 세울 필요가 있다.

2.2 BitTorrent 용어

BitTorrent 에서 사용하는 다양한 용어는 아래와 같이 정리할 수 있다.

Piece(조각)	공유 파일을 작은 용량으로 조각 낸 파일
Seeder(시더)	공유 파일의 완전한 파일을 가지고 있는 클라이언트
Seed File(시드 파일)	공유 파일 업로드 또는 다운로드 시 필요한 정보를 담고 있는 파일
Leecher(리처)	공유 파일의 일부분만을 가지고 있는 클라이언트
Peer(피어)	시더와 리처를 총칭
Tracker(트래커)	피어들의 정보를 관리해 파일 공유를 원활하게 하는 서버
Swam(스웜)	Tracker 에 등록 된 공유 파일마다 존재하는 일종의 데이터 베이스로 공유하며, 파일 식별자(Hash), 피어 리스트 정보를 가짐

표 1 – BitTorrent 용어 정리

3. BitTorrent Protocol 의 동작원리

BitTorrent 프로토콜의 원리는 충실히 P2P 를 따른다. 그러나 BitTorrent 프로토콜은 자신만의 공유 정책을 세워 효율성을 높였는데, 대표적인 공유 정책으로 "optimistic unchoking" 메커니즘이 있다. 해당 메커니즘은 클라이언트의 대역폭을 일부 강제적으로 할당하여 무작위로 피어들에게 조각을 보내어 모든 피어들이 일정한 조각을 가지도록 하며, 이와 같은 원리를 통해 공유 속도와 효율성을 증대하였다. 아래 그림을 통해 간단한 동작원리를 알아보도록 하겠다.

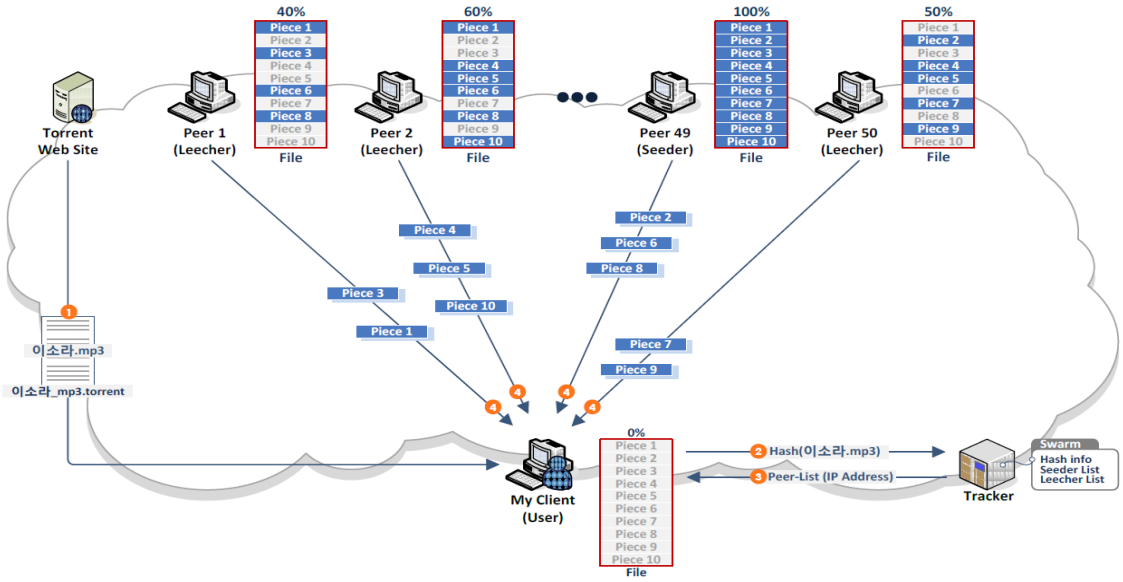

그림 1 – BitTorrent 동작 원리

(1) 시드 파일 다운로드

인터넷에는 토렌트 시드 파일 공유를 위한 수 많은 토렌트 커뮤니티가 존재한다. 국내에는 대표적으로 TorrentRG, eTorrent 등이 있으며, 파일 다운로드를 위해서는 시드 파일이 필요하므로 토렌트 커뮤니티 사이트에서 시드 파일을 다운로드 받는다. 이 때 반드시 시드 파일을 다운로드 할 필요는 없으며, 공유 파일의 식별자를 입력하거나 마그넷 주소를 입력하여 다운로드 할 수도 있다. 그러나 동작 과정은 아래와 동일하다.

(2) Tracker Request

시드 파일에는 트래커 주소, 공유 파일의 식별자 등이 포함되어 있으며, 토렌트 클라이언트 프로그램은 이러한 정보를 추출 해 트래커의 주소로 공유 파일의 식별자와 기타 정보를 전송한다. 이 때의 메시지 또는 패킷을 Tracker Request 라고 부른다. 이러한 Tracker Request는 파일 공유를 진행 중인 모든 피어들이 트래커에게 최초 수행하며, 트래커는 동일한 식별자 값을 보내온 피어들의 주소를 스웜의 피어 리스트 정보를 통해 관리하게 된다.

(3) Tracker Response

Tracker Request 패킷을 전송 받은 트래커는 피어들에게 Tracker Response 패킷을 전송하게 된다. 이 때 트래커는 피어들이 보낸 식별자에 대한 스웜이 존재하는지 파악하고, 존재하지 않는다면 다른 피어가 Tracker Request 패킷을 보낼 때까지 대기한다. 스웜이 생성되면 Tracker Request 동작을 수행하고 있던 피어 리스트를 피어들에게 전송한다. 이 때의 메시지 또는 패킷을 Tracker Response 라고 부른다. 피어 리스트는 기본으로 50 개의 피어 IP 주소로 구성되며, 만약 스웜에 저장되어 있는 피어들의 주소가 50 이상이라면 무작위로 피어 주소들을 리스트로 구성하여 전송하게 되고, 50 개 미만이면 모든 피어 리스트를 전송한다.

(4) 파일 공유 및 다운로드

트래커로부터 Peer 리스트를 받은 피어들은 서로에게 공유 파일의 식별자를 전송한다. 식별자를 전송 받은 피어들 중 파일 조각을 보유하고 있어, 해당 공유파일에 대한 공유가 가능한 피어라면, 동일한 식별자를 공유를 요청한 피어에게 전송하여 서로 간의 공유 세션을 형성한다. 이 때

피어들은 피어 리스트에 의해 최대 50 개의 세션을 동시에 형성하게 되고, 공유 파일은 일정 크기의 조각으로 나뉘어 공유된다. 그 후, 피어들은 자신이 가지고 있는 조각을 파악하고 자신이 가지고 있지 않은 조각을 다른 피어에게 요청하며, 자신이 가지고 있는 조각을 다른 피어가 필요로 하면 조각을 해당 피어에게 전송한다. 이와 같은 과정을 통해 다운로드와 업로드가 동시에 일어난다.

3.1. 시드 파일 구조

시드 파일은 일반적으로 토렌트 파일이라고도 불리며, 파일 공유를 위한 여러 메타데이터를 담고 있는 메타파일이다. 공유파일이 하나인 싱글 시드 파일과 공유파일이 여러 개인 멀티 시드 파일로 구분할 수 있으며, 각각의 구조는 아래 표와 같다. .

key	Type	Description
Info	Dictionary	싱글 시드 파일과 멀티 시드 파일의 구분 정보
-piece length	Integers	조각의 크기
- pieces	Strings	조각의 SHA1 값(20 Byte)
- name	Strings	공유 파일 이름
- length	Integers	공유 파일크기(Byte 단위)
Announce	Strings	트래커 주소
Announce-list (Option)	List	트래커 주소 그룹을 리스트 형식으로 가짐
Creation date (Option)	Integers	시드 파일의 생성 날짜, 1970 년 1 월 1 일 이후로 생성 날짜까지의 초를 값으로 가짐
Comment (Option)	Strings	시드 파일을 생성 시 생성자가 넣고자 하는 코멘트
Created by (Option)	Strings	시드 파일을 생성 시 생성자의 이름

표 2 - Single Seed File Structure

key	Type	Description
Info	Dictionary	싱글 파일 시드 파일과 멀티 시드 파일의 구분 정보
-piece length	Integers	조각의 크기(기본 256KByte)
- pieces	Strings	조각의 SHA1 값(20 Byte)
- name	Strings	공유 파일 이름.
- files	List (of dictionary)	여러 개의 파일이 속해 있는 디렉토리들의 리스트
- length	Integers	공유 파일 전체 크기
- path	List(of strings)	공유 파일들의 상대경로
Announce	Strings	트래커 주소
Announce-list (Option)	List	트래커 주소 그룹을 리스트 형식으로 가짐
Creation date (Option)	Integers	시드 파일의 생성 날짜, 1970 년 1 월 1 일 이후로 생성 날짜까지의 초를 값으로 가짐
Comment (Option)	Strings	시드 파일을 생성 시 생성자가 넣고자 하는 코멘트
Created by (Option)	Strings	시드 파일을 생성 시 생성자의 이름

표 3 – Multi Seed File Structure

파일의 구조는 위 표와 같이 정리 할 수 있으나, 정확한 오프셋은 정의할 수 없다. 이는 시드 파일에서 메타데이터를 다루는 방식 때문이며, 시드 파일의 정보들은 bEncoding 방식으로 다루어지기 때문이다. 위 표에서 언급한 각 Type들은 다음과 같은 방식으로 정의된다.

- Strings : <length of string>:<string>:

  Ex) 4:Test

- Integers :i<integer>e:

  Ex) i99e

- List :l[Strings1][Strings2][Strings3][...]e

  Ex) {spam, eggs, cheeseburger} ->l4:spam4:eggs12:cheeseburgere

- Dictionary : d[key1(strings)][value1(strings)][key2(strings)][value2(strings)][...]e

  Ex) {apple-red, lemon-yellow, violet-blue, banana-yellow}->

  d5:apple3:red6:banana6:yellow5:lemon6:yellow6:violet4:blueе

4. 망 내에서 최초 공유자 식별 방법

최초 공유자를 식별하기 위해서는 토렌트 네트워크 구성요소들인 피어와 트래커 간의 통신 과정을 이해해야 한다. 아래 그림은 트래커와 피어 간의 통신을 간단히 표현 한 것이다.

그림 2 - 트래커와 피어 간의 통신

Tracker Request에서 전달되는 정보들은 다음 표와 같다.

Parameter	Description
Info_hash	시드 파일에 포함되어 있는 공유 파일의 식별 값(SHA1)
Peer_id	클라이언트를 식별하기 위한 ID 로, 클라이언트 프로그램이 시작 될 때 생성 됨
Port	Tracker Request 메시지에서 사용되는 클라이언트의 TCP/UDP port number

Uploaded	클라이언트가 업로드 한 공유 파일의 총 크기(Byte 단위)
Downloaded	클라이언트가 다운로드 받은 공유 파일의 총 크기(Byte 단위)
Left	현재 남은 파일 다운로드 크기(Byte 단위)
Key (Option)	클라이언트의 IP 주소가 변경 되더라도 해당 클라이언트를 인식하기 위한 비공개 값(피어들 간의 공개 X)
Event	클라이언트의 파일 공유 상태를 의미 - Started : 파일 전송 시작 상태 - Stopped : 파일 전송 중지 상태 - Completed : 파일 전송 완료 상태
Numwant (Option)	전송 받을 피어 정보의 개수, 기본으로 50 개
Compact	전송 받을 피어 정보의 세부적인 사항을 설정하는 값 -Set 1: 피어들의 IP 주소와 Port 정보만 받음 -Set 0: IP 주소와 Port 정보를 포함해 피어들의 다양한 정보를 받음
No_peer_id	받는 정보 중 Peer-ID 정보를 받지 않겠다는 설정 값, Compact 의 값이 1 일 경우 무시
Ip (Option)	클라이언트의 IP 주소

표 4 - Tracker Request

Parameter	Description
Complete	현재 파일을 공유하고 있는 시더들의 수
Downloaded	해당 공유 파일의 다운로드 완료 횟수
Incomplete	현재 파일을 받고 있는 리처들의 수
Interval	클라이언트가 Tracker Request 를 전송하는 Interval(초 단위)
Mini interval (Option)	클라이언트가 Tracker Request 를 전송하는 최소 Interval(초 단위)
Peers	피어의 IP 주소 리스트

표 5 - Tracker Response

위의 표에서 알 수 있듯이 피어와 트래커는 상당히 많은 정보를 공유한다. 아래 그림은 실제 피어와 트래커가 주고 받는 패킷을 캡쳐한 모습이다.

그림 3 - Tracker Request/Response Packet

최초 공유자 식별을 위해 필요한 파라미터들은 다음과 같다.

Tracker Request	info_hash, uploaded, downloaded, left, key
Tracker Response	downloaded, Complete

표 6 – 최초 공유자 식별을 위해 필요한 파라미터

(1) Tracker Request

먼저 info_hash는 공유 파일의 식별을 위해서 필요하며, 최초 공유자라면 현재 사용자가 업로드 한 파일의 총 크기를 의미하는 uploaded의 값이 0보다 클 것이다.**(uploaded > 0)** 그와 동시에 현재 사용자가 다운로드 한 파일의 크기를 의미하는 downloaded 값이 0과 같을 것이다.**(downloaded == 0)** 이는, 최초 공유자는 자신이 완전한 파일을 이미 가지고 있어 다운로드를 하지 않기 때문이다. 그리고, 최초 공유자는 현재 남아있는 다운로드 파일의 크기를 나타내는 left 변수의 값 또한 0과 같을 것이다.**(left == 0)** 이유는 downloaded의 값이 0인 이유와 동일하다. 간단한 의사 코드로 정리하면 아래와 같다.

```
if( (uploaded > 0) and (downloaded ==0) and (left == 0) ) {
        최초 공유자;
} else {
        공유자 또는 다운로더
}
```

표 6에서 언급한 Key 변수의 경우는 트래커 서버 분석 시 필요한 값이다. 주소가 변경되더라도 피어를 식별하기 위한 값이므로, IP 주소가 변경되더라도 해당 식별 값을 가진 클라이언트의 통신을 모니터링 하여 확인하면, 실제 IP 주소를 파악 할 수 있을 것이다.

(2) Tracker Response

해당 공유파일의 다운로드 횟수를 나타내는 downloaded 변수의 값이 0이라면 해당 Response 메시지를 받는 클라이언트는 최초 공유자일 확률이 높다.**(downloaded == 0)** 그리고, 현재 공유파일을 공유하고 있는 시더들의 개수를 나타내는 complete 값 또한 0 또는 1일 경우 최초 공유자 일 확률이 높다.**(complete == 0 or complete == 1)** 그 이유는 자신이 처음으로 공유를 시작하게 되면 모든 피어들이 자신에게 조각을 요청 할 것이므로 공유자는 자신 혼자가 되기 때문이다. 간단한 의사 코드로 정리하면 아래와 같다.

```
if( (downloaded == 0) and (complete == 0 or complete == 1) ) {
        최초 공유자;
} else {
        공유자 또는 다운로더
}
```

위와 같은 조건의 통신을 하는 클라이언트를 망 내에서 모니터링만 할 수 있다면 조금 더 정확하게 검거 대상을 특정 지을 수 있을 것이다.

5. BitTorrent Client Artifact

이번에는 앞서 언급한 망 내에서 최초 공유자 식별 방법과는 달리 최초 공유자 또는 공유자를 검거 하였을 때 PC에서 획득 가능한 증거로 쓰일만한 흔적을 설명한다.

5.1. 레지스트리 흔적

토렌트 클라이언트는 스스로 레지스트리에 키를 만들어 흔적을 남기지는 않으므로, Windows에서 토렌트 클라이언트로 접근하는 레지스트리를 확인해야 한다. 보통 Windows에서는 MRUList로 최근 열린 파일들의 목록을 저장하고 있으며, 토렌트 시드 파일 또한 이 목록에 포함된다.

- HKCU₩Software₩Microsoft₩Windows₩CurrentVersion₩Explorer₩RecentDocs₩.torrent

위 레지스트리 키로 이동하여 보면 아래 그림과 같이 바이너리 데이터로 토렌트 파일의 정보가 저장되어 있는 것을 볼 수 있다.

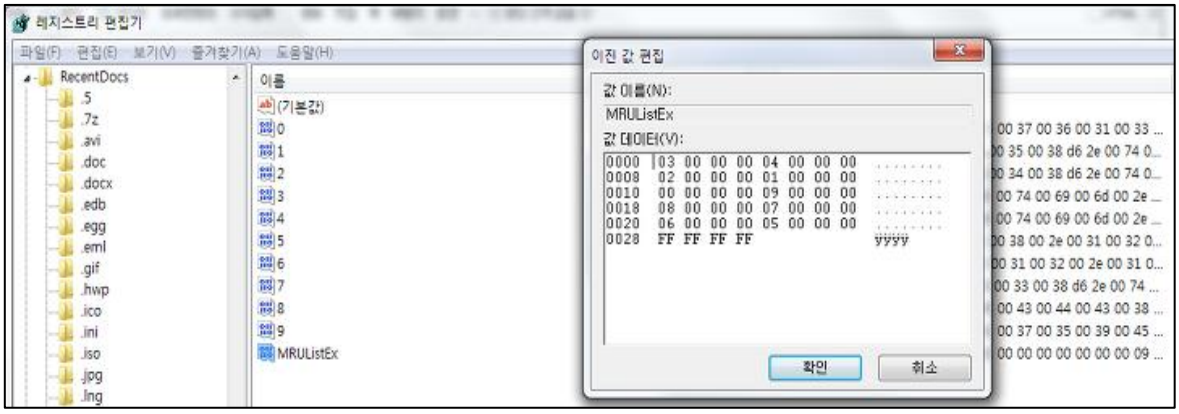

그림 4 - 최근 열어본 토렌트 시드 파일 목록

MRUListEx의 내용을 보면 각 바이너리 데이터들의 번호를 볼 수 있으며, 가장 최근의 파일은 0x00 오프셋에 있는 데이터 번호로 알 수 있다. 또한 해당 파일을 가장 최근에 열어본 시각을 파악하기 위해서는 해당 키의 LastWrittenTime을 통해 확인할 수 있다. 아래와 같이 간단하게 레지스트리 내보내기로 보면 시간 값을 확인 할 수 있다.

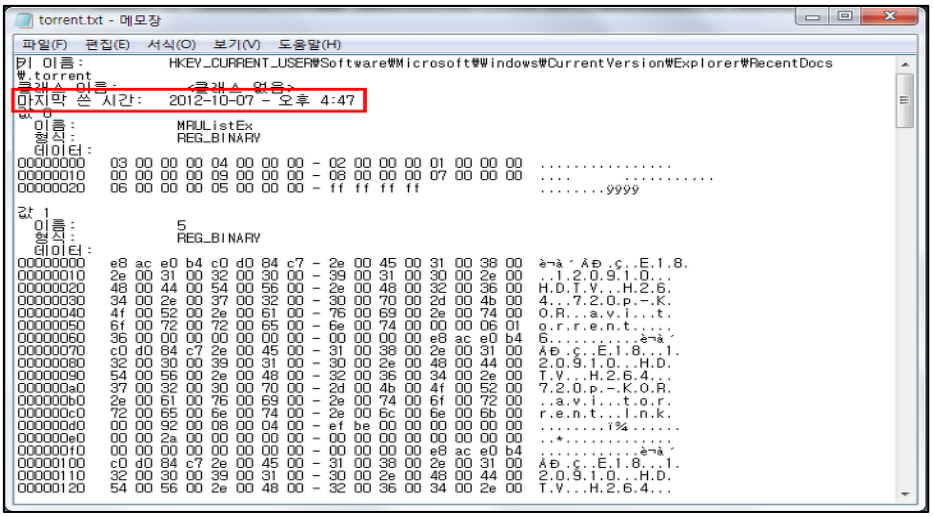

그림 5 - 가장 최근 토렌트 프로그램에 파일을 추가한 시간

그러나 해당 레지스트리 정보는 Windows 로컬 상에서 파일을 더블 클릭하여 열어야 생성되는 문제점이 존재한다. 일반적으로 사용자들은 파일을 다운로드 받아 열거나 실행 할 때 브라우저에서 제공하는 열기(IE) 또는 다운로드 완료목록(Chrome)을 이용한다. 이런 방식으로 해당 파일이 열리게 되면 해당 파일은 레지스트리에 추가 되지 않는다. 만약에 이런 습관이 있는 용의자의 컴퓨터를 조사한다면 레지스트리에서는 아무런 흔적도 발견하지 못할 수 있다. 그럼 어떻게 확실한 증거를 잡아야 할까? 아래 Windows의 Roaming Data를 확인해 보자.

5.2. Windows의 Roaming Data

Windows 는 계정마다 Roaming 폴더가 존재하며, 이 폴더에 다운로드 한 토렌트 시드 파일의 원본이 저장된다.

클라이언트	경로
uTorrent	₩Users₩<UserName>₩AppData₩Roaming₩uTorrent
BitTorrent	₩Users₩<UserName>₩AppData₩Roaming₩BitTorrent

표 7 – 클라이언트 별 토렌트 시드 파일 원본 저장 경로

국내에서 가장 많이 사용하는 uTorrent와 BitTotrrent 모두 동일한 경로에 시드 파일의 원본이 위치하게 되며, 아래 그림은 BitTorrent 클라이언트에서 특정 파일 다운로드 시도한 뒤 해당 경로를 확인한 화면이다.

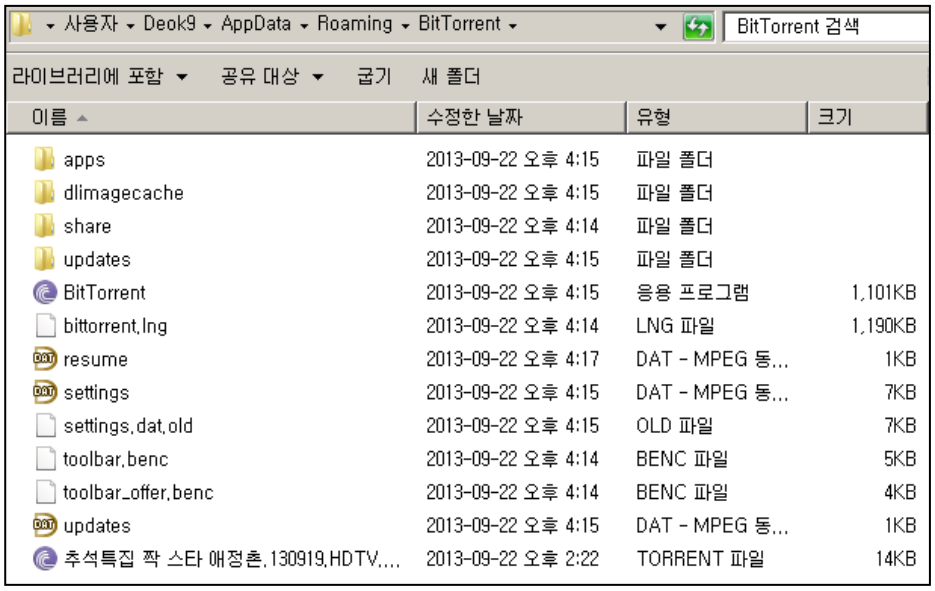

그림 6 – 원본 토렌트 시드 파일

Roaming 폴더의 시드 파일의 시간 정보를 통해 토렌트 파일의 다운로드 시작 시간을 알 수 있다. 아래 그림은 만든 날짜(C), 수정한 날짜(M), 액세스한 날짜(A) 가 모두 동일한 시드 파일이며, 이를 통해 토렌트 파일의 다운로드 시작 시간을 확인할 수 있다.

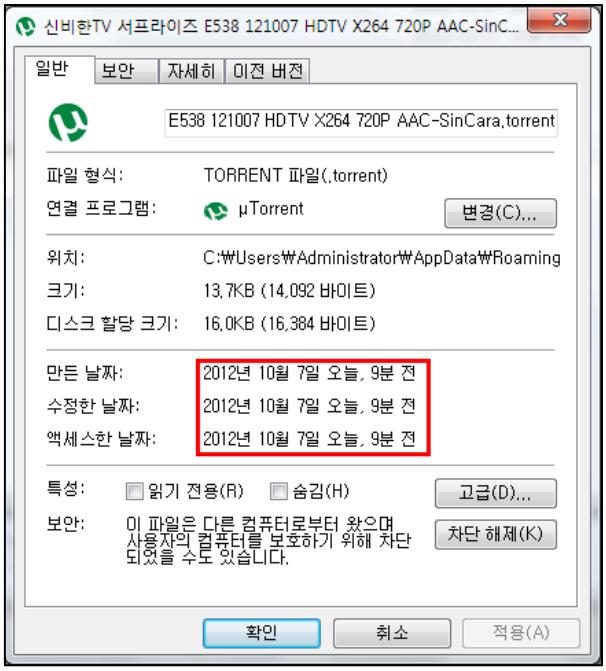

그림 7 - 토렌트 원본 시드 파일의 MAC Time

만약에 해당 파일이 한번 삭제되거나 다시 열렸을 경우에는 만든 날짜(C)와 액세스한 날짜(A)가 가장 최근에 열린 시간을 말해주며 수정한 날짜(M)가 처음 열렸던 시간을 말해준다.

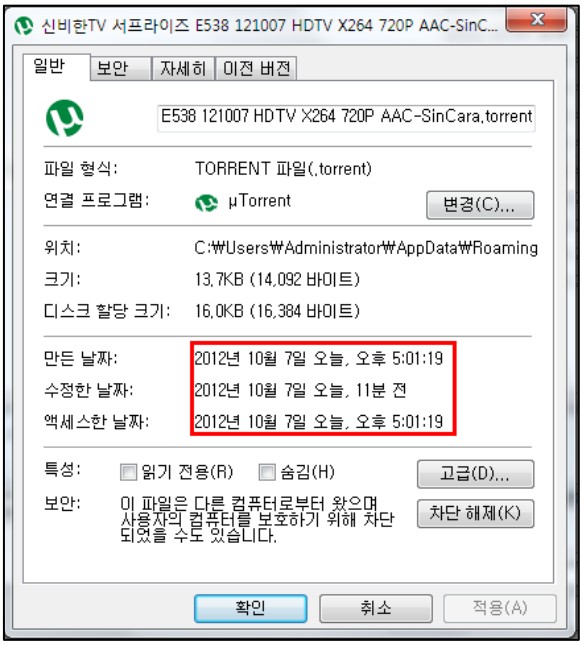

그림 8 - 파일이 다시 열렸을 때 MAC Time

원본 시드 파일이 저장되는 위치는 아래와 같이 클라이언트에서 옵션으로 설정이 가능하다.

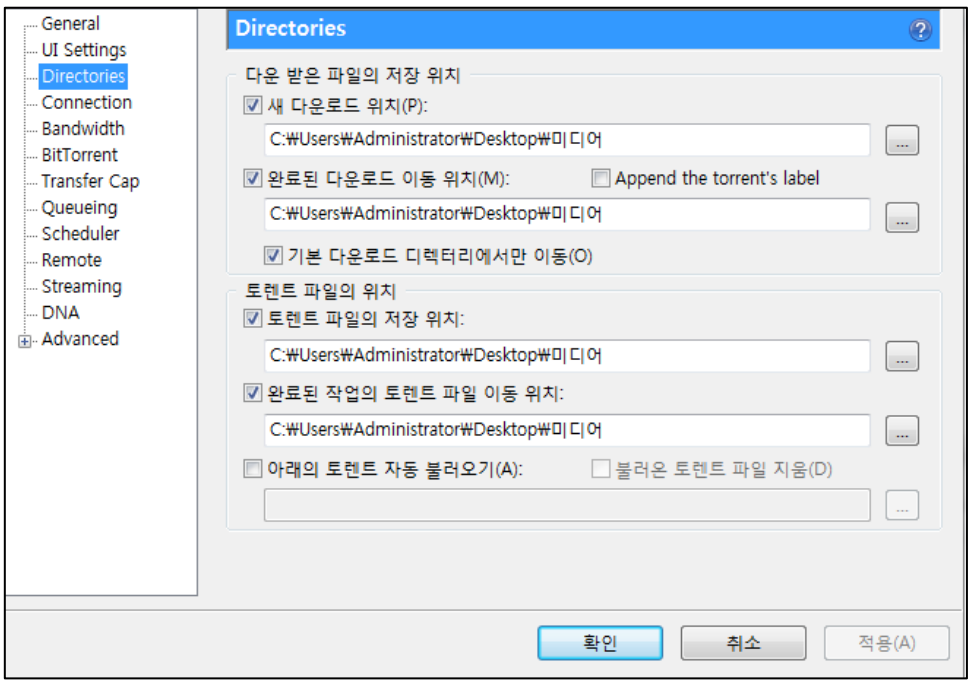

그림 9 - 클라이언트 프로그램 설정

위와 같이 원본 시드 파일의 경로는 사용자 임의로 수정 가능하므로, Roaming 폴더에 원본 시드 파일이 존재하지 않을 경우, Roaming 폴더 내의 'setting.dat' 파일을 분석하거나 분석 PC에 동일한 클라이언트를 설치 후 용의자 PC의 'setting.dat' 파일을 덮어 씌우면 쉽게 저장 위치를 알 수 있다.

추가로 '어떻게 원본파일이라고 확신 할 것인가?' 라는 의문이 들을 것이다. Roaming 폴더에 있는 시드 파일과 인터넷에서 다운받아 열었던 최초의 파일의 Hash 값을 비교해보면 이 의문은 쉽게 풀린다.

```
~/Desktop$ md5 *.torrent
MD5 (In Downloaded Seed File.torrent) = ebaf2e127328a7b65a4be0a7e703ac32
MD5 (In Roaming Folder Seed File.torrent) = ebaf2e127328a7b65a4be0a7e703ac32
```

그림 10 – 두 시드 파일의 Hash 값 비교

만약 Bitsnoop를 이용해 시드 파일을 다운받으면, 시드 파일의 제목이 해시 값으로 바뀌어 다운로드 될 때가 있다. 이러한 경우 Roaming 폴더에는 원본 파일의 제목으로 시드 파일이 저장되어 있으며, 아래와 같이 두 파일의 Hash 값 비교를 통해 동일 파일 여부를 알 수 있다.

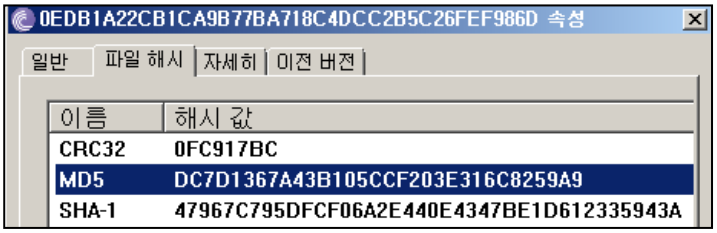

그림 11 – Bitsnoop를 통해 다운로드 한 시드 파일의 Hash 값

그림 12 – Roaming 폴더의 원본 파일 제목으로 저장된 시드 파일의 Hash 값

6. 결론

지금까지 BitTorrent와 관련하여 디지털 포렌식 관점에서 살펴보았다. 현재 우리나라에서는 수사기관에서 감시 사각지대였던 BitTorrent까지 수사영역을 확대시켰지만, 아직까지 수사대상들에 대한 정확한 기준이 존재하지 않아 억울한 피해자가 몇몇 발생하고 있다. 조금 더 깊이 있는 연구와 체계적인 시스템을 구축하여 정확한 수사와 검거가 이루어졌으면 한다.

[Reference]

- BitTorrent 프로토콜의 동작원리(Netmanias)

- Understanding of BitTorrent Protocol(F-Insight)

- Torrent File(File Formats Wiki)

- [생각] 토렌트 최초 유포자를 어떻게 검거 할까?(MaJ3stY)

- Torrent Forensics Artifact(MaJ3stY)