디지털 포렌식 증거 시각화 방안

Digital Forensics Evidence Visualization Plan

김예린, 김경숙, 이현섭, 채한빈*, 고석주**, 강대명***

Ye-Rin Kim, Kyung-Sook Kim, Hyun-Sub Lee, Han-Bin Chae*,
Seok-Joo Koh**, Dae-Myeong Kang***

dhm04022@naver.com, lovablekks@gmail.com, gustjq7027@naver.com, hanbinnnnnnnnn@gmail.com,
sjkoh@gmail.com, charsyam@naver.com

요 약

스토리지의 대용량화가 점점 진행되고, 국내법에 따른 선별수사 정책에 따라 실제 압수 시에 필요한 데이터를 찾는데 시간을 줄이는 것이 중요한 시대가 되었다. 하지만 아직 오픈소스 디지털 포렌식 툴은 현저히 부족하고, 기존의 툴은 분석의 효율성과 접근성이 떨어져 여러 문제가 있다. 이에 본 논문은 FAT32 파일시스템 기반의 디스크 이미지를 분석하여 레지스트리, 웹 히스토리, 이미지, 문서 등과 같은 파일의 세부 정보를 추출하여 사용자에게 필요한 정보를 시각화를 통해 직관적으로 제공하는 것을 목표로 한다. 이를 위해 PyQt와 wx를 사용한 UI 구현을 통해 특정 시간대에 변화되거나 접근된 파일들의 접근 빈도 등을 보여줌으로써, 선별 압수 시에 도움이 되는 기능을 제공한다. 또한 수사 단계에서 사용자 행위 분석 시 효율성 증대를 위하여 다양한 프로토타입을 만들어 비교하는 단계를 거쳐 가장 효과적인 시각화 툴을 연구 개발 하였다.

키워드 : 디지털포렌식, 시각화, FAT32, 레지스트리, 웹 히스토리

I. 서 론

정보화 시대가 본격화 된 만큼 사이버 범죄의 심각성이 사회에 대두되고 있다. 사이버 범죄들이 증가하며 해당 범죄의 중요한 단서나 증거가 디지털화되어 있는 경우는 늘고 있지만 이를 분석하여 유의미한 정보를 추출하고 이를 통해 사건의 단서를 찾아내는 것은 쉽지 않다.

사이버 범죄에 사용된 컴퓨터 데이터를 수집, 보호, 분석, 해석하는 포렌식 수사는 방대한 양의 데이터를 처리해야 한다. 하지만 분석 대상 장치의 수와 저장장치 용량의 증가로 인해 디스크 이미지 생성과 분석에 많은 시간이 요구되고, 모든 운영체제와 파일에 대한 분석이 어려울 뿐만 아니라 추출한 데이터 간 연관관계에 대한 분석도 필요하다. 따라서 수사관이 포렌식 수사 지원 소프트웨어를 사용하더라도 상당히 많은 시간이 소요되고 에러가 발생할 수 있다. 이 컴퓨터 포렌식 수사의 문제점 해결에 관해서는 Teerlinket al.(2006)에 의해서, 많은 정보 속에서 원하는 정보를 찾아내기 위한 방법으로 가장 유용하게 쓰이는 정보 시각화가 그 해결책이 될 수 있음을 시사한 바 있다[1].

본 논문에서는 USB와 같은 저용량 저장매체에서 주로 사용되는 FAT32 (File Allocation Table) 파일 시스템 기반의 디스크 이미지 파일 내부 정보를 분석하는 것을 목표로 한다. 이를 위해 우선적으로 FAT32 파일 시스템의 구조에 대한 이해를 바탕으로 한다. 더불어 실제 디지털 포렌식 수사 시 확보한 전체 데이터 중 의미 있는 증거를 직관적으로 파악할 수 있는 효율적인 정보 시각화 방안을 연구한다. 이를 통해 디스크 이미지 파일로부터 데이터를 추출하여 가공 과정을 거치고, 타임 라인을 활용하여 시간 및 빈도순으로 추출 정보를 보여줌으로써 사용자 행위 분석을 빠르고 효율적으로 할 수 있도록 한다. 또한 레지스트리, 이벤트 로그, 웹 히스토리 등 서로 다른 유형의 자료에 대하여 각각의 포렌식 도구를 사용하지 않고, 하나의 도구에서 분석을 수행할 수 있도록 하여 조사 시간을 단축할 수 있도록 한다.

*경북대학교 컴퓨터학부
**경북대학교 컴퓨터학부 (교신 저자)
***BeNX (교신저자)

II. 본 문

1. 디스크 이미지 분석의 접근법

1-1. FAT32 파일시스템

본 연구는 FAT32 파일 시스템 기반 디스크의 이미지를 분석하여 원하는 자료를 추출하는 기능의 툴을 제작한다. FAT 파일시스템의 전체적인 구조는 다음과 같다.

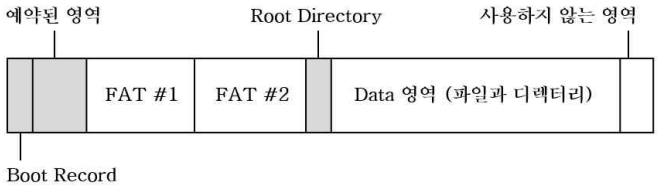

그림 1. FAT32 파일 시스템 구조도

VBR(Volume Boot Record)은 볼륨의 첫 번째 섹터에 해당하는 영역이며 FAT 파일시스템의 여러 설정 값들이 담겨있다. FAT 영역은 클러스터들을 관리하는 테이블이 모여 있는 공간이다. Data 영역에는 Directory Entry라 불리는 구조체들을 담고 있는 디렉터리와 파일이 저장된다[2].

Directory Entry의 시작 클러스터 번호를 참조하여 FAT 영역에서 해당 디렉터리의 정보가 저장된 시작 클러스터부터 마지막 클러스터의 번호까지를 알아낸다. 디렉터리의 정보가 저장된 클러스터들의 번호와 부트섹터에서 저장된 파일시스템의 Byte Per Sector , Sector Per Cluster 등의 정보를 통해 해당 디렉터리와 파일의 정보를 읽어온다.

먼저, Disk Image File을 읽어온 후 해당 파일의 VBR 영역을 분석한다. FAT 파일시스템의 VBR에 담긴 정보는 아래 표 1과 같다.

내용	시작 위치	사이즈
Byte per Sector(BPS)	11	2
Sector per Cluster(SPC)	13	1
Reserved Sector Count	14	2
Number of FAT	16	1
FAT Size 32	36	4
Root Directory Cluster	44	4

표 1. FAT32 VBR 항목

그 다음, Reserved 영역에 저장되어 있는 메타데이터를 기반으로 FAT 영역의 시작 주소와 데이터 시작 위치를 계산한다. First Data Sector = Reserved Sector Count + FAT Size 32 * Number of FAT이므로 Data 시작 영역의 Byte Offset은 First Data Sector * BPS로 구한다.

이후 Data 시작 영역의 Byte Offset을 사용하여 Directory List에서 출력할 가장 상위 디렉터리인 Root Dir의 위치를 계산한다.

Data 영역에서 Root Dir 위치 = Data 시작 영역의 Byte Offset + (Root Directory Cluster - 2) * SPC * BPS이다.

위의 계산을 바탕으로 데이터 영역에 들어와 분석을 하는데 먼저 일반 파일의 분석은 표 2의 Directory Entry 참조 항목에 해당하는 각 파일의 세부 정보를 추출하여 File List 출력이나 선택 옵션 별 기능 등을 수행할 때 해당 정보를 사용할 수 있도록 리스트를 만들어 저장해놓는다.

내용	시작 위치	사이즈	비고
Name	0	8	해당 파일의 이름을 나타내며 파일의 삭제유무를 판단한다.
Ext	8	3	파일의 확장자를 파악하고, 해당 속성을 통해 파일 포맷에 따라 분석 후 시각화한다.
Attribute	11	1	해당 Entry가 파일인지 디렉터리인지 LFN인지 파악한다.
Create Date	16	2	파일 생성 일자 순으로 정렬 가능하게 한다.
Last Access Date	18	2	최근 접근한 날짜를 나타내며 최근 접근된 순서로 파일을 정렬할 수 있게 한다.
First Cluster High 2 Bytes	20	2	해당 파일의 시작 클러스터 번호의 상위 2byte를 나타낸다.
First Cluster Low 2 Bytes	26	2	해당 파일의 시작 클러스터 번호의 하위 2byte를 나타낸다.
File Size	28	4	해당 파일의 크기를 나타낸다.

표 2. Directory Entry

디렉터리 및 파일 데이터의 시각화를 위해 기준이 되는 디렉터리의 시작 클러스터 번호를 파악하여 FAT 테이블을 참조, 디렉터리의 데이터를 전부 읽어온다. 읽어온 정보로 하위 디렉터리 및 속한 파일을 트리 형태로 출력한다.

트리에서 원하는 파일 선택 시 기본적으로 해당 파일의 데이터를 Offset에 따른 Hex값 형태로 출력한다. 추가적으로 하나의 파일을 선택할 시 새 창을 띄워 해당 파일의 분석 데이터에 대한 상세 정보를 시각화된 빈도 그래프와 ASCII view로 제공한다.

1-2. 레지스트리

레지스트리(Registry)란 윈도우에서 사용하는 시스템 구성 정보를 저장한 데이터베이스이다[3]. 레지스트리 하이브 파일을 분석하기 위하여 우선 레지스트리 하이브 파일의 구조를 설명하면 다음과 같다.

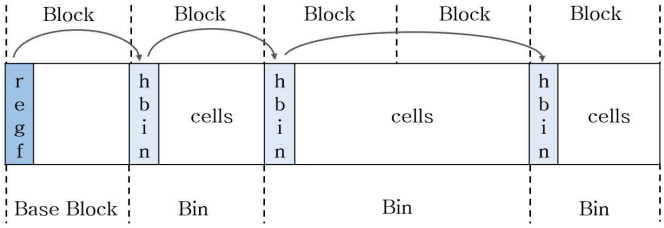

그림 2. 레지스트리 하이브 파일 구조도

Block의 크기는 4096byte이고 위에서 말한 FAT32 파일시스템 구조도에서 클러스터와 비슷한 개념이다. 그 중 첫 번째 Block이 Base Block이며 전체 레지스트리 하이브 파일에 대한 기본적인 정보를 저장하고 있다. Bin은 레지스트리 세부 정보를 담고 있는 서로 다른 Cell들의 집합이며, 크기에 따라 여러 블록을 내부에 가질 수 있다[4]. Bin의 제일 앞부분은 Hbin으로 Bin의 헤더정보를 가지고 있다.

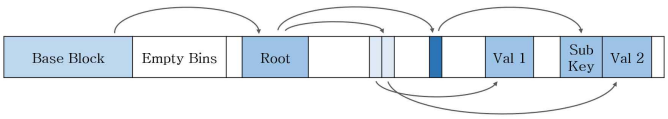

그림 3. 레지스트리 정보 탐색 방법

대략적으로 레지스트리 분석 방법을 설명하자면, 다음과 같다. Base Block에서 Root Key에 대한 Offset을 가져온다. 그 다음 Root Key Cell에 가서 Subkey List가 있는 Offset으로 이동한다. Subkey List에 있는 각 Key의 Offset을 이용하여 Key 정보를 확인하고 우리가 필요한 정보를 DB에 추가한다. Key 정보가 있는 곳에서 Value List Offset을 찾아서 확인 후 Value List에 있는 각 Value의 Offset을 찾아서 Value의 필요한 정보를 DB에 저장한다.

내용	시작 위치	사이즈
Signature(regf)	0	4
Time Stamp	12	8
Root key offset	36	4
File name(Unicode)	48	64

표 3. Base Block 항목

먼저 Root Key의 위치를 알기 위해서는 표 3에 있는 것과 같이 Base Block의 Root Key Offset을 가져온다. 여기서 실제 Root Key는 Base Block의 크기를 제외한 Offset이 적혀 있기 때문에 Base Block Size (4096byte) + Root Key Offset을 계산한 값에 있다.

내용	시작 위치	사이즈	비고
size	0	4	key의 크기
Signature (nk)	4	2	해당 key의 타입
Flag	6	2	2c이면 root key, 20이면 subkey
Time Stamp	8	8	마지막 수정날짜
Parent Key Offset	20	4	해당 parent key가 있는 offset
Number of Subkey	24	4	subkey의 갯수
Subkey List Offset	32	4	subkey list가 있는 offset
Number of values	40	4	value의 갯수
Value List Offset	44	4	value list가 있는 offset
Key Name	80	variable	key 이름

표 4. (Root) Key Cell

Root Key도 보통 Key Cell과 같이 표 4의 정보를 가지고 있다. 그 중 Subkey List Offset 정보를 가져와 Subkey List Offset + Base Block Size (4096byte)를 계산하면 Subkey List가 있는 Cell에 도착한다.

내용	시작 위치	사이즈
size	0	4
signature (“lf” or “lh”)	4	2
Number of Subkey	6	2
offset to Subkey	8	4
Subkey name or checksum	12	4

표 5. Subkey Cell (lf)

표 5와 같이 Subkey Cell은 lf, ri 두 가지의 Signature로 구분된다. lf는 해당 Key 마다의 Offset을 가져서 바로 Key cell로 넘어갈 수 있다. 반면 ri는 Subkey의 개수 정보와 Subkey Offset만을 가진다.

Subkey의 Signature를 찾고 나서 Key(nk) Offset을 알 수 있다. 앞과 마찬가지로 Key Offset + Base Block Size (4096byte)를 계산하여 이동하면 Key Cell이 있는 곳에 도착한다.

Key Cell에서 우리가 찾으려고 하는 Key의 모든 정보는 표 4와 같다. 여기서 Time Stamp와 필요한 정보들을 추출하여 우리가 사용하는 시간처럼 변환해서 DB에 저장한다. 여기서 DB에 저장한 정보를 이용하여 시간별 빈도 시각화를 진행한다.

그 다음 Value 값을 찾기 위해서는 표 4의 Value List Offset에 Base Block Size (4096byte)만큼 더한 다음 Value List에 있는 Value의 Offset을 찾아 표 6의 Value 정보를 DB에 저장한다.

내용	시작 위치	사이즈
size	0	4
signature(vk)	4	2
Value Name Length	6	2
Value Data Length	8	4
Data Type	11	1
Value data of Offset to Value Data	12	4
Value Type	16	4
Named Value	20	2
Value Name	24	Value Name Length

표 6. Value Cell

1-3. 웹 히스토리

또한 본 논문은 웹 히스토리 파일을 활용하여 사용자의 검색 기록, 방문 URL, 접속 시간 등의 정보를 추출한 후 인터넷 사용에 관한 빈도 시각화를 제공한다.

먼저 웹 히스토리 정보는 위와 다르게 우리가 직접 분석할 필요 없이 윈도우 파일 내부에 저장되어있다. History 파일은 각 브라우저마다 서로 다른 경로에 저장되어 있는데 논문에서는 대표적으로 Chrome의 History 파일을 예를 들어 설명한다. Chrome의 History 파일은 C:\Users\사용자이름\AppData\Local\Google\Chrome\User Data\Default\에 있는데 이것은 기본적인 DB 파일로, 일반적인 DB 분석 툴을 사용하면 쉽게 확인이 가능하다. 우리는 DB Browser for SQLite를 이용하여 DB의 내용을 확인했다.

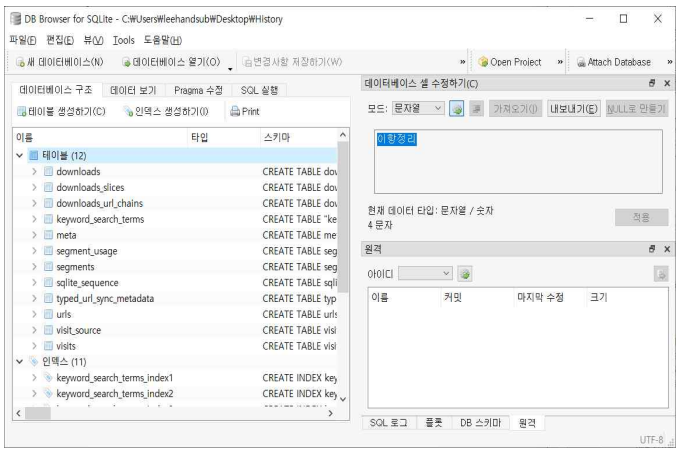

그림 4. Chrome의 History 파일

그림 4처럼 다운로드 기록, 검색어, 방문 URL, 접속 시간과 빈도가 DB형태로 저장되어 있는 것을 확인할 수 있다. 특히 우리는 URL 기록과 해당 URL에서 어떤 키워드를 검색했는지, 언제 방문했는지에 대한 시간 정보를 시각화시켜 제공한다.

2. 시각화방안

2-1. 배경

본 연구에서는 분석한 사용자 기록을 가장 효과적으로 보여줄 수 있는 시각화를 제공하는 것을 최우선의 목표로 한다. 데이터 시각화(Data Visualization)는 데이터 분석 결과를 쉽게 이해할 수 있도록 시각적으로 표현하고 전달하는 과정을 말한다. 데이터 시각화의 목적은 도표(Graph)라는 수단을 통해 정보를 명확하고 효과적으로 전달하는 것이다[5]. 결과를 글이 아닌 그림으로 보여줌으로써 사용자는 전체적인 분석 결과를 직관적으로 파악할 수 있으며 빈도 변화를 쉽게 인지할 수 있다. 이를 통해 수사 시 단서의 연결성을 찾는 데에 걸리는 시간을 단축할 수 있어 기존 포렌식 툴 사용에 비해 굉장히 효율적이다. 이에 본 논문에서는 추출 정보를 가장 잘 보여줄 수 있는 그래프를 연구하여 해당 방식으로 프로토타입을 제작한 후 개발을 진행하는 순서로 진행한다.

2-2. 시각화 디자인

본격적으로 시각화 기능을 개발하기에 앞서 간단한 프로토타입을 제작해보았다. 기본적으로 그래프는 시간을 기준으로 빈도를 나타내주는 역할이기 때문에 x축은 시간(Time), y축은 빈도(Frequency)로 설정한다.

그림 5. Time Filter, Type Filter

또한 그림 5와 같이 필터 기능을 통해 원하는 시간대나 타입의 사용 기록만을 선택하여 보는 것이 가능하도록 하여 수사 시 필요한 정보만을 선택적으로 확인하고 빠르게 분석할 수 있게 하였다.

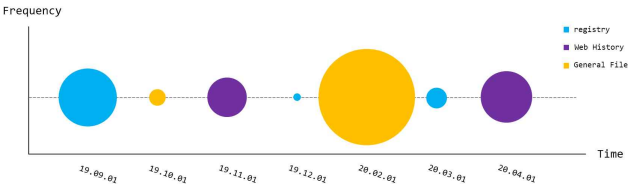

그림 6. All User Records

그림 6의 그래프는 특정 시간에 대하여 전체적인 사용자 기록의 빈도를 보여준다. 이를 통해 어떤 시간에 가장 많이 데이터의 변화나 접근이 있었는지에 대하여 빠른 파악이 가능하다. 또한 어떤 타입 별로 색깔 구분을 두어 해당 시간에 어떤 타입에 가장 접근 빈도가 높았는지 또한 바로 확인할 수 있다. 예를 들어 위 그래프에 표시된 19.09.01~20.04.01 사이의 기간을 기준으로 20.02.01에 표시된 원이 가장 크므로 이 때 가장 많은 접근이 일어났다고 예측할 수 있고, 이때 색은 노란색이므로 그 중에서도 일반 파일에 대한 접근 빈도가

높았을 것이라는 점을 바로 파악할 수 있다. 따라서 수사 시 집중적으로 조사해야하는 시간대를 특정지어주기 때문에 단서를 찾는 시간을 단축하는 데에 도움을 줄 수 있다.

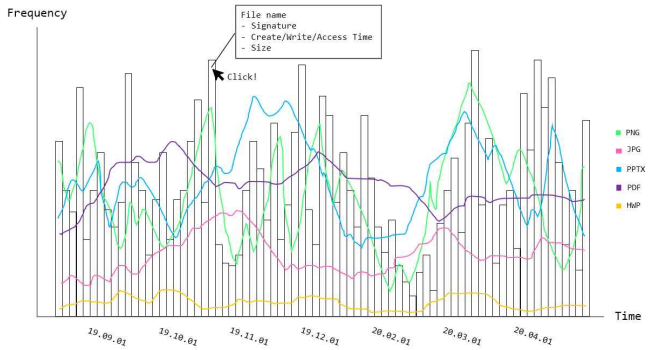

그림 7. Selected User Records (General File)

그림 6의 그래프는 타입 구분 없이 모든 기록에 대한 빈도를 합쳐서 보여줬다면 Selected User Records는 사용자가 선택한 특정 타입 (레지스트리, 웹 히스토리, 일반 파일)의 정보만을 보여준다. 특정 타입에 대한 정보만을 볼 경우 조금 더 세분화되고 자세한 타임라인의 변화 및 세부 정보를 볼 수 있다. 그림 7에서는 일반 파일을 예를 들어 보여준다. 일반 파일도 PNG, JPG, PPTX 등 다양한 시그니처를 가지는 파일들이 존재하기 때문에 색깔로 구분하고, 그들의 전체적인 빈도는 막대그래프로 표현하였다. 또한 특정시간에 대한 막대그래프를 클릭할 시 그 시간과 관련 있는 모든 파일들의 세부 정보를 추가적으로 띄워주어 전제적인 빈도의 흐름을 보면서 간편하게 세부 정보 또한 파악할 수 있도록 한다.

III. 시스템 결과

우선 테스트 진행을 위하여 빈 USB에 레지스트리 하이브 파일, 웹 히스토리 DB 파일, 일반 파일을 넣은 후 디스크 이미지 덤프 생성 과정을 통해 FAT32 기반의 디스크 이미지 샘플 파일을 제작했다. 이를 대상으로 Windows(OS) 환경에서 테스트를 진행하였으며, 그 결과는 다음과 같다.

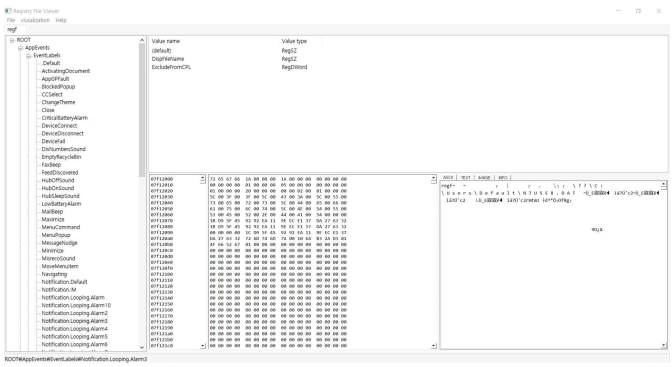

그림 8. 시스템 작동 화면 (main)

그림 8은 PyQt와 wx를 사용하여 구현한 디지털 포렌식 증거 시각화 툴이다. 파일 오픈을 통해 디스크 이미지 파일을 열면 화면 좌측에 내부 디렉터리 구조를 트리 형태로 출력한다. 여기서 원하는 디렉터리를 클릭하면 해당 디렉터리 내부에 있는 파일 리스트 및 파일 세부 정보를 우측 상단에 출력한다. 일반 파일 디렉터리를 선택할 경우 디렉터리 내부에 있는 PNG, JPG, PDF 등 일반 파일들의 이름, 시그니처, 데이터 정보를 출력하고, 레지스트리 하이브 파일 또는 하위 Key를 선택할 경우 해당 Key의 Value값에 대한 이름, Value Type, 데이터 정보를 출력한다. 마지막으로 웹 히스토리 디렉터리를 선택할 경우 Chrome, Internet Explorer, Whale의 히스토리 DB 파일들에 대한 이름, 타입(DB), 데이터 정보를 출력한다.

우측 상단에서 파일 또는 Value를 선택할 경우 하단에 해당 파일 또는 Value의 시작위치부터의 Offset과 Hex값이 매번 갱신되며 출력된다.

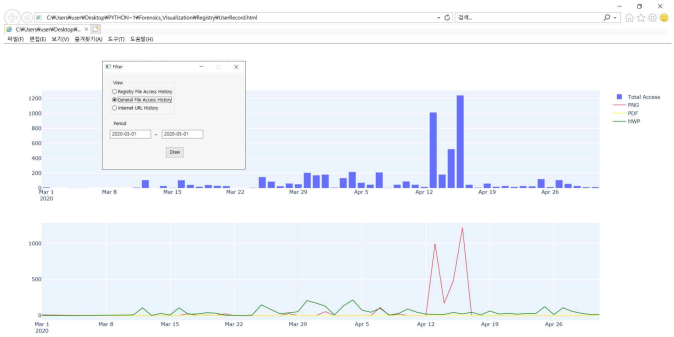

그림 9. 시스템 작동 화면 (visualization 클릭 시)

상단의 메뉴바에서 visualization 버튼을 클릭하면 시각화 그래프를 출력하는 새로운 HTML 창이 뜬다. 그림 9와 같이 원하는 시간과 타입을 설정할 수 있는 필터가 뜨고, 여기서 사용자가 원하는 시간을 설정하면 해당 시간에 대한 모든 사용자의 접근 기록의 빈도가 Total Access라는 막대그래프 형태로 출력된다. 그리고 선택한 타입에 따라 그에 대한 정보에 대한 세부 빈도와 상세 정보가 Total Access 아래에 꺾은선그래프 형태로 출력된다. 이 그래프를 통하여 사용자는 디스크 이미지 내부 사용자의 기록을 시각적으로 확인 가능하다.

IV. 결 론

본 논문에서는 일반 파일, 레지스트리 하이브 파일, 웹 히스토리를 포함하는 FAT32 파일 시스템 기반의 디스크 이미지 디지털 포렌식 분석 시각화 툴을 구현하였다. 디스크 이미지를 분석하여 정보를 추출하였고 추출한 정보를 DB에 저장 후 각 저장된 시간 정보들을 전체적으로 한 눈에 파악할 수 있도록 하고, 사용자가 선택한 타입의 정보만을 자세히 볼 수

있도록 타입 별 상세 그래프로 시각화한다. 또한 개발한 포렌식 분석 툴을 오픈소스로 공개함으로써 디지털 포렌식 분석 툴에 대한 접근성을 증가시켜 전문 포렌식 분석가뿐만 아니라 일반 사용자들도 쉽게 툴을 사용하여 분석 정보를 한눈에 볼 수 있게끔 한다. 추후 해당 툴에서 같은 시간동안 인터넷과 파일 접근, 프로그램을 같은 시간 내에 사용한 경우를 머신러닝을 통해 학습을 시켜 수사를 할 때 특정 시간 내 사용 기록의 연관성을 보여주는 인공지능 기능을 구축할 것으로 기대된다.

※ 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음 (2015-0-00912)

참 고 문 헌

[1] 효율적인 컴퓨터 포렌식 수사를 위한 정보 시각화 기법 연구 (정윤석 외 5인), 고려대학교 정보경영공학과, 2008.
[2] FAT 파일시스템 분석을 통한 디지털포렌직에 관한 연구 (장부성), 한서대학교 대학원, 2011.
[3] 레지스트리, 해시넷. http://wiki.hash.kr/index.php/%EB%A0%88%EC%A7%80%EC%8A%A4%ED%8A%B8%EB%A6%AC
[4] Winndow forensic Registry 구조 by 해커남, 2013. https://m.blog.naver.com/PostView.nhn?blogId=ifkiller&logNo=70157957567&proxyReferer=https:%2F%2Fwww.google.com%2F
[5] 데이터 시각화(data visualization), 위키백과. https://ko.wikipedia.org/wiki/%EB%8D%B0%EC%9D%B4%ED%84%B0_%EC%8B%9C%EA%B0%81%ED%99%94