BoB 11기 I WATCH U

스마트워치 분석도구

사용 설명서

BoB 11기 I WATCH U

목차

0. 도구 소개

I WATCH U의 스마트워치 포렌식 도구는 스마트 워치의 발전과 보급속도가 빨라지고 있는 추세이지만, 그에 반해 관련된 연구는 많이 이루어지지 않았습니다. 스마트폰 포렌식과 스마트 워치 포렌식 비교 시 검색 및 연구의 확연한 차이가 있음을 알게 되었고, 스마트폰보다 정확한 동작이나 건강정보 등 다른 스마트 기기에서 알 수 없는 정보를 의

Windows 기반의 도구로 제작되었습니다.
따라서 해당 문서에서는 Windows 10 환경을 기준으로 설명합니다.

사전 필요한 프로그램

- 버전 22H2(OS빌드 19045.2251) 이상
- Docker Desktop(Docker Desktop - Docker)
- Python3

> 다음의 도형 위에는 터미널 내 명령어 또는 코드와 관련되어 정리되어 있습니다.
> 사용자가 입력해야 하는 부분에 관련해서는 [***다음과 같이***] 표현되어 있으니 참고하시어 사용자 고유의 값으로 작성해 명령어를 입력하시면 됩니다.

1. 도구 설치

프로그램 다운로드 링크 :

https://github.com/YooDongseon/IWATCHU/tree/main/Smartwatch_Forensics_Tool

■ 사전 필요한 과정

(1) 스마트 워치의 OS별 다음과 같은 연결 상태를 준비해주시길 바랍니다.

A. Tizen OS (ex. Samsung Watch 3)

Tizen OS의 경우 디버깅을 통하여 추출한 데이터와 갤럭시 웨어러블 앱을 이용하여 추출한 데이터, 두 가지를 모두 사용합니다.

디버깅 레벨에서, Tizen 디버깅을 위해서는 sdb를 설치하여야 합니다. SDB는 Tizen Studio를 설치하면 자동으로 같이 설치됩니다. 변동 사항이 있을 수 있으니 각주 SDB 공식 문서를 참고하여 주시기 바랍니다.[1]

SDB를 이용해 기기와 연결한 이후, 덤프 파일이 저장될 폴더 디렉터리로 이동해(아래 사진에서는 tst)다음 명령어를 이용하여 가져올 수 있는 모든 데이터를 덤프합니다.

아래와 같은 명령어를 사용하는 이유는, 덤프 중간에 멈추는 특정 디렉토리들을 생략하기 위해서입니다.

표 1 디버그 레벨 덤프 명령어

```
$array=@("bin","boot","csa","dev","etc","home","lib","lost+found","media","mnt","nuget","opt","root","run","sbin","srv","tmp","usr","var")
for($i=0;$i -lt 18;$i++) {echo $array[$i]; sdb pull /"$($array[$i])" .\"$($array[$i])";}
```

```
PS C:\Users\Post G\Desktop\tst> $array=@("bin","boot","csa","dev","etc","home","lib","lost+found","media","mnt","nuget","opt","root","run","sbin","srv","tmp","usr","var")
PS C:\Users\Post G\Desktop\tst> for($i=0;$i -lt 18;$i++) {echo $array[$i]; sdb pull /"$($array[$i])" .\"$($array[$i])";}
```

그림 1 디버그 레벨 덤프 명령어 사용 예시

애플리케이션 레벨에서는 연결된 스마트폰의 “Galaxy Wearable” 앱의 버전 정보 탭에 들어간 뒤, Galaxy Wearable 로고를 5회 누르면 Get Dump 모드로 진입할 수 있습니다. 이 때 Run Gear Dump 기능을 이용해 휴대폰 상에 덤프할 수 있습니다. 이렇게 가져온 데이터를 이용하여 애플리케이션 레벨 분석을 진행할 수 있습니다.

[1] https://docs.tizen.org/application/tizen-studio/common-tools/smart-development-bridge/

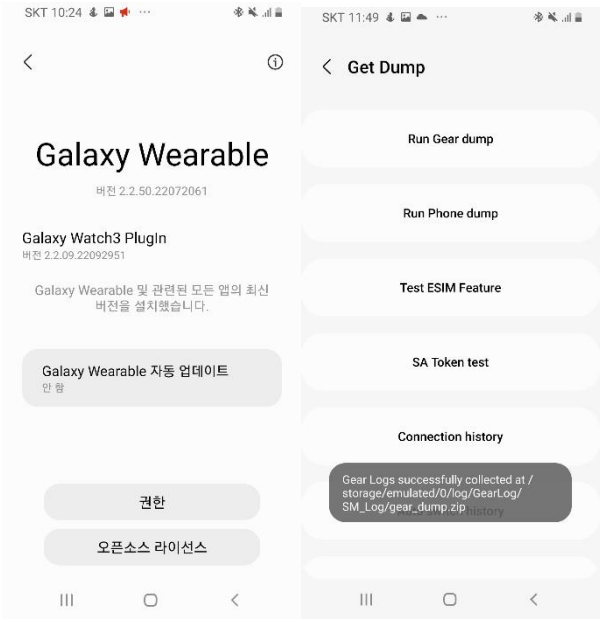


그림 2 애플리케이션 레벨 덤프 모드 진입 화면

B. Wear OS (ex. Samsung Watch 5)

Wear OS(Wear OS powered by SAMSUNG) 의 경우 ADB를 활용하여 데이터를 추출하고 분석합니다. adb.exe connect 명령어를 통해 미리 PC와 Galaxy Watch 5를 연결해주시기 바랍니다.

C. Watch OS (ex. Apple Watch)

Apple Watch는 백업 데이터를 기준으로 분석을 진행합니다. 아이폰과 애플워치의 연결을 해제하며 아이폰에 애플워치의 백업데이터가 남아있는지 확인한 후, 아이폰을 아이튠즈를 이용해 데이터를 백업해주시기 바랍니다.

(2) Python 설치

해당 문서는 Python 3 버전을 사용하였습니다. 도구 호환성을 위해 Python 3 버전 설치 및 사용을 추천드립니다.

(3) 패키지 설치

도구를 사용하기 위해 사전적으로 python 패키지 설치를 필수적으로 필요로 합니다.

위의 git 주소에서 requiremnets.txt를 다운 받은 후 requirements.txt가 있는 경로에서 다음의 명령어를 사용하면 도구에 사용하는 모든 패키지 설치가 완료됩니다.

```
> pip install -r requirements.txt
```

필요로 하는 패키지들에 대한 정보는 다음과 같습니다.

branca==0.6.0	contourpy==1.0.6	Pillow==9.3.0	packaging==22.0
certifi==2022.12.7	cycler==0.11.0	plum-py==0.8.5	pandas==1.5.2

charset-normalizer==2.1.1
matplotlib==3.6.2
numpy==1.23.5
idna==3.4
install==1.3.5
fonttools==4.38.0
elastic-transport==8.4.0
elasticsearch==7.17.8
elasticsearch-dsl==7.4.0
exif==1.4.0
folium==0.13.0
requests==2.28.1
pure-python-adb==0.3.0.dev0
pyparsing==3.0.9
python-dateutil==2.8.2
pytz==2022.6
six==1.16.0
urllib3==1.26.13
Jinja2==3.1.2
kiwisolver==1.4.4
MarkupSafe==2.1.1

2. 도구 실행

(1) Docker Desktop 설치 완료 후 회원가입 및 로그인

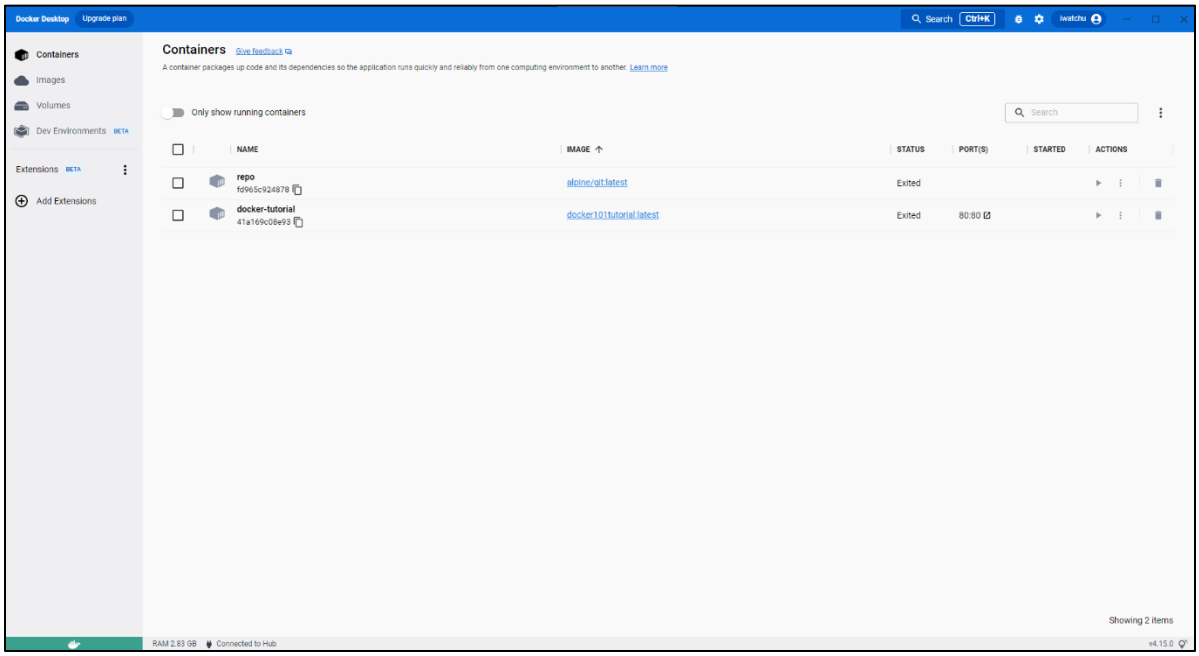


그림 3 _ 설치, 회원가입 및 로그인 완료된 Docker Desktop 화면

(2) 터미널에서 docker login

> docker login -u ***[사용자의 Docker Desktop 아이디]***
> Password: ***[사용자의 Docker Desktop 비밀번호]***

Login Succeeded 출력 시 성공적인 로그인 완료

(3) git에서 도커 이미지 다운로드

프로그램 다운로드 링크 : https://github.com/deviantony/docker-elk

> git clone
> https://github.com/YooDongseon/IWATCHU/tree/main/Smartwatch_Forensics_Tool/docker

(4) 도커 구동하기

> docker-compose up -d

(5) Kibana 접속

웹 브라우저에서 "**localhost:5601**"로 접속. 접속이 되는지 확인.

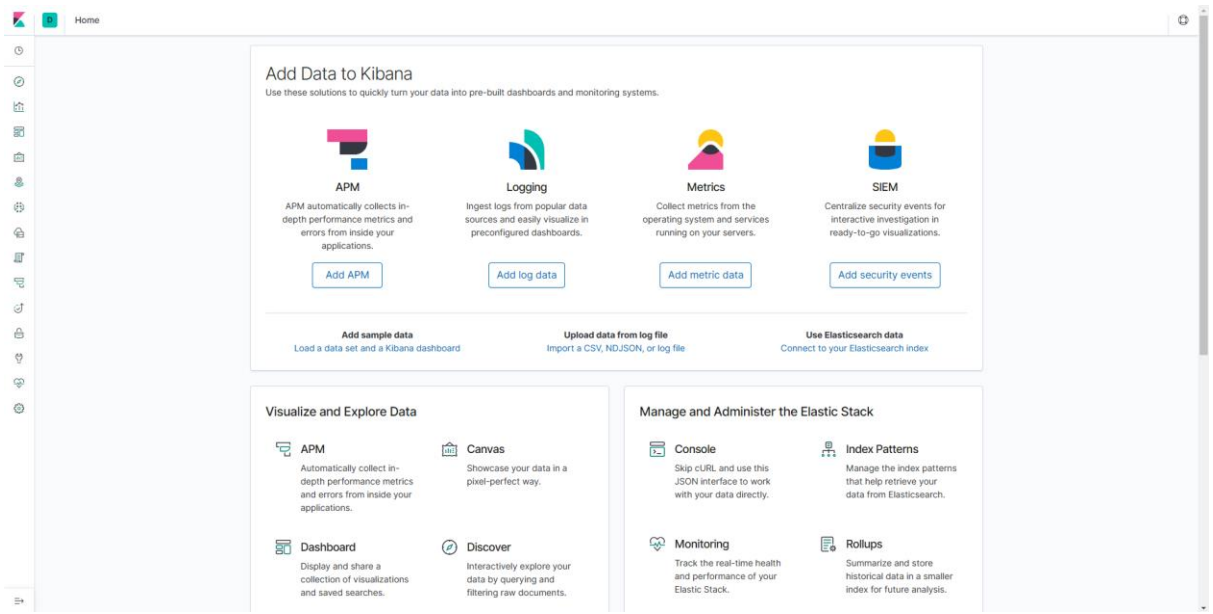


그림 4 _ 도커를 통해 로컬에 구축된 elk 서버 메인 페이지

(+) 도커 구동 멈추기

```
> docker-compose down
```

(6) 추출한 파일 ELK 서버에 올리기

```
> curl -XPUT 'http://localhost:25000' -H 'Content-Type: application/json' -d '@[파일 경로]\[파일이름].json' -A '[서버에서 보고 싶은 파일 이름]
```

파일은 json 형태로 변경해주어야 합니다. 관련하여 업로드 하는 예시는 다음과 같습니다.
만약 안 되는 경우가 있다면 다음의 예시처럼 ‘를 “로 변경하여 업로드 진행.

```
예시 > curl -XPUT "http://121.166.254.165:25000" -H "Content-Type: application/json" -d "@C:\Users\Wynt3r\Documents\BoB_Project\watch5\fin_daily_usage.json" -A "5daily_usagestats"
```

(7) 도구 파일 실행하기

```
> python IWATCHU.py [OS] –[플러그인]
```

플러그인을 -h 나 --help로 작성하시면 다음의 도구에 있는 플러그인 설명을 볼 수 있음.

표 2 _ Tizen OS app 플러그인

플러그인 이름	기능 설명
--csc	기기 csc 버전과 지역코드를 확보합니다.

--swc	소프트웨어가 출시된 지역 코드를 확보합니다.
--vconf	
--gps	GPS 데이터를 통해 이동경로를 jpg 파일로 제작하고, elk-stack에 업로드 합니다.
--reboot	재시작 로그를 확보하고 elk-stack에 업로드합니다.
--time	시간 정보를 확보하고 elk-stack에 업로드합니다.
--battery	배터리 소모 정보에 대한 그래프를 jpg 파일로 제작하고, elk-stack에 업로드합니다.
--res	시스템 자원 정보를 확보하고, elk-stack에 업로드합니다.
--appuse	앱 사용 로그를 확보하고, elk-stack에 업로드합니다.
--appinst	앱 설치 로그를 확보하고, elk-stack에 업로드합니다.
--bootcnt	부팅횟수를 확보합니다.
--all	상단의 모든 플러그인을 실행합니다.

표 3 _ Tizen debug 플러그인

플러그인 이름	기능 설명
--stat	기기 정보를 확보합니다.
--hstat	연결된 호스트 기기의 정보를 확보합니다.
--acc	계정 정보를 확보하고, elk-stack에 업로드합니다.
--comp	연결된 장치 정보를 확보하고 elk-stack에 업로드합니다.
--battery	배터리 소모 정보에 대한 그래프를 jpg 파일로 제작하고, elk-stack에 업로드합니다.
--wnoti	wnoti 서비스 데이터 베이스를 확보하고, elk-stack에 업로드합니다.
--ssacc	삼성 계정 관련 로그를 확보하고 elk-stack에 업로드합니다.
--contact	연락처 데이터 베이스를 확보하고, elk-stack에 업로드합니다.
--msg	메시지 내역을 확보하고, elk-stack에 업로드합니다.
--weather	날씨 정보를 확보하고, elk-stack에 업로드합니다.
--sensor	센서 데이터를 확보하고, elk-stack에 업로드합니다.
--batmon	배터리 모니터의 데이터를 확보하고 elk-stack에 업로드합니다.
--reset	재시작 관련 데이터를 수집합니다.
--calender	사용자 일정 데이터를 확보하고, elk-stack에 업로드합니다.
--notice	알림 데이터를 확보하고, elk-stack에 업로드합니다.
--alarm	알람 데이터를 확보하고, elk-stack에 업로드합니다.
--reminder	상기 알람 데이터를 확보하고, elk-stack에 업로드합니다.
--history	애플리케이션 동작 기록을 수집하고, elk-stack에 업로드합니다.
--all	상기된 모든 플러그인을 실행합니다.

표 4 _ Wear OS 플러그인

플러그인 이름	기능 설명
--apk	기기에 설치된 모든 apk 파일을 추출합니다.
--network	기기와 연결되었던 네트워크 이력을 확보하고, elk-stack에 업로드 합니다.
--package	서드파티 애플리케이션의 최초설치일자, 권한 정보를 확인합니다. elk-stack에 업로드 합니다.

--usage	앱 동작 기록을 확인하고, elk-stack에 업로드합니다.
--access	확보가능한 파일 목록을 생성하고, PC로 추출합니다.
--picture	호스트 스마트폰으로부터 동기화된 사진 파일을 추출하고, exif 정보를 추출하여 지도에 마킹합니다.
--del	삭제된 애플리케이션을 확인하고, 구글 플레이에 자동 검색합니다.
--all	상기된 모든 플러그인을 실행합니다.

만약 올린 데이터가 제대로 보이지 않거나 json 파일 내 존재하는 시간 정보가 올바르지 않다면 다음의 과정을 따라해주시면 됩니다.

(1) docker 설정 파일 중 logstash/pipeline 내 logstash.conf 파일 수정

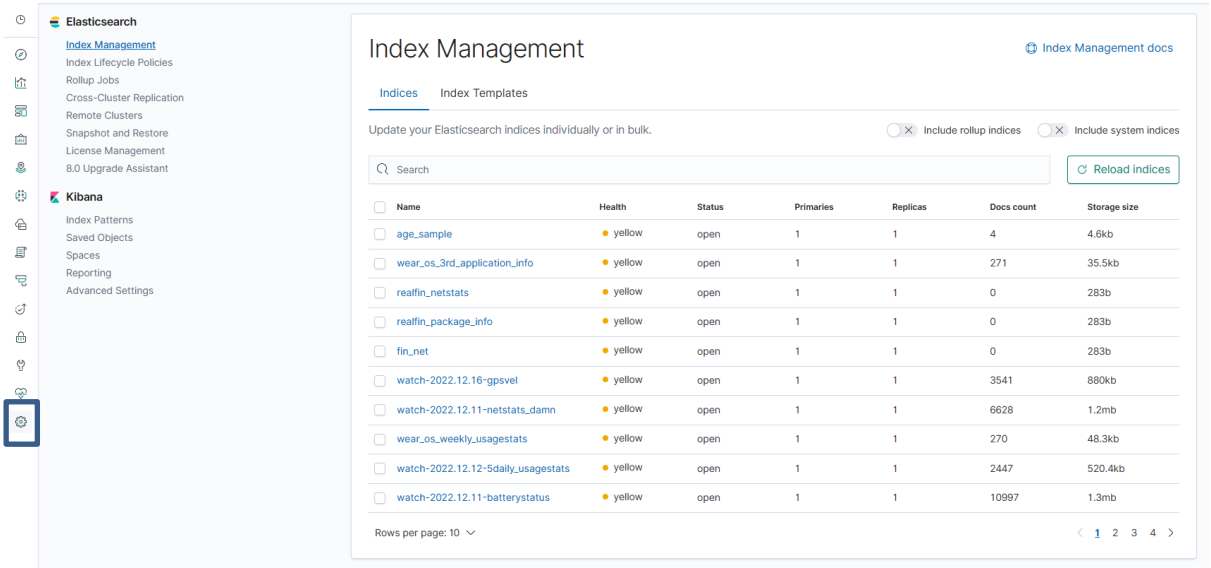
```
input {
    http {
        port => 25000
        codec => "json"
    }
}
filter {
    date {
    match => [ "Bdate", "UNIX_MS" ]
    #첫 번째 Bdate 위치에는 현재 설정되어 있는 날짜의 변수 명을, 두 번째 UNIX_MS
위치에는 현재 변수에 작성되어 있는 형식을 작성해주세요.
    target => "Bdate_new"
    #새롭게 선언될 날짜에 대한 변수명을 지정해줍니다.
    }
    date {
    match => ["Date", "yyyy.MM.dd HH:mm:ss.SSS"]
    target => "Date_new"
    }
}
output {
    elasticsearch {
        hosts => ["elasticsearch-iwu:29200"]
        index => "watch-%{+YYYY.MM.dd}-%{[headers][http_user_agent]}"
    }
}
```

(2) 추출한 파일 ELK 서버에 올리기

> curl -XPUT 'http://localhost:25000' -H 'Content-Type: application/json' -d '@***[파일 경로]\[파일이름]***.json' -A '***[서버에서 보고 싶은 파일 이름]***

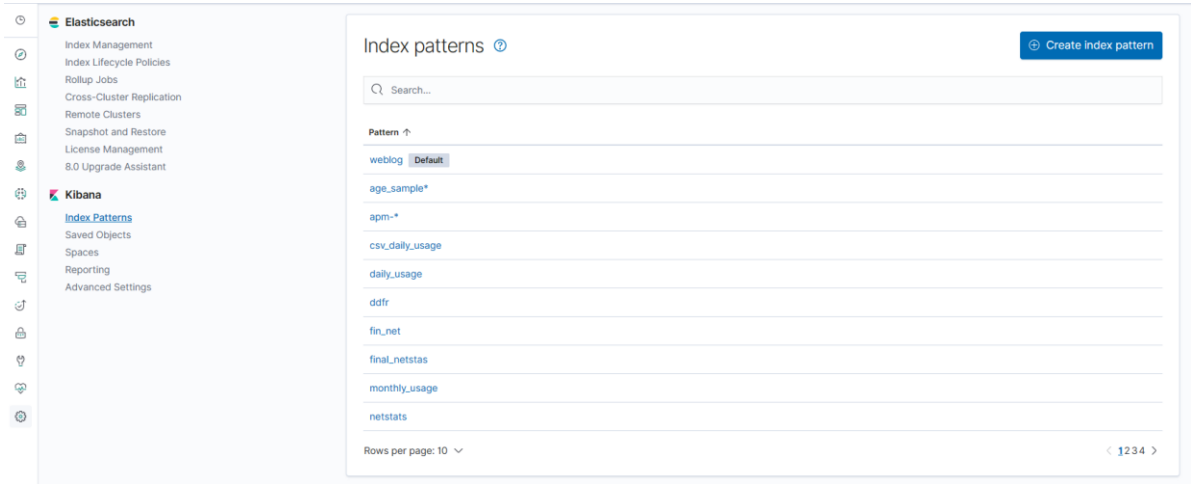
다시 파일을 올려주시기 바랍니다. 만약 이전에 파일을 업로드하였다면 3번의 순서에서 보고 싶은 이름으로 지정된 파일을 지우고 계속해주시길 바랍니다.

(3) localhost:25601 접속 후 설정의 Index Management



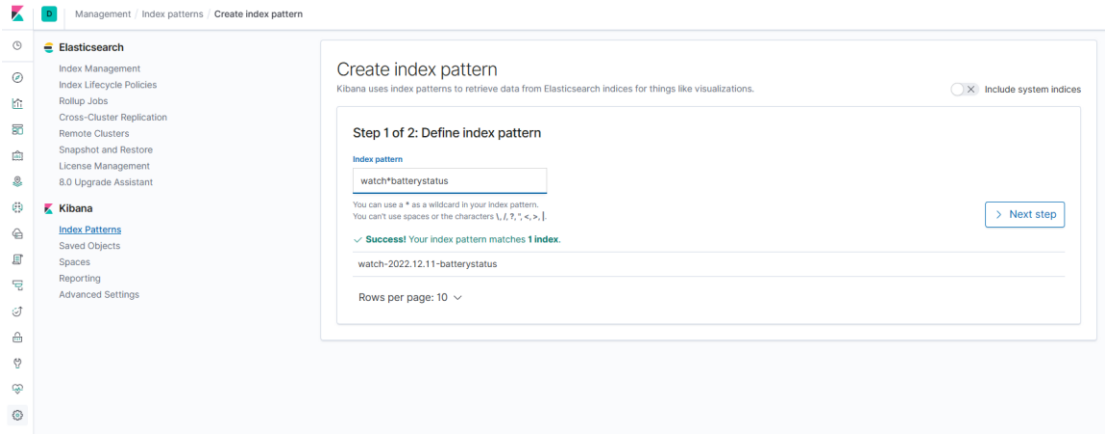
watch-"업로드한 날짜"-"앞서 서버에서 보고 싶은 파일 이름" 으로 잘 업로드 되었는지 확인.

(4) Index patterns 생성



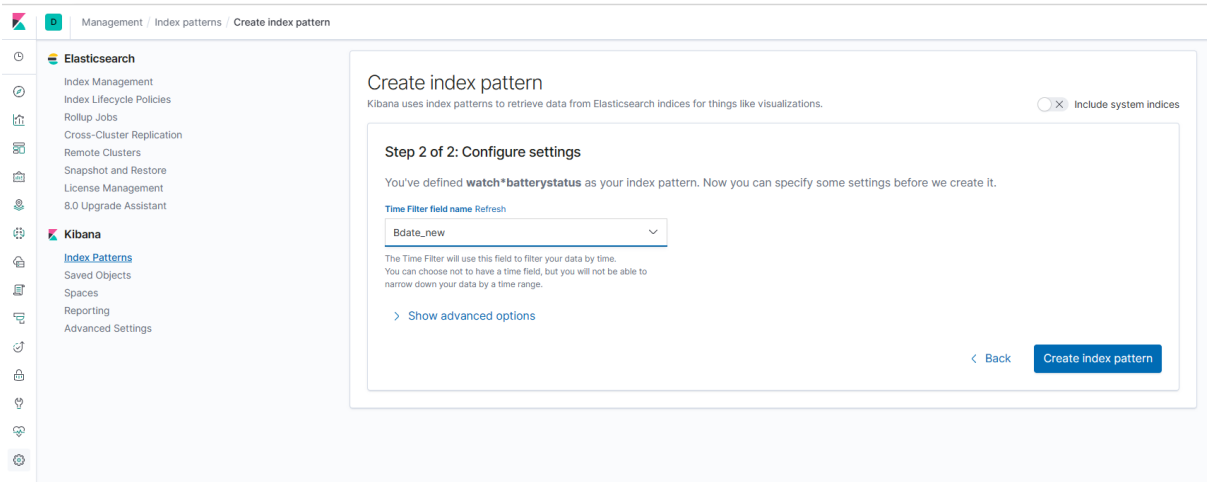
Index patterns로 접속하여 Create Index pattern 버튼 클릭

(5) Index pattern 정의



watch*"앞서 서버에서 보고 싶은 파일 이름" 으로 Success!라는 문구와 함께 아래에 원하는 데이터가 뜬다면 Next step을 클릭

(6) Index pattern의 시간 필드 설정



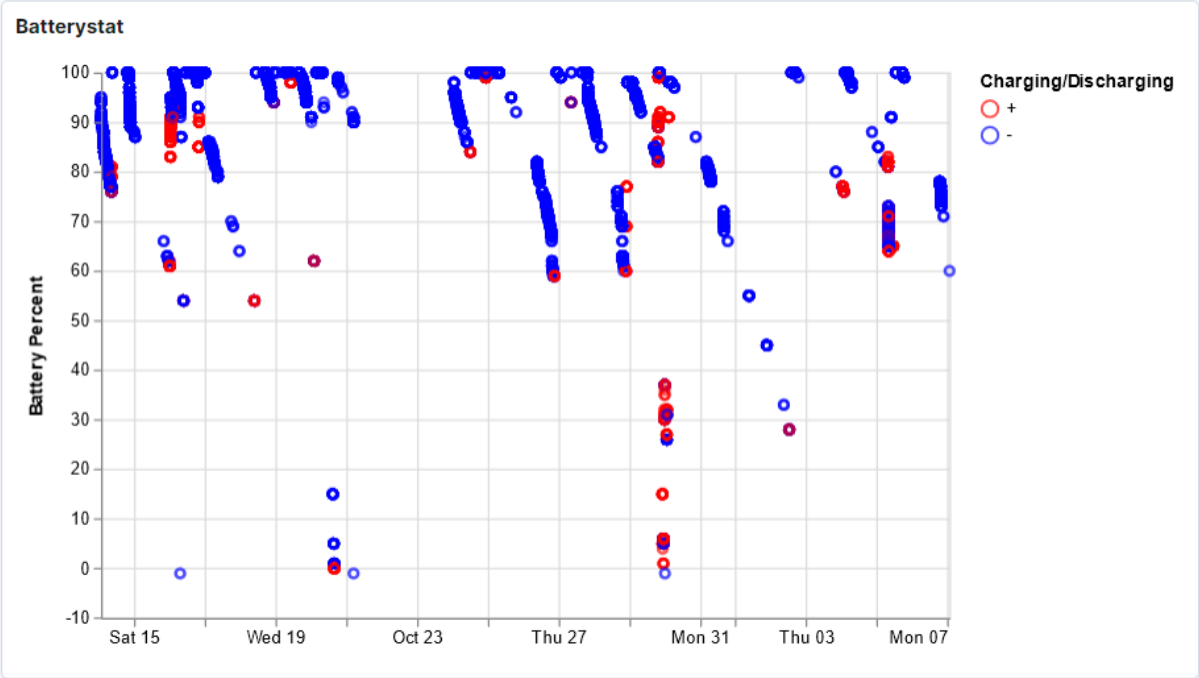
시간 필터로 사용할 field를 다음과 같이 지정해주고 create index pattern 버튼을 클릭.

3. 제공된 템플릿 소개

각 OS 별 추천하는 제공된 템플릿입니다. 데이터구성의 문제로 OS별 호환이 안 될 가능성이 있습니다.

1) Tizen OS

--battery



--battery 플러그인을 통해 얻은 배터리 충전/방전 기록을 시각화할 수 있는 템플릿입니다. 애플리케이션 레벨과 디버그 레벨 모두에서 확보할 수 있는 자료이며, 붉은 표시는 충전 중임을, 푸른 표시는 방전 중임을 나타냅니다.

--gps

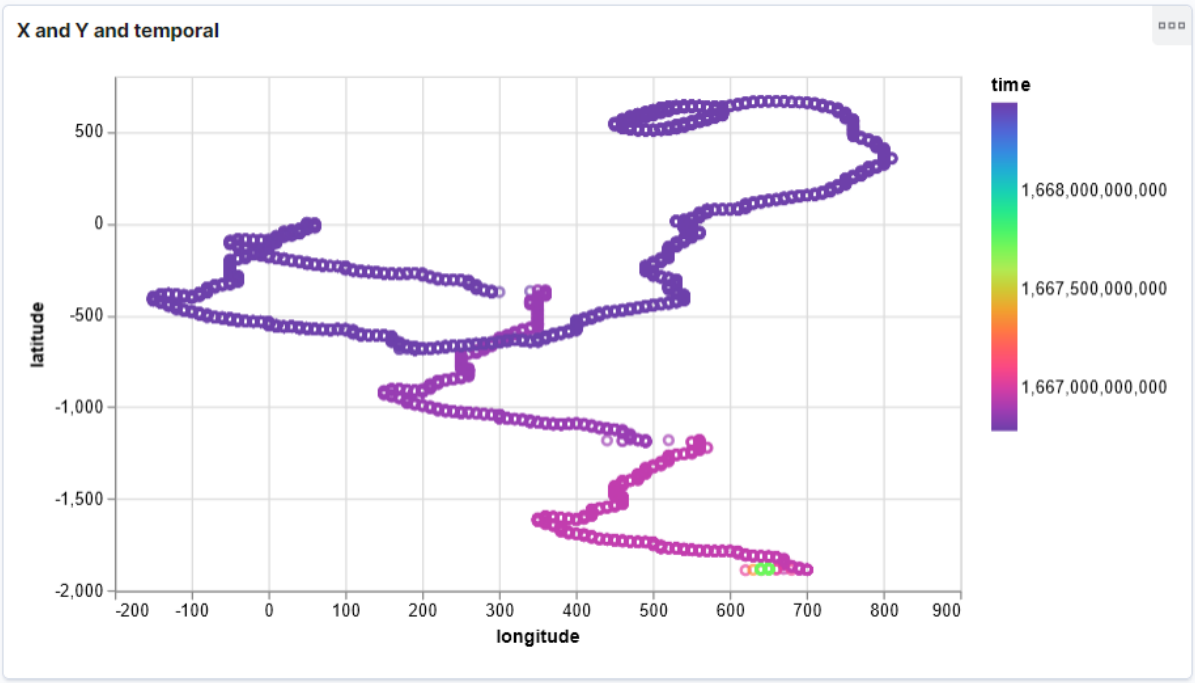


그림 5 –gps 플러그인 시각화(경도-위도-시간)

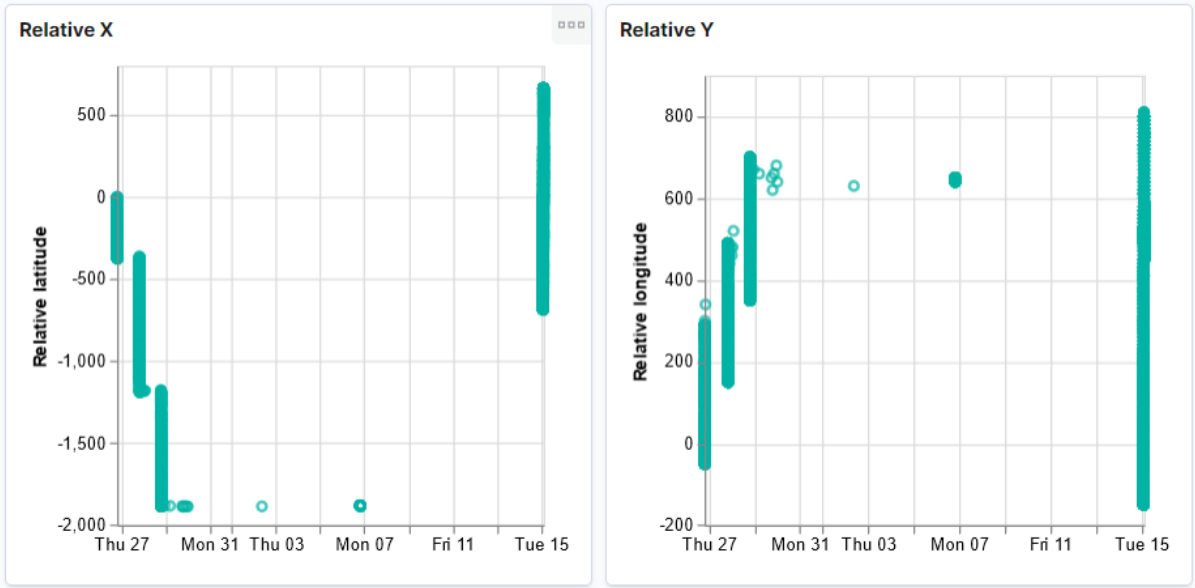


그림 6 –gps 플러그인 시각화(시간-위도, 시간-경도)

--gps 플러그인을 통해 얻은 일부 마스킹된 GPS 자료에서 소수점 아래 식별 가능하고 의미 있는 자릿수를 수학적으로 계산하여 얻은 이동 경로를 표시해줍니다(위도 유효 단위: 약 11m, 경도 유효 단위: 약 1.1m). 또한 무지개 색의 연속된 색상으로 표시하여 어떤 이동이 연속적으로 이루어졌는지 알 수 있습니다(그림 3).
또한 각 위도, 경도를 시계열적으로도 표시하여 연속적 이동을 쉽게 분할할 수 있습니다(그림 4).

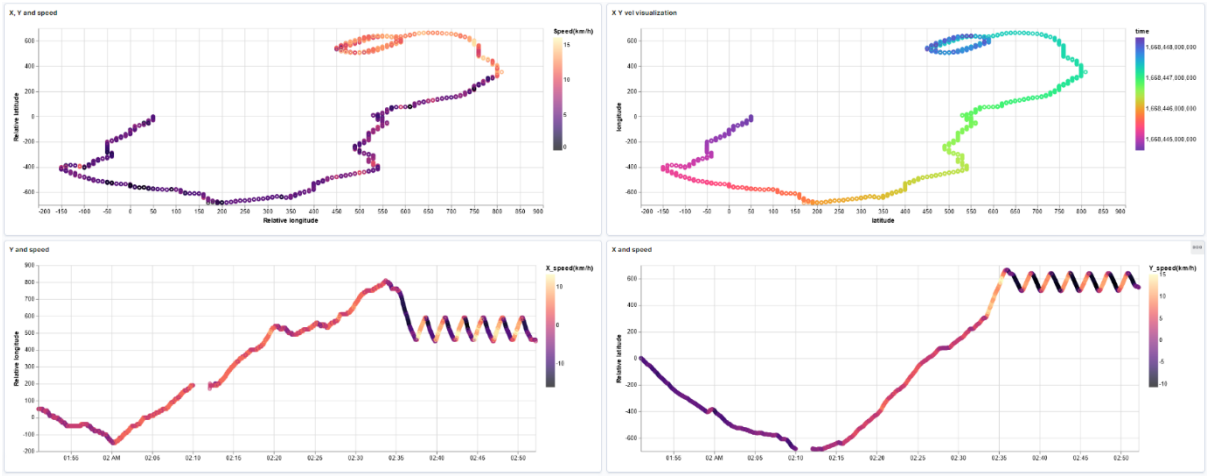


그림 7 –gps 플러그인 시각화(왼쪽 위부터 시계방향으로 경도-위도-속력, 경도-위도-시간, 시간-경도-경도 방향 속력, 시간-위도, 위도 방향 속력)

이렇게 얻어진 개별 연속적 이동에 대하여서 추가적인 속도 분석도 가능합니다. 속도 분석은 10개 측정 기록을 단위로 하여 처음과 끝의 변위와 이동 시간을 나누어 계산됩니다. 왼쪽 위와 아래쪽 사진들에서 색상은 속력을 의미하며, 오른쪽 위 사진에서 색상은 시간을 의미합니다.

아래 사진들에서는 왼쪽 위 사진에서는 겹쳐서 확인할 수 없는, 반복된 이동경로(운동장 트랙)를 확인할 수 있습니다. 총 6바퀴 반 정도를 돈 것을 확인할 수 있습니다.

2) Wear OS

--network

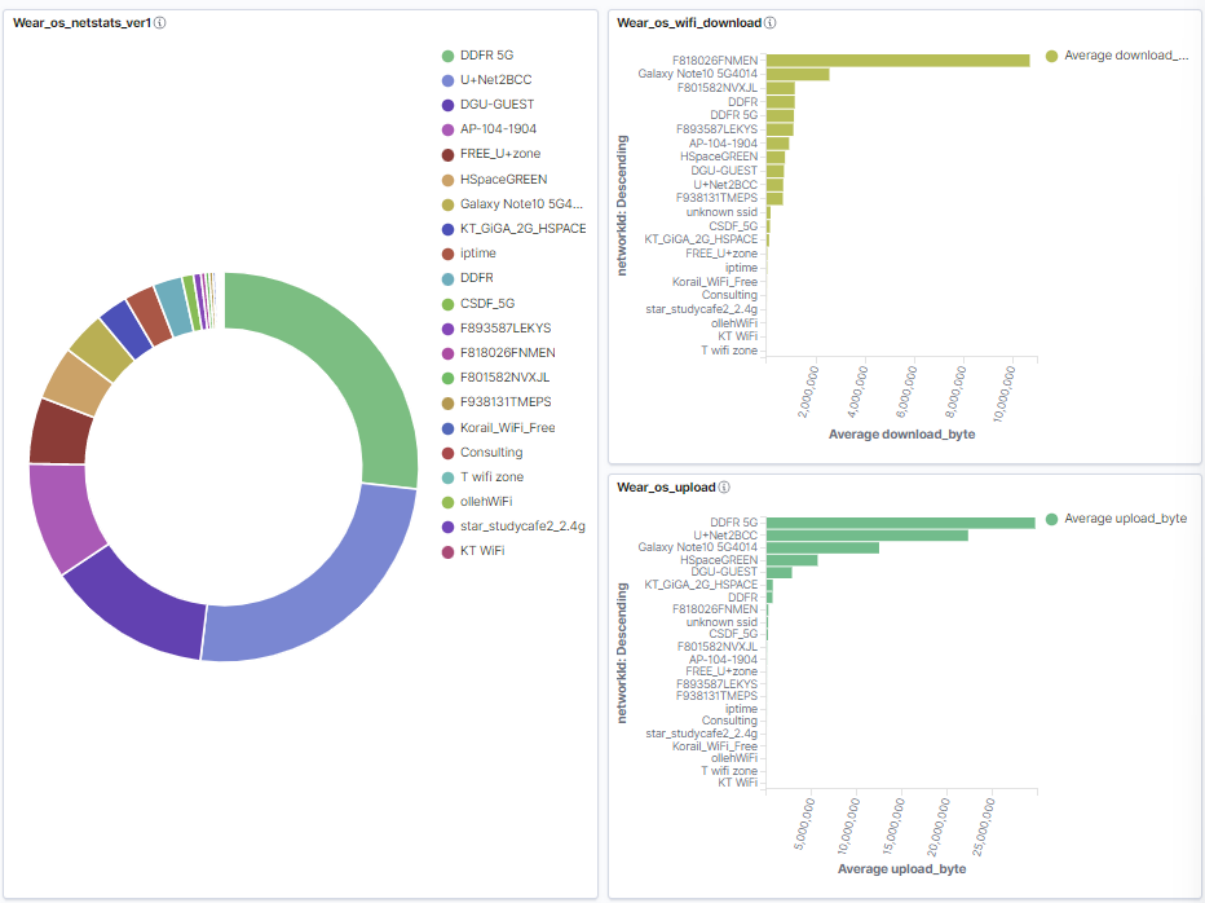


그림 8 _ netstats 템플릿

--network 플러그인을 통해 확보한 네트워크 연결 이력을 활용하는 템플릿입니다. 가장 로그 개수가 많은 순서대로 SSID를 원형그래프로 표현하며, 평균 다운로드/업로드 바이트 순으로 막대그래프로 시각화 합니다.

--package

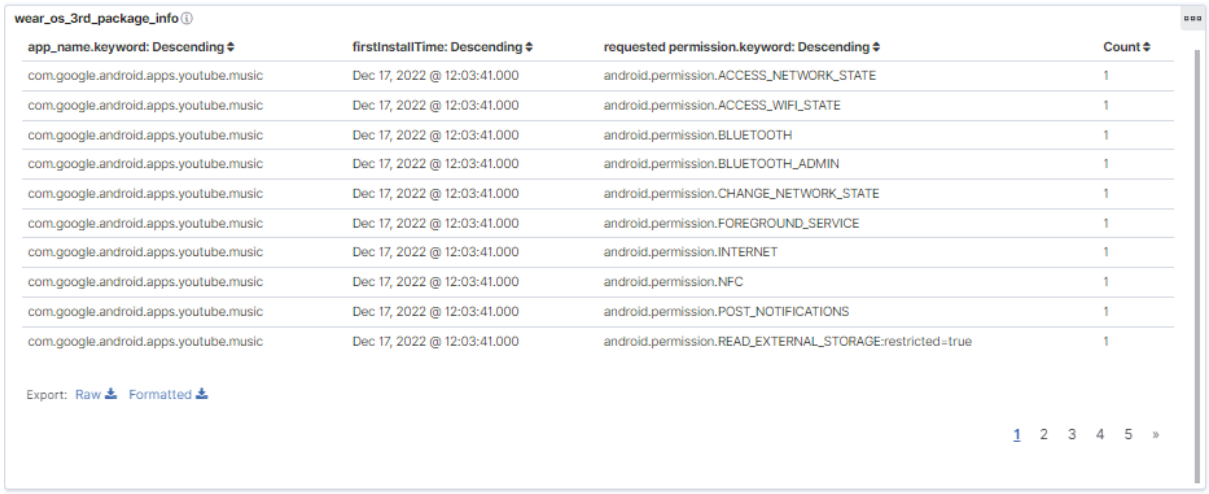
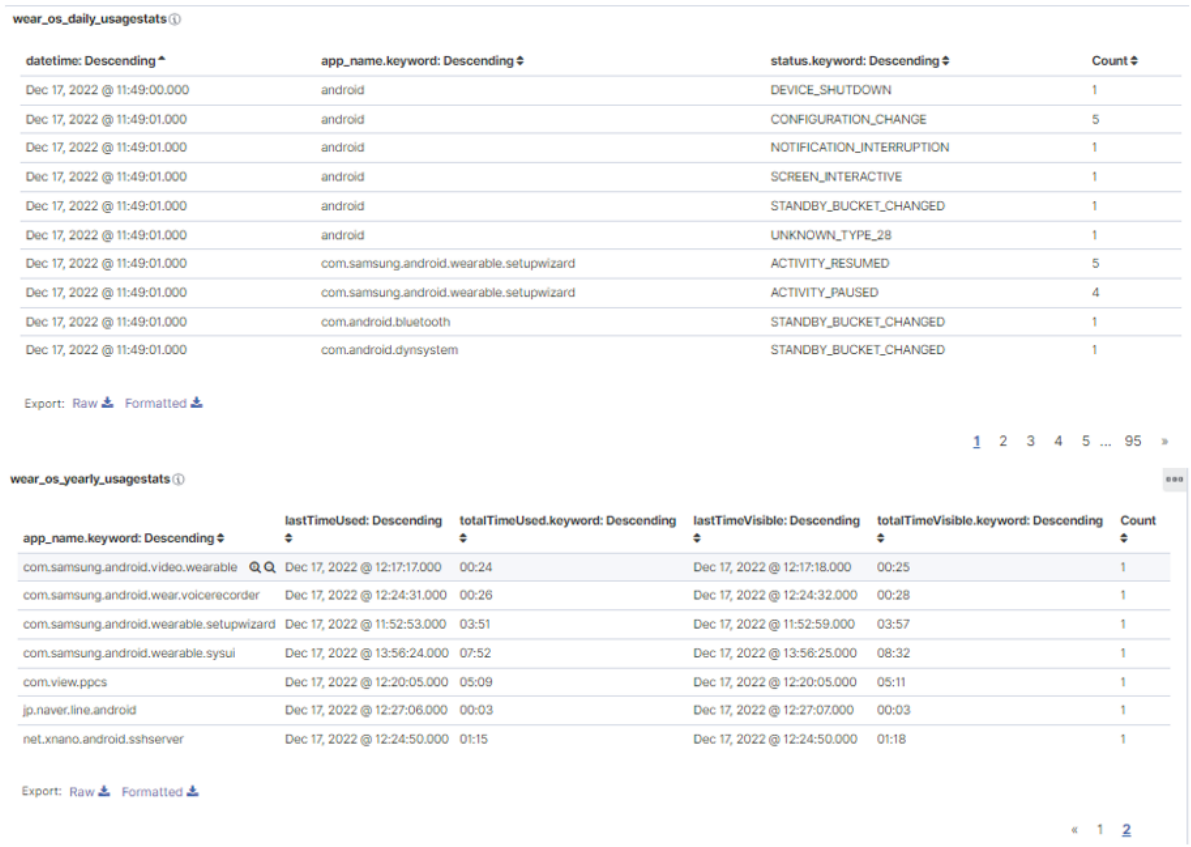


그림 9 _ package_info 템플릿

--package 플러그인을 통해 전송된 서드파티 패키지의 최초설치일자와 요청 권한 정보를 표로 나타냅니다.

--usage



wear_os_daily_usagestats

datetime: Descending	app_name.keyword: Descending	status.keyword: Descending	Count
Dec 17, 2022 @ 11:49:00.000	android	DEVICE_SHUTDOWN	1
Dec 17, 2022 @ 11:49:01.000	android	CONFIGURATION_CHANGE	5
Dec 17, 2022 @ 11:49:01.000	android	NOTIFICATION_INTERRUPTION	1
Dec 17, 2022 @ 11:49:01.000	android	SCREEN_INTERACTIVE	1
Dec 17, 2022 @ 11:49:01.000	android	STANDBY_BUCKET_CHANGED	1
Dec 17, 2022 @ 11:49:01.000	android	UNKNOWN_TYPE_28	1
Dec 17, 2022 @ 11:49:01.000	com.samsung.android.wearable.setupwizard	ACTIVITY_RESUMED	5
Dec 17, 2022 @ 11:49:01.000	com.samsung.android.wearable.setupwizard	ACTIVITY_PAUSED	4
Dec 17, 2022 @ 11:49:01.000	com.android.bluetooth	STANDBY_BUCKET_CHANGED	1
Dec 17, 2022 @ 11:49:01.000	com.android.dynsystem	STANDBY_BUCKET_CHANGED	1

Export: Raw Formatted

1 2 3 4 5 ... 95 »

wear_os_yearly_usagestats

app_name.keyword: Descending	lastTimeUsed: Descending	totalTimeUsed.keyword: Descending	lastTimeVisible: Descending	totalTimeVisible.keyword: Descending	Count
com.samsung.android.video.wearable	Dec 17, 2022 @ 12:17:17.000	00:24	Dec 17, 2022 @ 12:17:18.000	00:25	1
com.samsung.android.wear.voicerecorder	Dec 17, 2022 @ 12:24:31.000	00:26	Dec 17, 2022 @ 12:24:32.000	00:28	1
com.samsung.android.wearable.setupwizard	Dec 17, 2022 @ 11:52:53.000	03:51	Dec 17, 2022 @ 11:52:59.000	03:57	1
com.samsung.android.wearable.sysui	Dec 17, 2022 @ 13:56:24.000	07:52	Dec 17, 2022 @ 13:56:25.000	08:32	1
com.view.ppcs	Dec 17, 2022 @ 12:20:05.000	05:09	Dec 17, 2022 @ 12:20:05.000	05:11	1
jp.naver.line.android	Dec 17, 2022 @ 12:27:06.000	00:03	Dec 17, 2022 @ 12:27:07.000	00:03	1
net.xnano.android.sshserver	Dec 17, 2022 @ 12:24:50.000	01:15	Dec 17, 2022 @ 12:24:50.000	01:18	1

Export: Raw Formatted

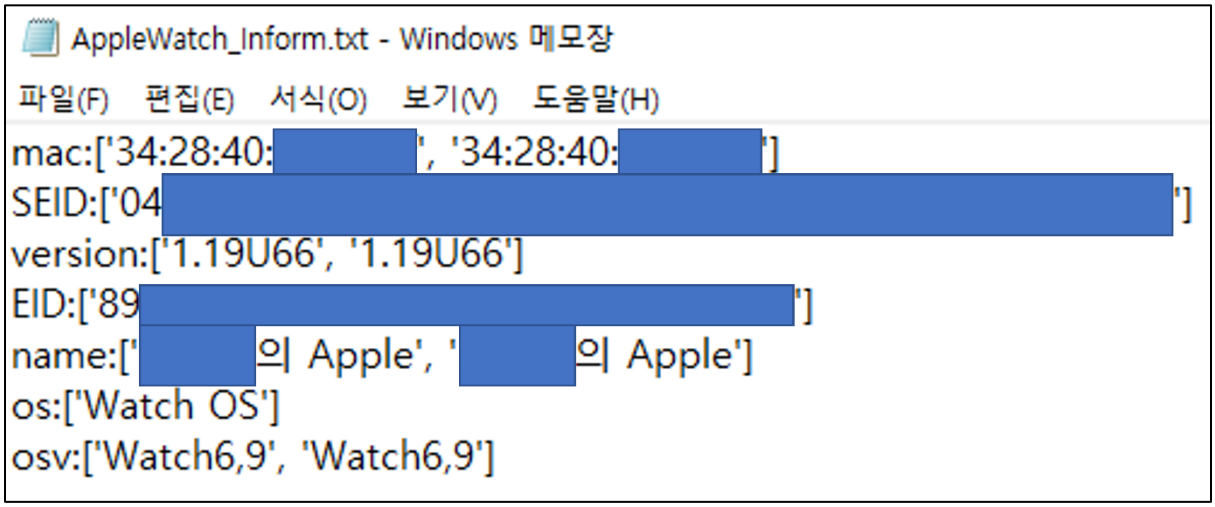
« 1 2

그림 10 _ usagestats 템플릿

--usage 플러그인을 통해 전송된 앱 동작 기록에서 24시간 동작기록과 1년 동작기록을 표로 시각화 합니다. 24시간 동작 기록에서는 하루 동안 발생한 앱의 활성화, 중단, 알림 발생 시각 등의 정보를 알 수 있습니다. 24시간 동작 기록에서는 삭제된 앱의 동작기록이 남지 않는 경우가 존재합니다. 1년 동작 기록에서는 앱이 마지막으로 실행된 일자와 총 사용된 시각을 알 수 있습니다. 삭제된 앱의 기록도 남습니다.

3) Watch OS

Watch OS의 정보를 수집할 경우 연결되었던 스마트워치의 기기 정보 확인이 가능합니다.



위의 이미지와 같이 mac주소, SEID, EID, 장비의 이름등을 확인할 수 있으며, txt파일로 추출됩니다.