

안드로이드에서 eBPF를 이용한 타임스탬프 변경 이벤트 감지

WDSC 2025

Author : 안균승, 김선재, 정연수, 김보겸, 조성제

Presenter : 안균승

Affiliation : 단국대학교

Email : staran1227@dankook.ac.kr

INDEX

01

서론

02

관련 연구

03

날짜 및 시간 변경 탐지 방법

04

실험 및 결과 분석

05

결론



01

서론

서론 – 포렌식 수사

사회

‘전투기 촬영’ 10대 중국인들 “인천·제주·김포공항도 찍었다”...2~3차례 입국 이력

입력 2025.04.10 (16:00)



요약

이들은 경찰 조사에서 “평소 비행기 사진을 찍는 취미가 있다”는 취지로 진술한 것으로 알려졌다.

경찰은 이들을 출국 정지 조치하는 한편, **휴대전화 포렌식** 등을 통해 대공 혐의점 여부 등을 조사하고 있습니다.

만약 사용자가 포렌식 수사 이전에
시간을 **조작**한다면
알리바이가 **조작**될 수 있음

서론 – 안티 포렌식, 타임 스탬프 변경

❖ 안티 포렌식

❖ 디지털포렌식 수사를 방해하기 위해 증거를 은닉, 변조, 파괴하는 기법 전반을 의미함

❖ 타임 스탬프 변경

❖ 파일, 시스템 리소스의 메타데이터를 조작하여 실제 이벤트 발생 시점을 감추는 기법

(예시) 시나리오

타임 스탬프	아티팩트
10:30	사진 촬영
11:00	사진 전송
11:30	갤러리 이미지 삭제
12:00	메시지 삭제

(예시) 조작된 시나리오

타임 스탬프	아티팩트
08:00	갤러리 이미지 삭제
09:30	메시지 삭제
17:30	사진 촬영
18:00	사진 전송

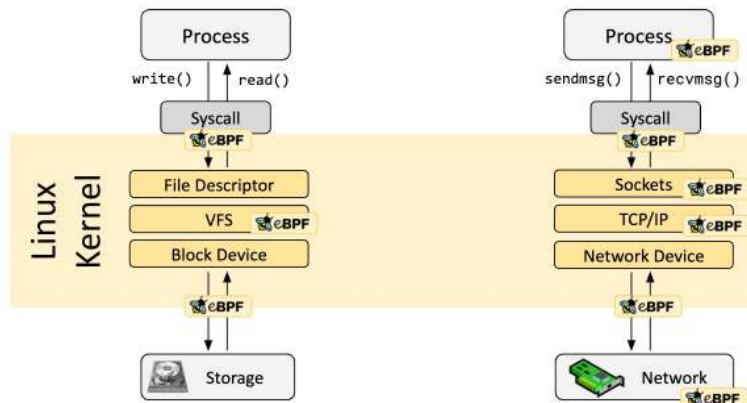
삭제 및 전송한
사진이 없는 것
처럼 보임

서론 – eBPF

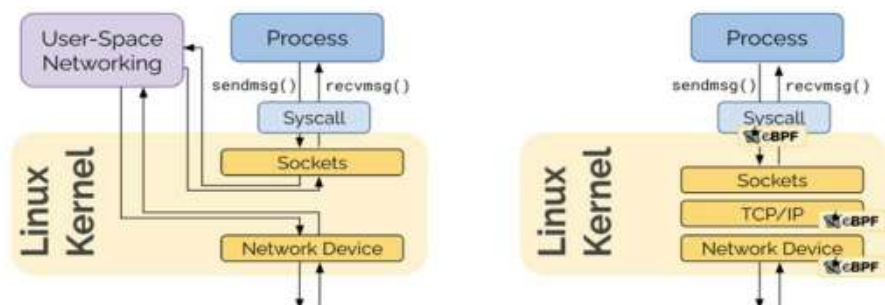
❖ eBPF

- ❖ 리눅스 커널 내에서 사용자 정의 프로그램을 안전하게 작성하여 실행시켜주는 프로그래밍 기술임
- ❖ 커널이나 애플리케이션에서 발생하는 **특정 이벤트 지점**을 **후킹(Hooking)**할 수 있게 됨
- ❖ 따라서, 프로그램이 어떤 사용자 라이브러리나 다양한 라이브러리를 사용하더라도, eBPF는 해당 라이브러리 내부 동작을 몰라도 **최종적으로 호출되는 시스템 콜만 추적하면 분석이 가능함**
- ❖ 또한, 기존 리눅스 명령어와 달리 **커널 내부에서 작동하므로 필터링 시간이 더 적게 듭**

eBPF 후킹 가능 지점



eBPF와 기존 모니터링 명령어와 차이점



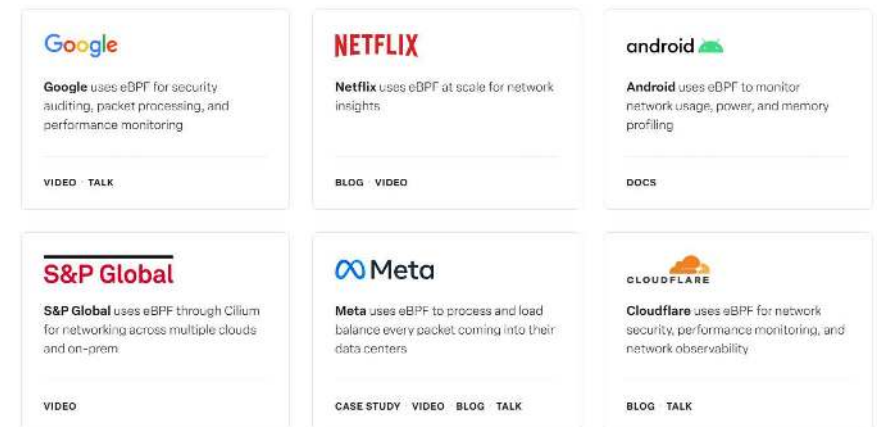
출처: eBPF Documentary

서론 – eBPF 이용 예시

- ❖ Google: 보안 감사, 패킷 처리, 성능 모니터링
- ❖ Netflix: 대규모 네트워크 인사이트 수집
- ❖ Android: 네트워크·전력·메모리 사용 모니터링
- ❖ S&P Global: 멀티 클라우드 네트워킹(Cilium)
- ❖ Meta: 데이터센터 패킷 처리·로드밸런싱
- ❖ Cloudflare: 네트워크 보안·관측성·성능 모니터링

eBPF 사용 예시

Organizations in every industry use eBPF in production



출처:eBPF Documentary

서론 – bpftrace, 목적

❖ Bpftrace

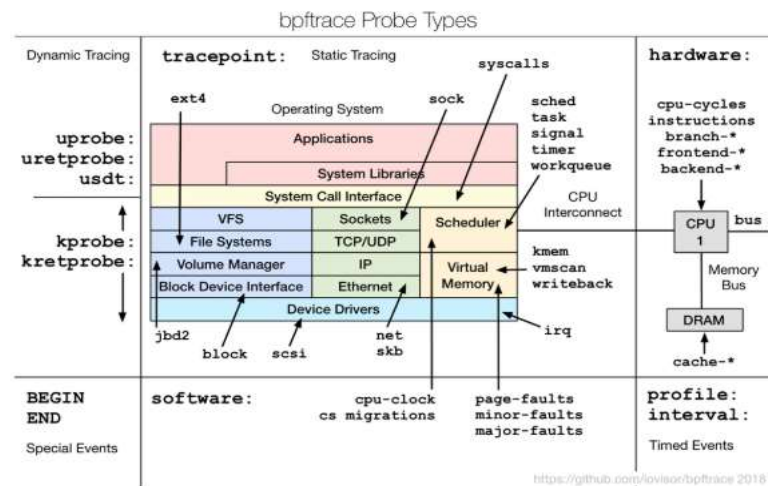
❖ C언어로 작성된 스크립트를 컴파일하여 BPF를 통해 커널 내 추적이 가능해주게 하는 오픈소스 도구

❖ ExtendedAndroidTools

❖ 오픈소스 프로젝트로 bpftrace와 python 등을 안드로이드 버전에 맞게 크로스컴파일 해주는 도구

❖ 이를 이용하여 안드로이드에서 시간 변경 이벤트를 **실시간으로 탐지**하여 디지털 포렌식 분석의 **신뢰성 보장**

bpftrace의 후킹 가능 지점





02

관련 연구

관련 연구

- ❖ “리눅스와 안드로이드 시스템에서 타임스탬프 조작 식별을 위한 로그 분석 기법”(이산 등.)
 - ❖ 리눅스 및 안드로이드 환경에서 **로그 기반 분석**을 통해 타임스탬프 조작을 탐지하는 기법을 제안함
- ❖ “Forensic Detection of Timestamp Manipulation for Digital Forensic Investigation” (Oh et al.)
 - ❖ MS Windows NTFS 파일 시스템 상의 타임스탬프 조작을 탐지하기 위한 **정적 분석 기법**을 제시 함
- ❖ “BPFroid: Robust Real Time Android Malware Detection Framework” (Agman et al.)
 - ❖ **eBPF**를 이용하여 **실시간** 안드로이드 악성앱 **탐지**를 시도함
- ❖ 반면 본 연구에서는 **실시간으로 타임스탬프 변경 이벤트를 감지**하려고 함



03

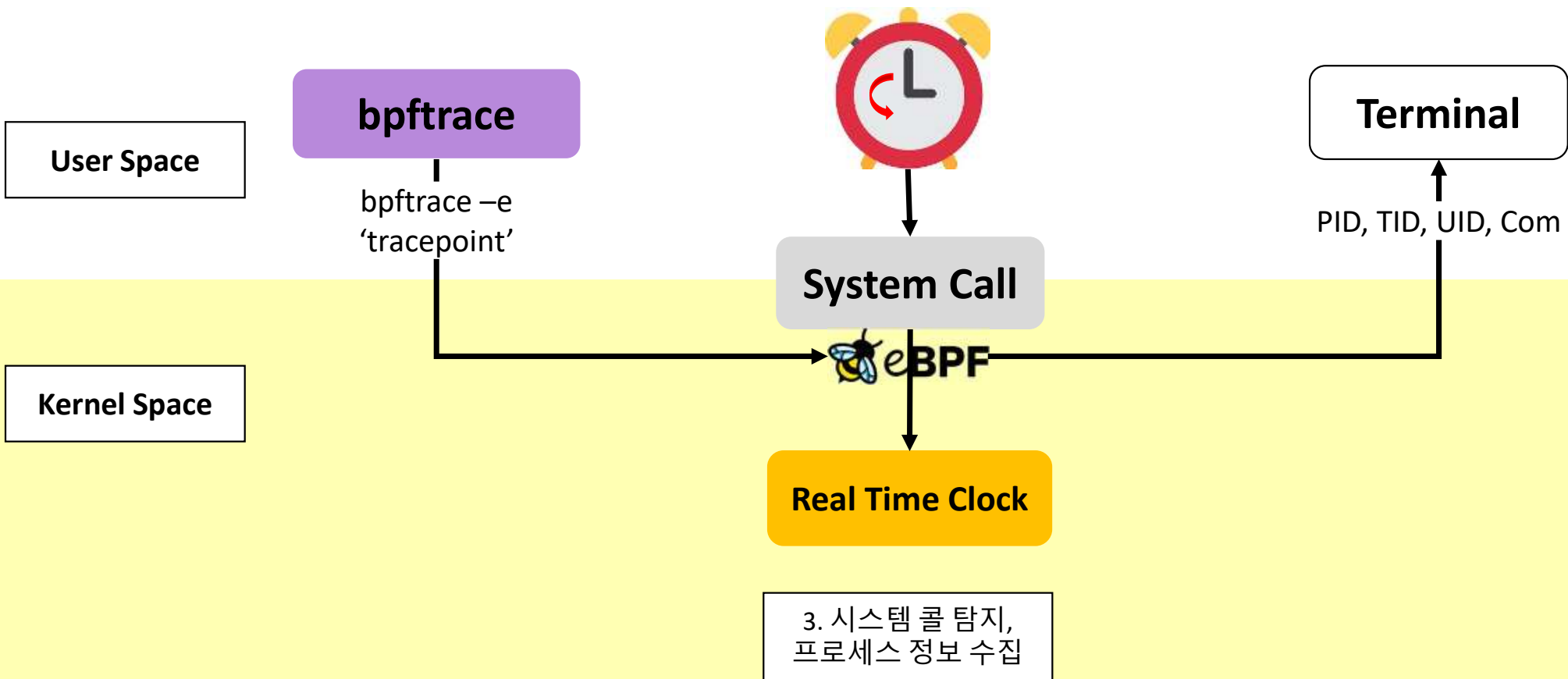
날짜 및 시간 변경 탐지 방법

날짜 및 시간 변경 탐지 방법

1. bpftrace를 이용해 원하는
혹 지점에 스크립트를 작성

2. 사용자가 기기의 시간 조작

4. 터미널창 출력



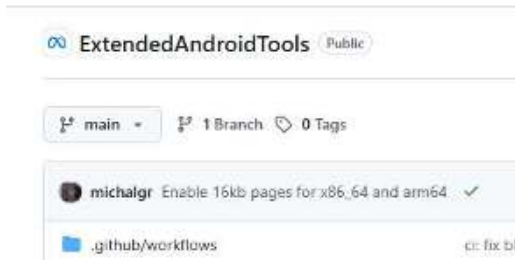
04

실험 및 결과 분석

실험 및 결과 분석 – 실험 환경 설정

- ❖리눅스 OS 환경에서 **ExtendedAndroidTools**를 설치
- ❖깃허브에 적힌 명령어를 통해 **bpfttrace**를 자동 다운로드 및 **크로스 컴파일** 함
- ❖ADB를 통해 안드로이드 기기의 **커널헤더 파일**을 추출해 압축해제 함
- ❖크로스컴파일된 bpftools(bpfttrace)와 압축해제된 커널헤더파일을 ADB를 통해 이동 시킴

ExtendedAndroidTools 깃허브 저장소



ExtendedAndroid Tools 사용 명령어

```
# Build the Docker image
./scripts/build-docker-image.sh

# Run the environment
./scripts/run-docker-build-env.sh

# Build a target of your choice from within the container
> make python
> make bpftools
```

A23N 커널 헤더 파일

```
a23:/sys/kernel # ls
alta_bigdata  config  iommu_groups
boot_adsp     dload  ion
boot_cdsp     fscaps  irq
cgroup        gpu     kheaders.tar.xz
```

사용된 기기 및 도구

안드로이드 폰	eBPF 도구	디버깅 도구
Galaxy A23N Android 14	ExtendedAndroidTools, bpfttrace	Android Debug Bridge 1.0.41

실험 및 결과 분석 - bpftrace 스크립트 실행

❖ ./bpftrace -i 명령어를 통하여 현재 사용가능한 훅(hook) 지점을 확인

❖ 아래 명령어를 통해 bpftrace 실행

```
./bpftrace -e `uprobe:/system/lib64/libc.so:settimeofday{  
    printf("PID: %d\n", pid); printf("COM: %s\n", comm);  
    printf("TID: %d\n", tid); printf("UID: %d\n", uid);  
    system("cat /proc/%d/cmdline", pid);  
}
```

후킹에 사용된 시스템 콜

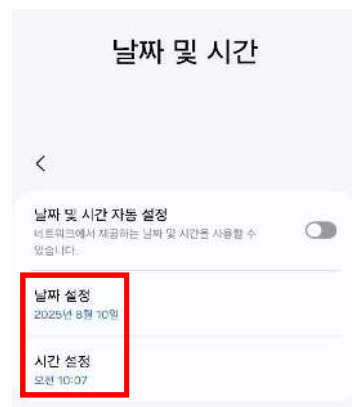
시스템 콜	설명
settimeofday()	시스템의 날짜 및 시간을 변경하는 시스템 콜

실험 및 결과 분석 - 실험 환경 및 사용 명령어

시간 변경에 사용된 명령어

명령어	사용 방법	설명
시스템 설정 앱에서 시간 변경	일반 -> 날짜 및 시간 -> 날짜 설정	시스템 시간을 사용자 지정 날짜 및 시간으로 설정
\$ date MMDDhhmmYYYY.ss	adb shell 접속 후 명령어 실행	시스템 시간을 사용자 지정 날짜 및 시간으로 설정
\$ toolbox date MMDDhhmmYYYY.ss	adb shell 접속 후 명령어 실행	시스템 시간을 사용자 지정 날짜 및 시간으로 설정
\$ hwclock -s	adb shell 접속 후 명령어 실행	하드웨어 클록을 시스템 시간으로 설정

시스템 설정 앱에서 시간 변경



date 명령어를 통한 시간 변경

```
a23:/ # date
Fri Aug 15 13:01:17 KST 2025
a23:/ # date 081022052025
Sun Aug 10 22:05:00 KST 2025
```

toolbox date 명령어를 통한 시간 변경

```
a23:/ # toolbox date
Fri Aug 15 22:06:04 KST 2025
a23:/ # toolbox date 081022052025
Sun Aug 10 22:05:00 KST 2025
```

hwclock -s 명령어를 통한 시간 변경

```
a23:/ # date
Fri Aug 15 22:08:01 KST 2025
a23:/ # hwclock
1970-11-21 11:33:31+0000
a23:/ # hwclock -s
a23:/ # date
Sat Nov 21 20:33:35 KST 1970
```


실험 및 결과 분석 - 실험 결과 및 분석

시간 변경에 사용된 앱 또는 명령 실행 결과

	시스템 설정 앱	date	toybox date	hwclock
PID	1318	13065	17150	13061
Com	binder:1318_7	date	toybox	hwclock
TID	2429	13065	17150	13061
UID	1000	0	0	0
Cmdline	system_server	X	X	X

시간 변경 이벤트 탐지 분석 결과

시간 변경 방법	탐지	실행 명령어
시스템 앱 설정	가능	binder:1318_7
date	가능	date
toybox date	가능	toybox
hwclock -s	가능	hwclock



05

결론

결론 – 목적, 기여

❖연구 목적

- ❖안드로이드 환경에서 발생하는 시간 조작(타임스탬프 변경) 행위를 **실시간**으로 탐지할 수 있는 기법 제안
- ❖로그 기반 정적 분석의 한계를 보완하고, **커널 수준에서의 직접 감시**를 통해 위·변조에 강한 탐지 체계 구현

❖연구 기여

- ❖eBPF 기반의 실시간 탐지 프레임워크를 구현해 `settimeofday()` 시스템 콜을 후킹함으로써 시간 변경 시도를 식별
- ❖시스템 앱, 사용자 명령어, 저수준 도구까지 범용적으로 탐지할 수 있도록 설계
- ❖ExtendedAndroidTools와 bpftrace를 활용해 실제 안드로이드 기기에서 구현·실험

결론 – 성과, 향후 연구

❖연구 성과

- ❖ 모든 시간 변경 방법(date, toybox date, hwclock, 시스템 설정 앱)에서 **실시간 탐지 성공**
- ❖ 실행 주체(PID, TID, UID, 프로세스명, cmdline)까지 확보하여 **분석 신뢰성 향상**
- ❖ 기존 로그/메타데이터 기반 방식 대비 **위·변조 대응 능력과 실시간성 확보**

❖향후 연구

- ❖ 안드로이드 OS의 다양한 기기에서 호환성 검증 및 최적화
- ❖ bpftrace 스크립트 경량화 및 실행 속도 향상
- ❖ 시간 변경 외 다른 안티포렌식 행위(로그 삭제, 메타데이터 변조 등) 확장 대응
- ❖ 탐지 이후 자동 대응 절차 설계(알림, 로그 백업 등)

Q&A
