이웃(IoT) : IoT 기기 디지털 포렌식 도구

팀 이웃(IoT): 김시환(컴퓨터공학 2019102158), 정태규(컴퓨터공학 2020105661)

요 약

디지털 포렌식은 현대 사회에서 중요한 역할을 수행하고 있는 분야 중 하나로, 디지털 기기 및 시스템에서 증거를 수집하고 분석하여 범죄나 인위적인 활동을 조사하는 핵심 기술이다. 하지만 제조사에 따라 다른 파일시스템으로 인해 많은 어려움이 있다. 본 논문에서는 다양한 파일시스템 내에서 동작이 가능한 효율적인 방식의 IoT 기기 디지털 포렌식 도구를 제시한다. 제안하는 방법은 IoT 기기에서 추출한 로그 데이터로 학습시킨 BERT 를 활용하는 방법으로, 딥러닝 알고리즘 기반의 포렌식 도구를 개발하여 기존 포렌식의 복잡성을 해결하고자 한다

1. 서론

1.1. 연구배경

과거에는 컴퓨터와 스마트폰과 같은 전통적인 디지털 기기가 디지털 포렌식의 주요 대상이었다. 그러나 현재의 사회에서는 IoT 기기가 우리 주변에 널리 보급되어 있다. 스마트 홈 시스템, 스마트 카메라, 스마트 냉장고 및 의료 기기와 같은 IoT 기기는 우리의 일상 생활에 녹아들어 있으며, 기업과 정부 기관에서도 널리 사용되고 있다. 이러한 IoT 기기는 다양한 데이터를 생성하고 저장하며, 이 데이터는 사건 및 사고의 중요한 증거로 사용될 수 있다.

IoT 기기는 기존의 디지털 포렌식 환경과 다른 독특한특성을 가지고 있어 기존 도구 및 기술로는 충분히 대응하기 어려운 도전 과제를 제기한다. 그 중 하나는 다양한 파일 시스템의 사용이다. IoT 기기는 다양한 플랫폼과 운영 체제로 구성되어 있으며, 일반적인 컴퓨터나 스마트폰과는 다른 특성을 가지고 있다. 또한 메모리가 제한적이기 때문에 디지털 포렌식 도구의 성능 및 용량 제약이 중요한 고려 사항이다.

IoT 기기에 대한 적절한 디지털 포렌식 도구의 부재는 범죄 조사, 기업 보안 사고 대응, 사회 문제 해결 및 개인 정보 보호에 대한 심각한 리스크를 초래할 수 있다. 따라서 IoT 기기의 디지털 포렌식을 위한 효율적이고 특화된 도구가 필요하다.

따라서 이 프로젝트는 IoT 기기에 대한 디지털 포렌식을 수행하기 위한 도구를 개발하는 데에 중점을 두고 있으며, 이러한 도구는 현재의 디지털 포렌식 커뮤니티와 법 집행 기관에 큰 가치를 제공할 것으로 기대된다. 또한, 이 프로젝트는 보안 및 개인 정보 보호에 대한 새로운 관점을 제시하여 사회적으로 의미 있는 결과물을 얻을 것이다.

1.2. 연구목표

본 연구의 주요 목표는 다양한 파일 시스템을 가진 IoT 기기에서 동작하는 디지털 포렌식 도구를 개발하는 것이다. IoT 기기는 OpenWrt, BusyBox 등과 같은 다양한 플랫폼을 사용하여 동작하는데, 이러한 기기들에 대한 디지털 포렌식 전문가 및 수사관이 증거 수집 및 분석을 보다 효과적으로 수행할 수 있는 도구를 개발하고자 한다.

IoT 기기는 다양한 로그 데이터를 생성하며, 이 데이터는 사건 조사 및 사고 분석에 중요한 역할을 한다. 따라서 이 도구는 파일 시스템, 로그 파일, 데이터베이스 등 다양한 데이터 소스에서 데이터를 추출하고 분석하는데 사용될 것이다.

2. 관련개념

2.1. 디지털 포렌식

정보기기에 내장된 디지털 자료를 근거로 삼아 그 정보기기를 매개체로 하여 발생한 어떤 행위의 사실 관계를 규명하고 증명하는 보안서비스 분야로서, 포렌식 방식에 따라 크게 라이브 포렌식(Live Forensic)과 오프라인 포렌식(Offline Forensic)으로 나뉜다.

2.1.1. 라이브 포렌식 (Live Forensic)

라이브 포렌식은 디지털 포렌식 분야에서 사용되는 기술 중 하나로, 실시간 환경에서 컴퓨터 시스템, 네트워크, 또는 디지털 장치에서 즉시 수집된 데이터를 분석하는 프로세스를 의미한다. 하드디스크와 같은 비휘발성 매체뿐만 아니라 컴퓨터 메모리와 같은 휘발성 저장매체로부터 데이터를 얻는 것으로 이루어진다. 라이브 시스템상에서의 디지털 포렌식을 수행하기 위해서는 운영체제가 사용중인 휘발성 메모리와 비휘발성 메모리를 접근할 필요가 있다. 라이브 포렌식은 현장에서 사건 조사나 사이버 공격 대응과 같은 긴급한 상황에서 사용된다.

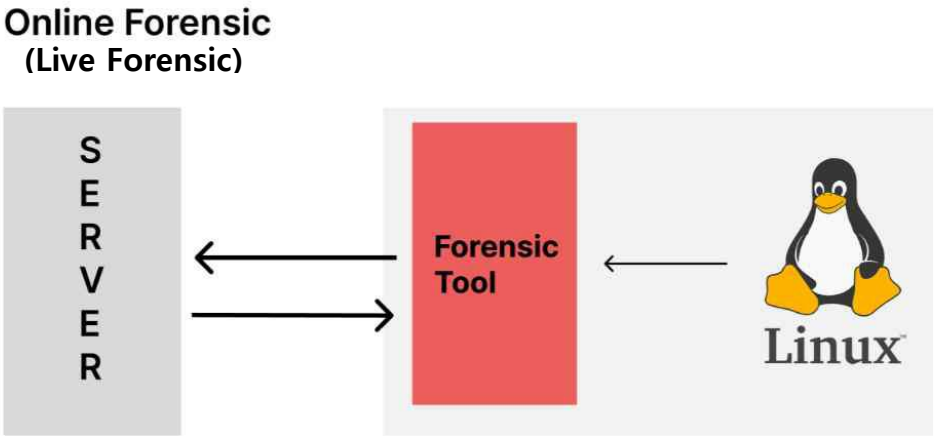

그림 1. 라이브 포렌식 동작 방식

2.1.2. 오프라인 포렌식(Offline Forensic)

오프라인 포렌식(Offline Forensics)은 IoT 기기에서 칩오프(chip-off)를 수행하여 칩에서 데이터를 수집하고 분석하는 프로세스이다.

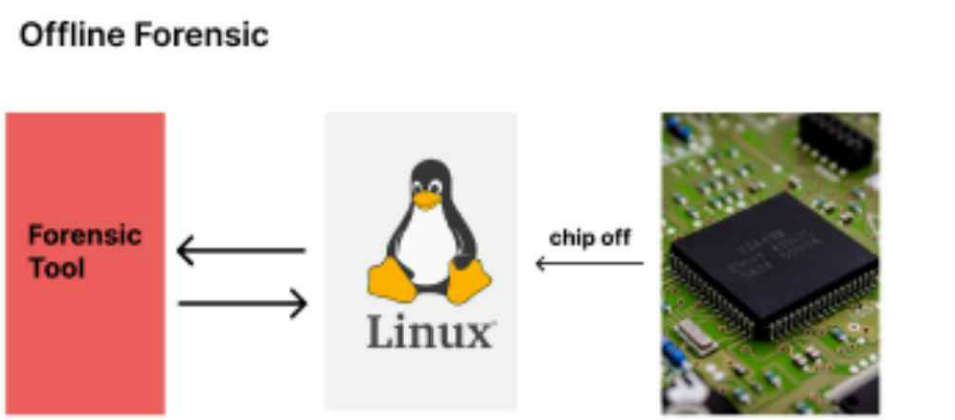

그림 2. 오프라인 포렌식 동작 방식

2.2. 파일 시스템

2.2.1. ext4 파일 시스템

리눅스 운영 체제의 파일 시스템 중 하나로, Android 운영 체제의 기본 파일 시스템 중 하나이다. 사용되는 IoT 기기로는 스마트 TV 와 스마트 워치 등이 있다.

2.2.2. 플래시 파일 시스템(Flash File System)

플래시 메모리(Flash Memory)를 기반으로 하는 데이터 저장 및 관리 시스템이다. 예시로는 JFFS2(Journaling Flash File System2), UBIFS(UBI File System) 등이 있다. IoT 기기로는 OpenWrt 를 사용하는 공유기 등에서 주로 많이 사용된다.

2.2.3. 읽기 전용 파일 시스템(Read-Only File System)

데이터를 읽을 수만 있고 수정할 수 없는 파일 시스템을 말한다. 읽기 전용 파일 시스템은 데이터의 무결성을 유지하고 변경을 방지한다. 이는 악의적인 변경이나 시스템 오류로부터 데이터를 보호하는 데 도움이 된다. 예시로는 Squashfs, Cromfs 등이 있다.

3. 프로젝트 내용

3.1. 시나리오

3.1.1. Live Forensic

Live Forensic 은 실시간으로 IoT 기기에서 디지털 증거를 수집, 분석하고 해석하는 과정을 말한다. 우리가 개발할 Live Forensic 도구의 시나리오는 다음과 같다.

1. 포렌식 대상 디바이스에 우리가 개발한 포렌식 소프트웨어를 설치한다.
2. 소프트웨어는 선택한 디바이스 또는 시스템에서 데이터를 실시간으로 수집한다.
3. 소프트웨어는 수집한 데이터를 분석하여 악성 활동, 해킹 시도, 비정상적인 동작 등을 탐지한다.
4. 탐지된 활동에 대한 증거를 확보하여 필요한 경우 보호 조치를 취한다.

3.1.2. Offline Forensic

Offline Forensic 은 IoT 기기나 기기내의 저장매체를 가져와 디지털 증거를 수집, 분석하고 해석하는 과정을 말한다. 우리가 개발할 Offline Forensic 도구의 시나리오는 다음과 같다.

1. 칩오프(chip-off)와 같은 방식을 통해 포렌식 대상 디바이스 또는 저장매체에 접근한다.
2. 대상 디바이스 또는 저장매체의 파일 구조를 확인한다.
3. 저장되어 있는 데이터를 수집한다.
4. 소프트웨어는 수집한 데이터를 분석하여 악성 활동, 해킹 시도, 비정상적인 동작 등을 탐지한다.
5. 탐지된 활동에 대한 증거를 확보하여 필요한 경우 보호 조치를 취한다.

3.2. 고려사항

3.2.1. 디렉토리별 분석 고려사항

/ (최상위 경로)	* 최상위 디렉토리
/home	* 사용자 홈 디렉토리 - 계정별로 관련된 모든 정보를 저장
/dev	* 장치 관련 디렉토리 - 일반적으로 실행 파일이 존재하지 않는 경로이므로, 실행 파일이 있을 시, 주의 깊게 확인이 필요
/var	* 대부분의 로그가 저장되는 경로

/etc	* 전반적인 시스템 설정 정보를 저장 - 악성 설정 정보에 대한 추적에 용이
/usr	* Default 로 지정되는 Application 설치 경로
/bin	* 명령어가 저장되는 경로 - 보통 실행 파일을 덮어쓰는 등의 방식으로 악성 행위 수행
/proc	* Process 정보를 저장하는 경로

3.2.2. 로그 파일별 분석 고려사항

리눅스 시스템 로그 파일

- 리눅스의 로그들은 주로 /var/log 디렉토리에 저장하게 된다.
- /tmp 디렉토리에 저장되는 경우도 있다.

/var/log/massages

- 시스템에 문제가 생겼을때 가장 먼저 확인해보는 로그 파일.
- syslog.conf 설정파일에 로그를 남기지 않도록 지정된 내용을 제외한 모든 항목들이 기록
- 많은 항목들이 기록되기 때문에 일반적으로 grep 명령어를 통해서 확인

/var/log/secure

- 시스템에 접속한 사용자에 대한 기록이 남는 파일.
- .ssh 접속이나 텔넷 접속이 발생했을 경우 접속한 시간, 사용자명, 클라이언트 IP 등이 기록
- 시스템 해킹이 의심될 경우 이 파일을 열어서 접속 기록을 확인

/var/log/maillog

- 전송되거나 수신된 메일에 대한 내용들이 기록으로 남는다.

/var/log/cron

- crontab 으로 설정한 작업들이 정상 수행되었는지 확인할 수 있는 로그

/var/log/dmesg

- 시스템 부팅시 로그가 기록된다 명령어를 . dmesg 통해 출력되는 내용이 이 파일에 기록된다.

/var/log/wtmp

- 최근 접속 사항이 기록되는 파일이다 명령을 . last 통해 출력되는 내용이 기록된다.

/var/log/lastlog

- 각 사용자의 마지막 로그인 내용이 기록되어 있다 명령어를 . lastlog 통해 관련 내용 확인 가능하다

xferlog

- FTP 데몬으로 파일을 전송한 내용이 기록된다.

/var/log/httpd/access_log

- Apache 서비스 데몬의 로그 파일이다 웹. 서비스를 운영할 경우 해당 내용이 기록된다.

/var/log/httpd/error_log

- Apache 서비스 데몬의 에러와 관련된 내용이 기록된다

3.2.3. 휘발성 데이터 수집 고려사항

분류	명령어/경로 및 내용
호스트 식별	* hostname - 침해 사고 조사 시(특히 다수 대상), 대상 식별을 위해 사용
시간 정보	* date - 현재 시스템에 대한 시간 정보 확인 * date + %s - Unix Time으로 변경하여 시간 정보 출력
시스템 Timezone	* cat /etc/timezone - 시스템 Timezone 설정 상태 확인 (date 명령어로도 확인 가능)
부팅 기준 시간	* uptime - 시스템 부팅 후, 작동한 시간 확인 - 현재 로그인한 사용자 수 - 현재 시간 - 특정 시간동안의 시스템 평균 메트릭 등

IP 설정 정보	* ip ad 또는 ifconfig - 호스트 식별을 위해 사용하며, eth, ens등의 NIC 설정 정보 확인 가능 - 네트워크 패킷 덤프를 위해 확인이 필수적임 * 명령어 출력 정보 분류 - lo, ens33 = NIC - UNKNOWN = NIC가 작동하나, 연결이 없는 상태 - UP = NIC 작동 및 연결이 있는 상태
무차별 수집	* ifconfig eth숫자 promisc - 수사관 성향에 따라 사용 여부가 다름 - 보통 네트워크 차단, 전체 수집이 필요한 상황 등에 따라 사용
Network Dump	* tcpdump -i eth숫자 -vv -w 파일명.pcap - 지정한 eth 대상으로 파일명.pcap이란 패킷 덤프 파일을 기록 * wireshark 파일명.pcap - tcpdump로 생성된 파일의 네트워크 통신 이력 추적 진행 * tcpdump -i any - 모든 인터페이스에 대한 네트워크 통신 Dump
소켓 정보	* netstat -ss - 활성화 된 소켓 정보에 대한 수집 명령어 * netstat -rn - 라우팅 테이블 정보 수집 * netstat -anop - 최적 옵션 설정으로, 출력 결과 전체를 볼 필요는 없으며, 명령어 상단에 tcp, udp. raw 등의 프로토콜 정보를 한정하여 선별 확인을 권장
포트 정보	* nmap -sT 또는 -sU 대상 - netstat으로 수집하지 못한 열린 포트에 대해 추가 정보 수집이 가능 * lsof -i -P -n - nmap과 동일 목적으로 사용 가능한 명령어 (list open file)

실행중인 파일 정보	*lsof -u 유저명 - 유저 이름을 별도로 지정하여 실행중인 파일에 대한 정보를 출력
마운트 정보 출력	* mount 또는 df -TH - 마운트 된 디스크 정보 출력 명령어
커널 모듈 정보	* lsmod - 일종의 라이브러리 정보로 로드 된 커널 모듈을 탐색하는 명령어 - 어떤 모듈이 의심스러운지 찾기 위한 전체 리스팅이 목적 * modinfo 커널모듈이름 - 분석할 커널 모듈 이름을 입력하여 상세 정보를 출력 - 기본 설치 모듈이 아닐경우 의심해야함
사용자 이벤트	* ausearch -ui 숫자 -i - uuid 기반으로 사용자 이벤트를 검색하며 일종의 감사 목적의 로그 (별도 설치가 필요한 명령어) - 해당 감사 로그를 설정하지 않았다면, /var/log/audit/audit.log 정보 (auditd)가 없음 * auditd 설정 방법 1) auditd 설치 2) cd /etc/audit/plugins.d 3) vi 또는 mousepad를 통해 syslog.conf 파일 열기 4) 파일 내용 중 active 정보를 yes로 변경 후 저장하여 로깅 활성화 (active = yes) 5) service auditd restart * ausearch -ui 숫자 -i | grep -i session 및 ausearch -ui 숫자 -i | grep service - auditd가 설정되어 적재되는 로그들이 존재할 시, grep -i를 통해 추적 - 특히 session과 service로 필터링하여 서비스 유형, 시간 정보, uid, service_start/stop, session_open/close 등의 정보를 상세히 보길 권장

ELF 파일 정보	* readelf -a 파일명 - 악성코드 초동 분석 시, 가장 많이 사용하는 명령어 - 재부팅 여부에 따라서 해당 정보가 달라질 수 있기 때문에 휘발성 정보로 분류 * .rela.plt section - 해당 섹션에 나온 정보는 elf 파일에서 사용했던 함수의 정보를 확인할 수 있음 - 보통 해당 부분에서 socket 함수(외부 통신)를 가장 많이 찾음
프로세스 상세 정보	* pstree - 프로세스들을 상호 연관된 경우 트리 형태로 출력해주는 명령어 - 프로세스 정보 파악 시, 우선적으로 실행하여 연관 프로세스에 대한 파악 후 추가 확인 진행 * top - CPU ,Memory 점유 정보를 출력하며, 리소스 과다 점유 프로세스는 채굴기인 경우가 다수 * ps -aux - 시스템 / 유저 프로세스에 대한 구분을 하며 확인 진행을 권장 - [대괄호]로 묶인 프로세스의 경우 커널의 자체 데몬 프로세스로 굳이 볼 이유는 없음 (유저 프로세스를 중점으로 확인하길 권장)
SWAP 영역 정보	* cat /proc/swaps - 메모리 스왑 영역에 대한 정보 출력
Disk 파티션 정보	* cat /proc/partitions - 디스크 파티션 정보 출력
Kernel 메세지 로그	* dmesg - 커널 로그를 출력해주며 연결된 장치 정보 및 오류 정보 등의 중요 정보가 다수 저장
실행중인 서비스 정보	* systemctl --state=active --type=service - Live Forensic 환경에서 실행중인 서비스를 확인할 수 있는 명령어

Memory Dump	* 해당 Live 시스템의 Memory가 저장한 모든 정보를 파일로 저장하는 것 - volatility 툴을 많이 사용하며, 분석에 Profile 또는 ISF 정보가 필요하기 때문에 과정이 복잡함

3.2.4. 비휘발성 데이터 수집 고려사항

분류	명령어/경로 및 내용
시스템 정보	* cat /proc/cpuinfo - cpu 정보를 출력하나, cpu의 취약점 발현이 쉽지 않아서, 필수적이진 않음 * cat /proc/self/mounts - 명령어가 실행 된 마운트 지점에 대한 출력
커널 정보	* cat /etc/os-release - Linux OS 정보가 상세히 담겨 있는 텍스트 파일 내용 출력 * uname -r - Live 상태로 확인되는 커널 버전 정보 * cat /proc/version - File 상태로 확인되는 커널 정보 * cat /etc/hostname - hostname 명령어와 동일한 출력이지만, 공격자가 변조시키는 Case도 존재 ** 커널 정보의 비교는, 동일 OS를 구축하여, 동일 경로에 대한 비교로 빠르게 선별할 수 있음

사용자 계정 정보	* cat /etc/passwd (사용자 계정 정보 파일) - 사용자 이름 - shadow password - UID - GID - Comment - Home Directory - 사용자 절대 경로 로그인 쉘 (단, nologin으로 입력되어 있을 시, 사용자가 접근할 수 없는 계정) ex) root:x:0:0:root:/root:/usr/bin/zsh - 해당 정보는 특히, root 권한으로 사용되는 계정 여부 확인이 필수적 * cat /etc/shadow (사용자 암호 정보 파일) - 계정명 - Hash + Salt를 조합한 Password (첫 시작 부분의 "$문자$" 사이의 문자 값에 따라 어떠한 암호 알고리즘을 사용하는지 식별 가능하며, 해당 정보는 암호를 Crack할 수 있는지 여부를 판단하는데 사용) - john the ripper 툴과 pass 사전 파일을 통해서 복호화를 수행하는 경우 다수 존재 * cat /etc/group (사용자가 속할 수 있는 Group에 대한 정보 파일) - root 권한을 행사 할 수 있는 그룹에 속한 계정이 어떤 것들인지 식별하는 것이 중요 - root, lxd, wheel(su), sudo 그룹이 대표적으로 권한 상승이 가능한 그룹 * cat /etc/sudoers (root 권한으로 상승이 가능하도록 설정 사항을 기록한 파일)
현재 로그인 된 사용자, 로그인 History	* w - wtmp Log 파일에 대한 정보 중 로그인한 사용자가 어떤 방식으로 로그인 했는지 등을 확인 가능 - tty (컴퓨터와 직접 연결된 통신 포트) 및 pts (원격지 터미널 정보 or 가상 터미널 연결) 등 확인 가능 * last - /var/log/wtmp Binary 로그 파일 내용을 평문으로 출력 - sudo 명령 사용 이력, 인증, 원격 사용자 로그인 등을 확인 가능 (사용자 계정과 시간 정보를 Check 하는것이 핵심)

	* lastb - /var/log/btmp를 참조하여 마지막 로그인 정보 또는 과거에 잘못 접속한 계정의 이력을 확인 가능
자동 실행, 스케줄링 정보	*/home/계정명/.bash_profile - root를 제외한 다른 계정의 홈 디렉토리에 존재하는 사용자 별 자동 실행 프로그램에 대한 설정 * /etc/profile - 전체 사용자를 대상으로 적용되는 자동 실행 프로그램 설정 - 실행 권한(+x)이 존재하는 파일 이름을 삽입하여 무한 실행하는 등의 공격 수행 - 해당 File 악용 Case가 상당히 많은편 * /etc/init.d - 부팅 시, 자동으로 실행되는 Binary 정보 * /etc/service - 지정된 포트, 프로세스 등을 자동으로 허용하는 파일 * cron - 스케줄러로 시간, 요일, 조건에 의해 실행 (= Windows 작업 스케줄러) - Linux 버전에 따라 이름이 상이하며 cron도 경로, 종류가 다양함 1) /var/log/cron 2) /etc/cron.d 3) /etc/cron 4) /etc/crontab 5) /var/spool/cron/crontabs - crontab 또는 crontabs 파일 내용을 설정해야 cron 관련 경로에 배치 된 실행 권한이 존재하는 파일이 실행 - 단순히 경로에 파일만 넣는 경우 실행이 되지 않음 / 단, 공격자가 사용 후 crontab 내용 삭제는 가능

Linux Log File	* /var/log - cron, web, system, application 등의 다수 로그 확인 가능 - 각각의 로그 디렉토리는 access.log (성공), error.log (실패) 정보가 존재 - 특히 error.log를 우선적으로 보며, 실패한 최초 시간을 확인하여 선별 추적 진행이 가능 * /var/log/secure(구 버전) or /var/log/auth.log - 원격에서 접속한 사용자의 로그 확인이 가능 - 악성 서비스로 추측되는 로그가 존재할 시, syslog도 참조하여 해당 서비스가 어떤 명령어를 실행 했는지 확인할 때도 유용 * /var/run/utmp 또는 /var/adm/utmpx - 현재 로그인한 사용자 상태 정보를 담고 있는 파일 - w, who, finger 명령어로 확인 가능 * /var/log/wtmp 또는 /var/adm/wtmpx - 성공한 로그인/아웃 정보 및 System Boot/Shutdown 히스토리를 담은 파일 - last 명령어로 확인 가능 * /var/log/btmp 또는 /var/adm/loginlog - 실패한 로그인 정보를 담은 파일 - lastb 명령어로 확인 가능 (솔라리스는 text 파일) * /var/log/lastlog 또는 /var/adm/lastlog - 마지막으로 성공한 로그인 정보를 담은 파일 - lastlog 명령어로 확인 가능 (솔라리스 finger)
숨겨진 파일 및 디렉토리	* find /경로명 -name ".*" 2>/dev/null - 지정 경로에 대해 숨김 파일을 출력하는 명령어 - 단, /(최상위 경로) 기준으로 설정한 find 명령어는 부하가 크고 시간이 오래 걸려 권장하지 않음 * ls -al - 디스크가 Mount 된 경우는 해당 명령어로 숨김 파일 정보의 확인이 수월 - 이미징 확인 툴에서도 ".파일명" 형태로 즉시 숨김 파일 인지가 가능

File 서명 (일종의 File Header)	* xxd 파일명 - 파일의 처음 10 ~ 20byte에 존재하는 헤더 정보에서 확장자를 비교하여 서명 상태 확인 - hexa 값과 ASCII 값을 구분하여 확인하는 등의 방법이 존재 (비교적 번거로운 과정)
File 확장자	* file 파일명 또는 strings 파일명 - xxd 명령 대신 file 명령어로 쉽게 확장자 비교가 가능 - 파일이 특정되면 strings 명령어로 문자열을 뽑아내서 추가 분석 진행 *zip, jpg, png, pdf, xls, doc, exe, elf정도의 Header는 어떤식으로 ASCII, Hexa가 구성되어 있는지 숙지하는게 분석에 매우 도움
쓰기 가능 File	* find 경로 -writable -type f 2>/dev/null - 쓰기 권한(+x)이 존재하는 파일 확인
시간 정보	* touch -t 시작시간 start 후, touch -t 종료시간 end - 시작 시간을 지정한 파일명과, 종료 시간을 지정한 파일명을 통해 범위를 지정하여 시간 범위 내에서 수정된 파일의 검색이 가능 * find 경로 -newer start -a ! -newer end - 지정한 파일보다 더 최근에 생성되거나, 변경된 파일 검색 가능 * 시간으로 파일명 생성 시, yyyymmddhhmm.ss 값으로 설정이 가능 ex) touch -t 202305011123.00 start 및 touch -t 202305102359.00 end * MAC (Modify, Access, Change) - Modify : 파일 내용에 대한 변경 시간 - Access : 파일 접근에 대한 시간 - Change : 파일 속성값의 변경에 대한 시간 (단, Access 시, Change도 동일하게 변경) * stat 파일명 - live 상태에서 stat 명령어를 통해 파일 생성 시간(Birth)값 까지 추가적으로 확인 가능 (정적 상태에서 분석 시, Disk 이미지 분석 Tool에서 created time 정보를 토대로 타임라인 구성 가능

악성 의심 정보	* rkhunter --check --rwo - /dev 경로 위주로 탐색을 수행하는 악성 정보 탐색 도구 (별도 설치 필요) * chkrootkit - 시스템 Binary 파일을 검사하여, 루트킷이 만든 수정 사항을 감지하는 셸 스크립트 도구 (최근 침해 사고 동향 상, 루트킷 방식으로 숨기기 보다는 대놓고 퍼트려 놓는 방식이 다수)
패키지 설치 정보	* dpkg -l - 설치된 모든 패키지의 이름, 버전 정보 등 출력 * dpkg -L 패키지명 - 해당 패키지가 설치된 모든 경로를 출력
사용자 홈 디렉토리 숨김 파일	* /home/계정명/.local - 해당 디렉토리는 일종의 휴지통 역할로 최근 문서 열람, 파일 및 폴더 열람 정보 등을 확인 가능 - .local/share/recently-used.xbel 파일에 최근 열람한 문서 정보를 확인 가능 (visited, bookmark, application name 항목을 통해 시간, 파일명, 접근에 사용된 앱 이름 등 기록) * /home/계정명/.mozilla - 웹 브라우저 전용 로그 디렉토리로 사용자 별 해시값으로 이름을 가진 로그 디렉토리가 존재 (.mozilla/firefox) - places.sqlite : 방문 정보 - cookies.sqlite : Web Cookie 정보

4. 제안하는 포렌식 도구

본 장에서는 IoT 기기에서 수집한 로그들을 분석하는 딥러닝 알고리즘 기반의 포렌식 도구를 제안한다. IoT 기기에서 정상적인 동작시의 로그와 인위적인 접근을 한 후의 로그들을 수집하고, 이 데이터들로 딥러닝 모델(NLP 모델)을 학습시켜서 IoT 기기 내부에서 어떠한 침해 사례가 있었는지 알려주는 방식의 포렌식 도구이다.

포렌식 도구에 적용한 딥러닝 모델(NLP 모델)은 Google 에서 공개한 AI 언어 모델로, 문장을 인식해서 그 문장이 어떤 클래스에 속하는지 처리하는 태스크를 수행하는 모델 BERT (Bidirectional Encoder Representations from Transformers)이다. IoT 기기가 정상 동작하고 있을 때의 로그를 취합하여 이를 정상 클래스로, IoT 기기에 문제를 남겼을 때의 로그를 취합하여 이를 비정상 클래스로 분류하도록 BERT 를 학습시켜 이용하였다.

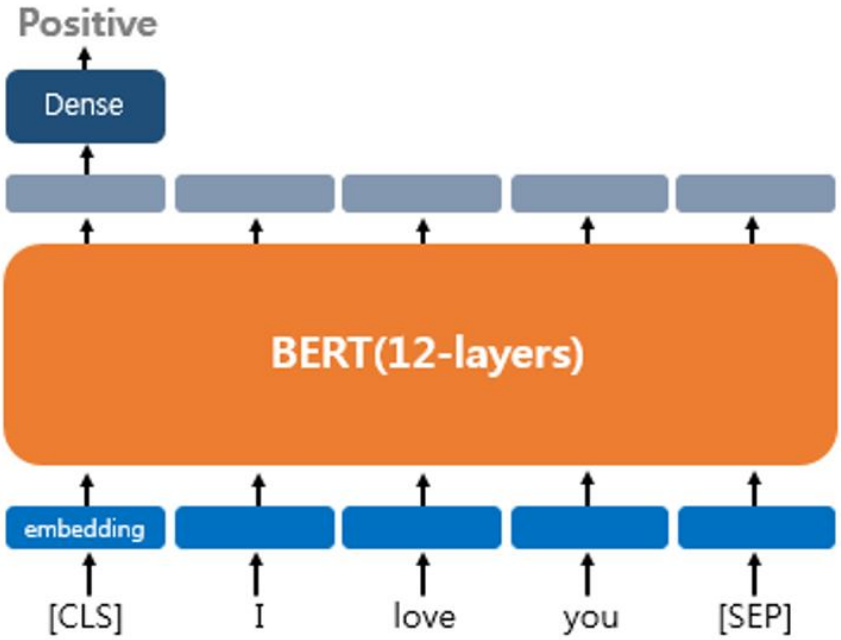

그림 3. BERT 모델 태스크 수행 과정

라이브 포렌식의 경우, IoT 기기에 실시간으로 로그를 취합하고 BERT 모델에 통과시키는 동작을 하는 포렌식 도구를 구현하려 하며, 오프라인 포렌식의 경우, IoT 기기에서 칩오프를 통해 얻은 데이터들을 BERT 모델에 통과시켜 분석하는 도구를 구현하려 한다.

4.1. 로그 추출

대부분의 IoT 기기들은 임베디드 리눅스 시스템으로 이루어져 있으며, 이러한 IoT 기기들의 로그는 주로 파일 시스템에서 /var/log 디렉토리에 저장된다.

경로	설명
/var/log/messages	리눅스 시스템 전체적인 로그
/var/log/secure	접속한 유저 인증 정보
/var/log/cron	crontab 에 등록된 예약 작업 실행 여부
/var/log/dmesg	시스템 부팅 시 기록되는 로그
/var/log/wtmp	최근 접속 사항 기록
/var/log/lastlog	각 계정의 가장 최근 로그인 기록
/var/log/syslog	syslog 가 생성하는 공통 로그

표에 정리된 내용은 앞서 보았던 /var/log 디렉토리에 저장되는 대표적인 리눅스 시스템의 로그 파일들이다. 이처럼 기기마다 /var/log 디렉토리에 저장된 로그 파일들을 추출하여 이용한다. 다만 본 연구에서 분석을 진행하였던 TP-Link Archer C2300 공유기는 /var/log 디렉토리에서 별도의 로그 내용을 확인할 수 없었다. 따라서 /var/log 디렉토리에서 별도의 로그 내용을 확인하기 어려운 일부 기기의 경우, 그림 5와 같이 logread 명령어를 사용하여 로그를 읽어내 이용한다.

그림 4. 분석을 진행하였던 TP-Link Archer C2300 공유기

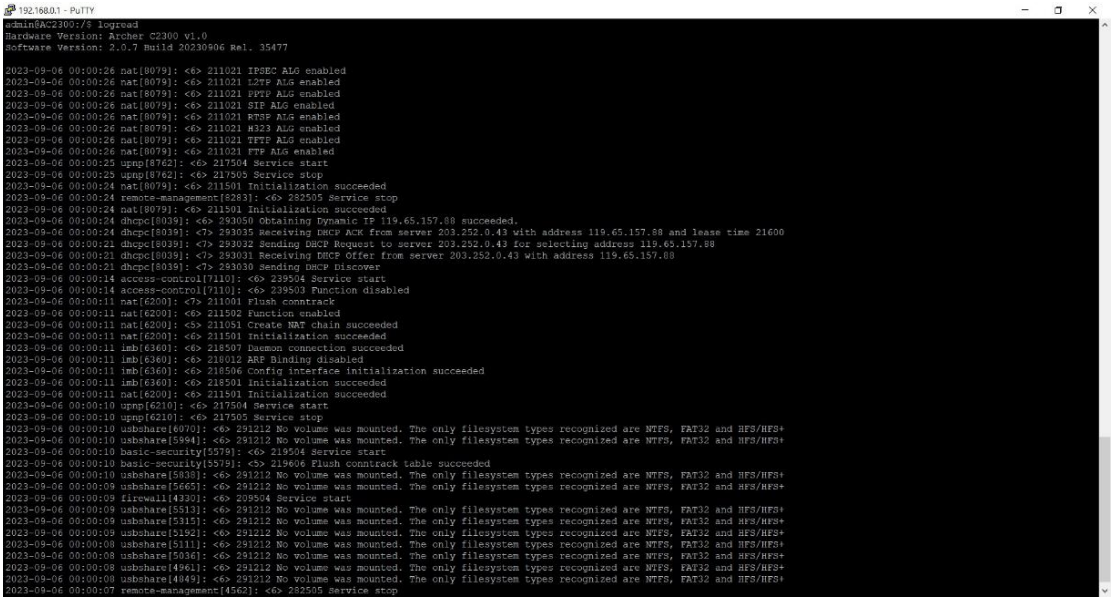

그림 5. TP-Link Archer C2300에서의 logread 명령어 입력 결과

그림 6 는 로그 데이터 샘플을 구하기 위해 리눅스 환경으로 구성되어 있는 웹 서버에 접속한 뒤, scp 를 통해 /var/log 디렉토리에 위치한 log 데이터들을 수집한 모습이다. 수집한 결과, lastlog, syslog, wtmp 등 다양한 로그 파일을 확인할 수 있었다.

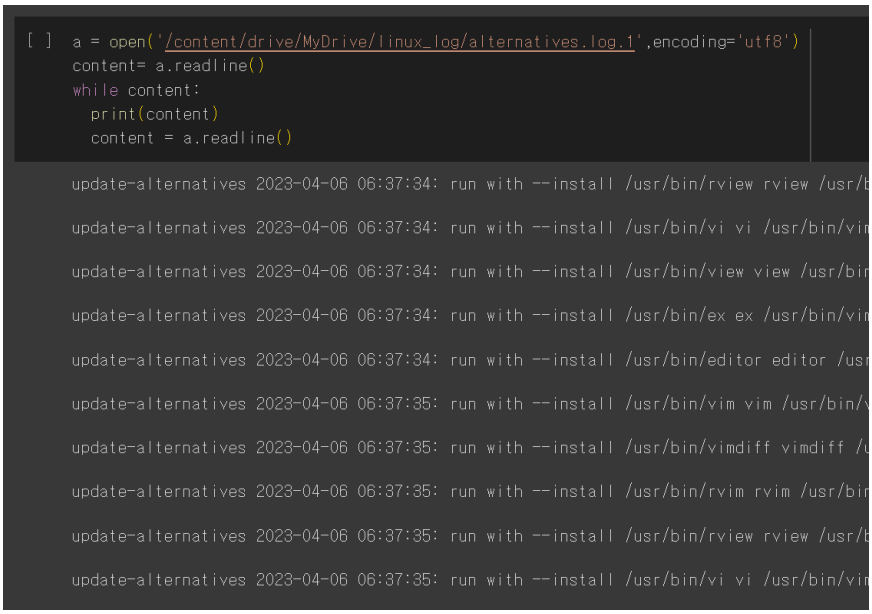

그림 6. 로그 데이터 예시

4.2. 로그 데이터 전처리

로그 데이터들을 수집한 후에 구글 Colab을 통해 Bert 모델이 알아들을 수 있는 형식으로 전처리를 진행하였다.

```python
# 데이터 전처리를 위한 Dataset 클래스 정의
class CustomDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_len):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = str(self.texts[idx])
        label = int(self.labels[idx])

        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=self.max_len,
            return_token_type_ids=False,
            pad_to_max_length=True,
            return_attention_mask=True,
            return_tensors='pt'
        )

        return {
            'text': text,
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'label': torch.tensor(label, dtype=torch.long)
        }
```

그림 7. 로그 데이터 전처리

4.3. BERT 모델 학습

4.3.1. 학습 과정

로그 데이터 전처리를 진행한 후, 아래와 같은 환경에서 BERT 모델을 훈련시켰다.

[학습 환경]

Optimizer : Adam (최적화 알고리즘 중 하나로, 경사 하강법의 변형)

loss : CrossEntropy (분류 문제에서 주로 사용되는 손실 함수로, 모델의 예측과 실제 레이블 간의 차이를 측정)

learning rate : 0.005 (모델이 가중치를 업데이트하는 정도)

batch_size : 32 (각 반복에서 모델이 처리하는 데이터의 샘플 수)

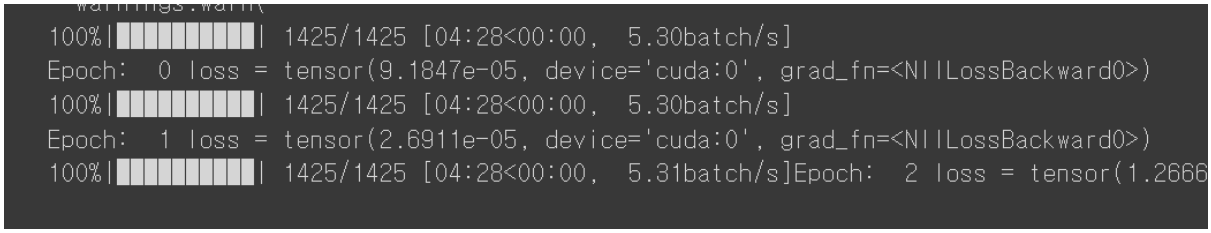

그림 8. BERT 모델 학습 과정

4.3.1. 학습 결과

학습시킨 모델을 확인하여 보기 위해 외부에서 접속 후 생긴 로그를 훈련시킨 모델에 입력해 본 결과, 최종적으로 "침투 흔적 발견"을 출력하는 것을 확인하였다.

```
# 침투 로그데이터 예
input_text = "iptables: denied incoming connection from 192.168.0.3 to port 22

# 텍스트를 토큰화하고 모델 입력 형식으로 변환
encoding = tokenizer.encode_plus(
    input_text,
    add_special_tokens=True,
    max_length=128,
    return_token_type_ids=False,
    pad_to_max_length=True,
    return_attention_mask=True,
    return_tensors='pt'
)

# 모델 예측 수행
model.eval()
with torch.no_grad():
    input_ids = encoding['input_ids'].to('cuda')
    attention_mask = encoding['attention_mask'].to('cuda')
    outputs = model(input_ids, attention_mask=attention_mask)
    logits = outputs.logits

# 로짓에서 예측값 얻기
predicted_class = torch.argmax(logits, dim=1).item()

# 예측 결과 출력
if predicted_class == 1:
    print("침투 흔적 발견")
else:
    print("문제 없음")
```

```
침투 흔적 발견
```

그림 9. 학습시킨 모델의 테스트 과정

4.4. 사례 연구

IoT 기기에서 활용이 가능한 것을 확인하기 위해 라즈베리 파이에 학습시킨 BERT 모델과 python 파일을 설치한 뒤, python 파일을 실행하여 /var/log 디렉토리를 주기적으로 체크하는 과정을 진행하였다.

외부에서 잘못된 비밀번호를 입력하여 라즈베리파이에 SSH 접속을 시도한 결과, 아래의 그림 10 에서 볼 수 있듯이 이 때 생긴 로그를 확인하고 비정상적인 접속이 있었음을 발견하여 "침투 흔적 발견"을 출력하는 것을 확인할 수 있다.

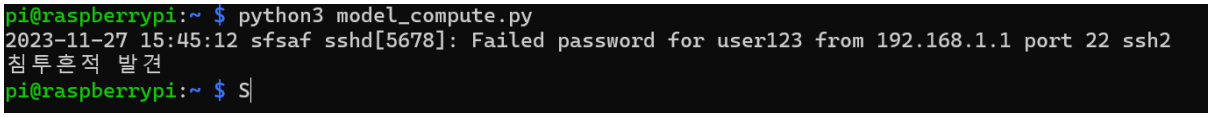

그림 10. 라즈베리 파이에서의 테스트 결과

본 과정을 통해 앞서 학습시킨 BERT 모델을 포렌식 도구에 적용하고, 해당 포렌식 도구를 실제 IoT 기기에서 작동시키며 활용해볼 수 있음을 기대할 수 있다.

5. 결론 및 기대효과

본 연구는 리눅스 기반의 OpenWrt, BusyBox 등으로 구성된 IoT 기기에 대한 디지털 포렌식 도구를 개발하고자 하는 데 주요 목표를 두고 진행되었다. 이러한 디지털 포렌식 도구의 개발은 다음과 같은 결론을 도출한다.

첫 번째로 IoT 기기의 보안 강화이다. IoT 기기는 점점 더 중요한 역할을 하고 있으며, 이에 따라 보안 문제가 증가하고 있다. 본 연구에서 개발한 포렌식 도구는 IoT 기기의 로그 데이터를 통해서 실시간/비실시간으로 문제상황을 식별하고 알려줌으로써 보안 강화에 도움을 줄 것으로 기대된다.

두 번째는 수사기관의 디지털 포렌식 능력 향상이다. IoT 기기에 대한 수사는 피해자가 침해사례를 인지한 후에 진행된다. 인지가 늦어질 경우, 수사가 늦어지게 되어 어려움을 겪는다. 이때 디지털 포렌식 전문가와 수사기관은 이 도구를 활용하여 IoT 기기에서 실시간으로 침해사례를 인지하며, 빠른 로그 분석을 통해 증거 수집을 더욱 효과적으로 수행할 수 있을 것이다. 이를 통해 범죄 조사, 기업 보안 사고 대응, 사회 문제 해결에 대한 능력이 향상될 것이다.

세 번째로는 사회적 안전망 구축이다. 이 연구를 통해 IoT 기기에 대한 침해를 사용자가 빠르게 확인할 수있다는 점에서 범죄 예방 및 조사, 개인 정보 보호, 기업 보안, 사회적 문제 해결과 같은 다양한 분야에서 기여할 수 있다.

이 도구의 개발을 통해 이러한 도전 과제를 해결하고 기술적인 능력을 향상시킬 수 있다.

6. 향후 연구

향후 연구로는 정기적으로 실행되는 자동화된 작업을 지정하는 데 사용되는 시스템 스케줄러인 cron job 을 활용하여 주기적으로 로그들을 수집하여 모델을 거쳐 침해활동이 있는지 확인하도록 할 계획이다. 그 후 이를 웹서버로 전송하여 실시간으로 많은 IoT 기기들을 동시에 확인할 수 있는 도구로 발전할 수 있도록 연구할 계획이다.

추가적으로 로그 데이터뿐만 아니라 침해 사례를 인지할 수 있는 데이터는 파일시스템 내에 다양하게 존재하는데, 추후에는 파일 시스템을 분석하여 인지할 수 있도록 확장하는 연구를 진행할 예정이다.

7. 참고문헌

[1] 정익래, 홍도원, 정교일, "디지털 포렌식 기술 및 동향", 전자통신동향분석 제 22 권 제 1 호

[2] 정규식, 김정길, 곽후근, 장훈, "유무선공유기를 이용한 임베디드 리눅스 시스템 구축 및 응용"

[3] 이진오, 손태식, "IoT 플랫폼에 탑재되는 안드로이드 및 리눅스 기반 파일시스템 포렌식"

[4] 박현진, "인공지능(AI) 언어모델 'BERT(버트)'는 무엇인가", 인공지능 신문