

IPTIME DBG

문서번호 Ver. 1.0

개 정 이 력

[illegible]¹ 변경 사유: 변경 내용이 이전 문서에 대해 최초작성/승인/추가/수정/삭제 중 선택 기입

² 변경 내용: 변경이 발생하는 위치와 변경 내용을 자세히 기록(장.절과 변경 내용을 기술한다.)

목 차

1. 개요	4
1.1 본 문서의 목적.....	4
2. UART	4
2.1 USB to UART.....	5
3. AFL FUZZER	10
3.1 AFL Logic	10
3.2 AFL Fuzzer 설치 및 사용법	11
3.3 ASAN 설치 및 사용법.....	16
4. FIRMAE	17
4.1 설치 및 사용법.....	17
4.2 동적 디버깅	18

1. 개요

1.1 본 문서의 목적

본 문서는 IOT 진단을 수행할 경우, 신규 취약점에 대한 정보 및 점검 절차를 기술하기 위한 문서로, 신규 입사자와 IOT 를 처음 수행하는 자에게 쉽게 따라 할 수 있도록 하는 것이 그 목적이다.

[분석 도구]

분석도구	버전	기능
IPTIME	ipTIME N604 black 9.98.2	분석대상
Kali linux	v2021-4a	분석 환경 OS
binwalk	v2.2.1	펌웨어 이미지 내의 파일과 코드를 추출
FirmAE	-	펌웨어 에뮬레이션 구동 자동화 프레임워크
gef	-	리버스 엔지니어링 도구
gdb-multiarch	v3	크로스컴파일러 아키텍처 분석 도구
AFL fuzzer	2.52b	Fuzzer 도구

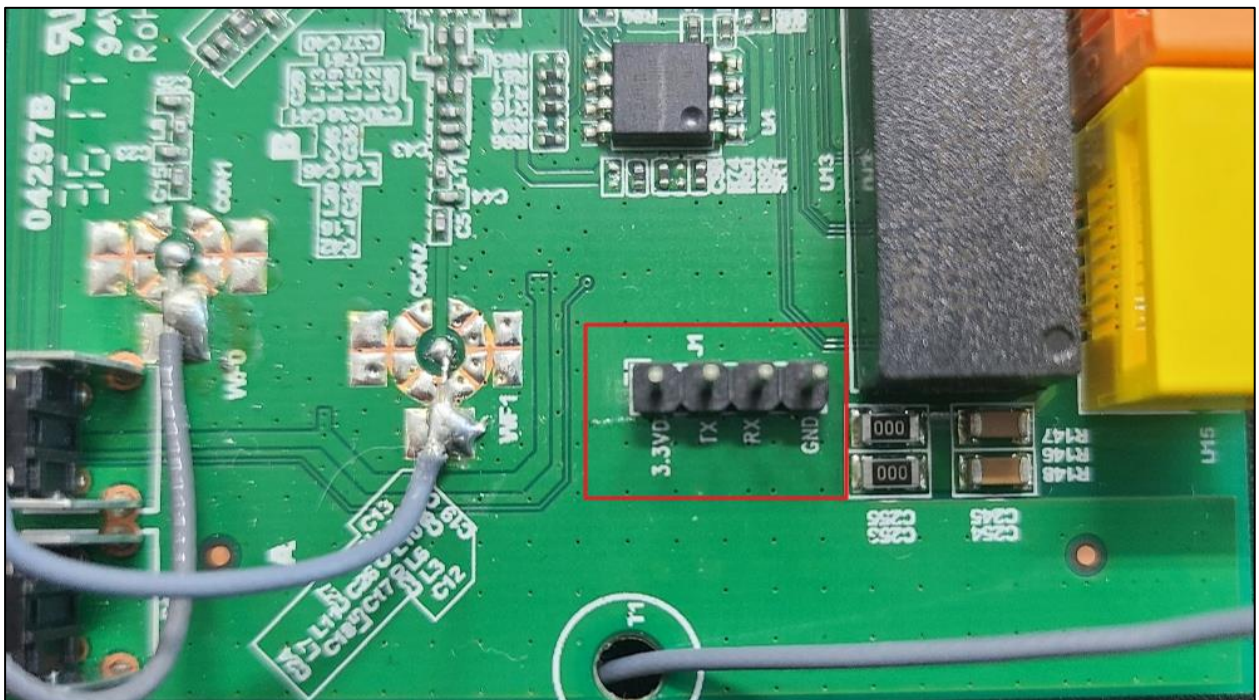
2. UART

UART 는 병렬 데이터의 형태를 직렬 방식으로 전환하여 데이터를 전송하는 컴퓨터 하드웨어의 일종이다. UART 핀을 찾아 얻을 수 있는 데이터는 커널, OS 메시지, 디버그 메시지, 오류 메시지, 숨겨진 메뉴, Command Shell 등이 있다. UART 핀을 찾는 방법은 여러가지 방법이 존재하지만 본 문서에서는 ipTIME 을 기준으로 설명한다.

2.1 USB to UART

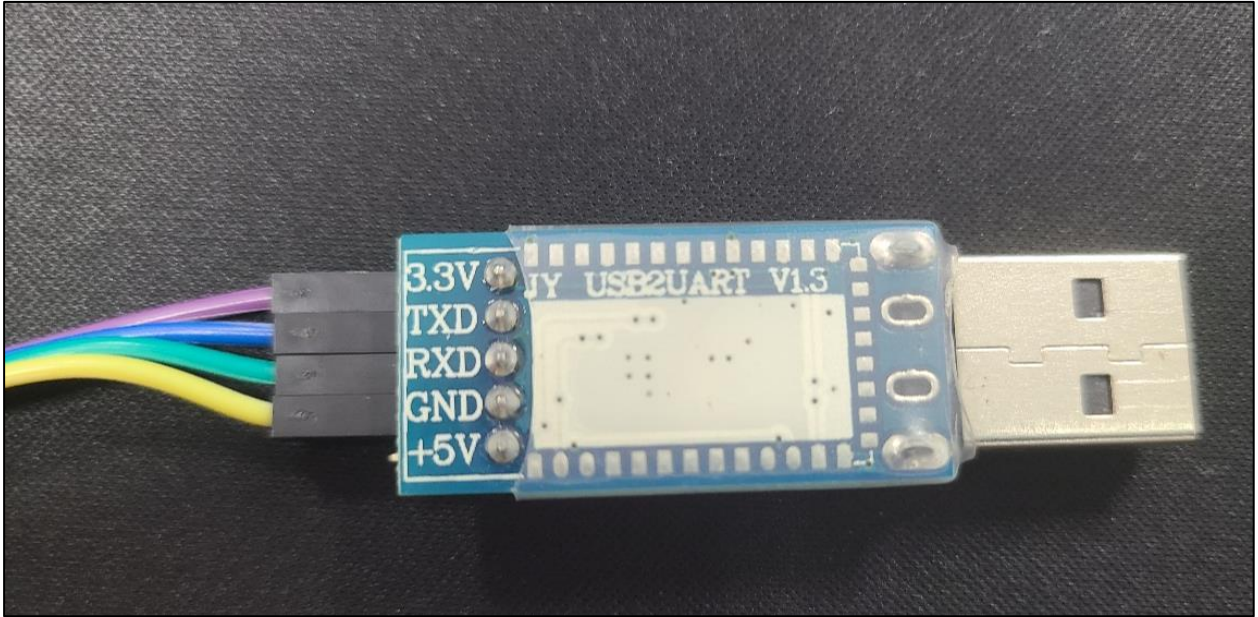


진단 시 분석대상이 될 ipTIME 과 USB to UART 가 필요하다. USB to UART 와 점퍼는 ipTIME 의 UART 포트에 연결하여 PC 와 연결하기 위해서 반드시 구비가 되어 있어야 한다.



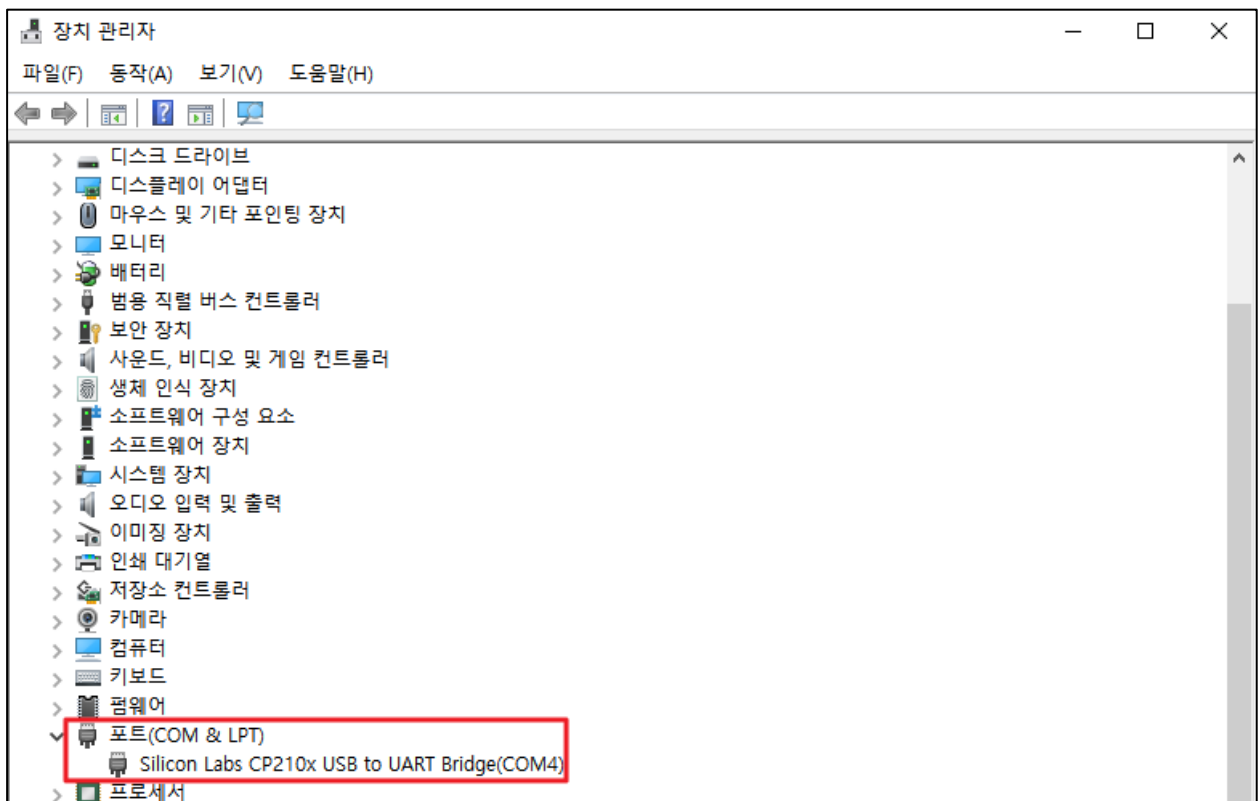
ipTIME 의 경우 UART 핀이 제거되지 않아 분해 시 쉽게 찾아볼 수 있다. UART 핀은 3.3VD, TX, RX, GND 로 구성되어 있으며 각 핀의 역할은 다음과 같다.

3.3VD : 3.3VCC 전압, **TX** : 데이터 송신 핀, **RX** : 데이터 수신 핀, **GND** : 그라운드 핀

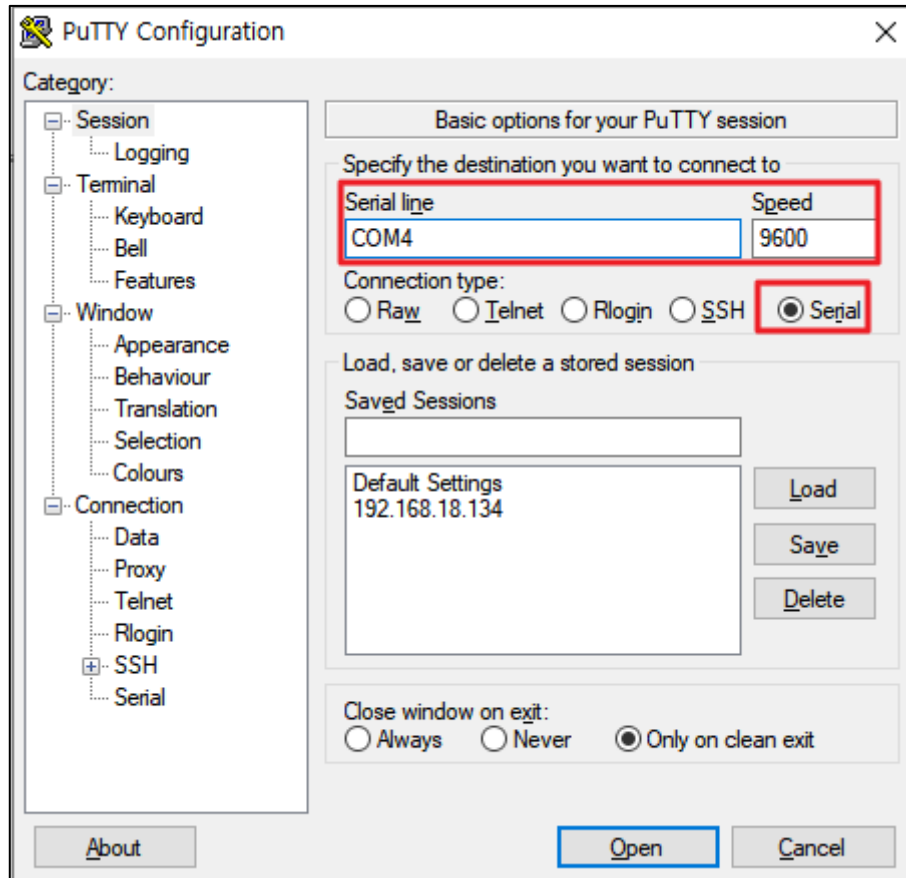


USB to UART 변환기에도 UART 각 핀에 대한 설명이 있으며 점퍼를 이용하여 ipTIME 과 연결하면 된다. 연결 시 에는 ipTIME 과 PC 간의 통신이 이루어 지도록 TX 와 RX 핀을 교차하여 연결해 준다.

ex) TX → RX 연결, RX → TX 연결



케이블을 연결한 이후에는 장치관리자에서 USB 포트가 인식이 되는지 확인하고 인식이 되지 않는 경우 CP2102, PL2303, FTDI 등과 같은 USB 드라이버를 설치해야 한다.



연결 이후 원격접속 프로그램을 이용하여 시리얼 포트에 접속을 시도한다. 접속할때 사용되는 Speed 는 Baudrate, Serial Port 라고도 불리며 해당 값은 무작위대입을 통해 구해야 한다. 시리얼 포트는 115200, 57600, 38400, 19200, 9600 을 사용한다.

```
COM4 - PuTTY
Movable zone start PFN for each node
early_node_map[1] active PFN ranges
0: 0x00000000 -> 0x00002000
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 8128
Kernel command line: console=ttyS1,57600n8 root=/dev/mtdblock5 console=
PID hash table entries: 128 (order: -3, 512 bytes)
Dentry cache hash table entries: 4096 (order: 2, 16384 bytes)
Inode-cache hash table entries: 2048 (order: 1, 8192 bytes)
Primary instruction cache 64kB, VIPT, 4-waylinesize 32 bytes.
Primary data cache 32kB, 4-way, PIPT, no aliases, linesize 32 bytes
Writing ErrCtl register=00028cf8
Readback ErrCtl register=00028cf8
Memory: 28152k/32768k available (3163k kernel code, 4564k reserved, 712k data, 144k init, 0k highmem)
NR_IRQS:128
console [ttyS1] enabled
Calibrating delay loop... 381.95 BogoMIPS (lpj=763904)
pid_max: default: 32768 minimum: 301
Mount-cache hash table entries: 512
NET: Registered protocol family 16
bio: create slab <bio-0> at 0
Switching to clocksource Ralink Systick timer
NET: Registered protocol family 2
--> ipconflict proc OK !!!!!!!!!!!
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 1024 (order: 1, 8192 bytes)
TCP bind hash table entries: 1024 (order: 0, 4096 bytes)
TCP: Hash tables configured (established 1024 bind 1024)
TCP reno registered
```

해당 ipTIME 은 57600 포트로 연결되었으며 연결 시 커널, OS 메시지가 무한루프되어 커맨드 쉘을 획득할수 없는 상태이다. 하드웨어는 중지되지 않는 무한루프의 특징을 가진다.

번호	제목	날짜	조회
공지 01	ipTIME Mesh 접속기 v1.24	2021-10-06	533349
공지 02	ipTIME 검색기 2.32	2021-10-27	438071
공지 03	ipTIME NAS 도우미	2019-04-17	333710
공지 04	ipTIME 11ac/11n USB 무선랜카드 MEDIATEK 드라이버	2018-01-03	1169415
공지 05	ipTIME NAS 유틸리티 v1.72 (ipDISK Drive, ipTIME Cloud)	2021-11-22	1804763

ipTIME은 무한루프에서 빠져나오기 위해 magic key 값이 존재한다. 해당 key 값을 확인하기 위해 공식 홈페이지에서 OS 메시지에서 확인된 해당 버전의 동일 펌웨어를 다운받는다.

```
code@code-virtual-machine:~/FirmAE$ ls -al | grep n604bl
-rw-rw-r-- 1 code code 3441916 1월 10 15:04 n604bl_kr_9_982.bin
code@code-virtual-machine:~/FirmAE$ binwalk -e ./n604bl_kr_9_982.bin

DECIMAL      HEXADECIMAL    DESCRIPTION
-----
0            0x0            uImage header, header size: 64 bytes, header CRC: 0xC4EC2D86, created: 2016-12-13 09:20:56, image size: 3441
852 bytes, Data Address: 0x80000000, Entry Point: 0x8000C120, data CRC: 0x8DE743AC, OS: Linux, CPU: MIPS, image type: OS Kernel Image, com
pression type: lzma, image name: "n604bl"
64           0x40           LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes, uncompressed size: 4119628 bytes

WARNING: Extractor.execute failed to run external extractor 'sasquatch -p 1 -le -d 'squashfs-root-0' '%e': [Errno 2] No such file or dire
ctory: 'sasquatch', 'sasquatch -p 1 -le -d 'squashfs-root-0' '%e' might not be installed correctly

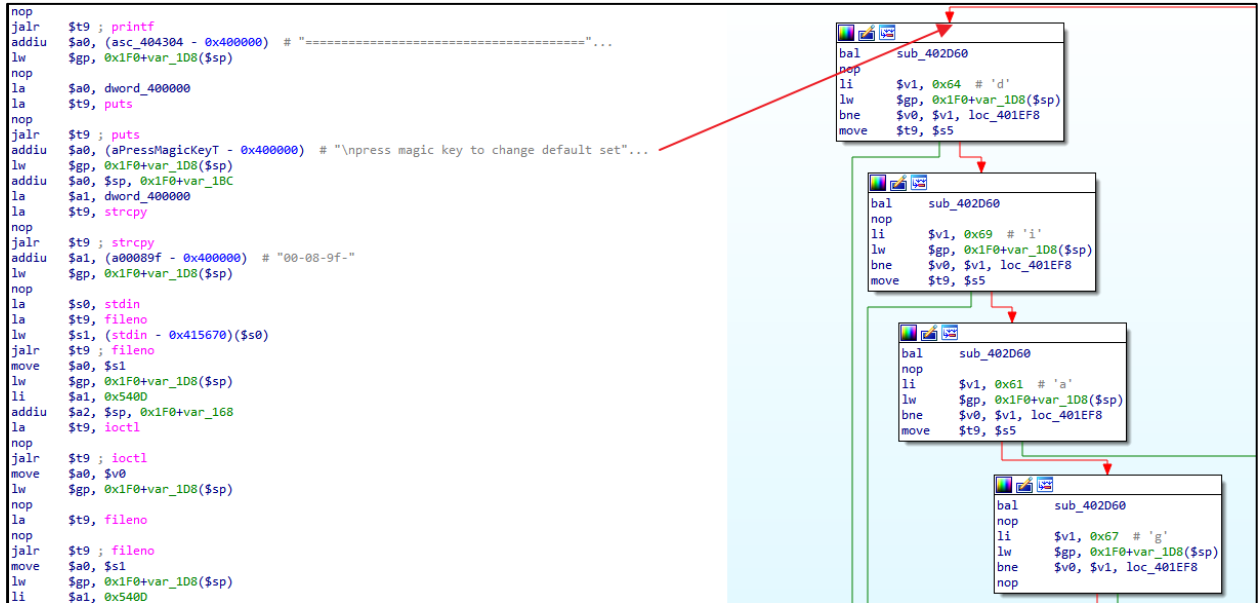
WARNING: Extractor.execute failed to run external extractor 'sasquatch -p 1 -be -d 'squashfs-root-0' '%e': [Errno 2] No such file or dire
ctory: 'sasquatch', 'sasquatch -p 1 -be -d 'squashfs-root-0' '%e' might not be installed correctly
1344764      0x1484FC       Squashfs filesystem, little endian, version 4.0, compression:xz, size: 2094554 bytes, 1261 inodes, blocksize
: 131072 bytes, created: 2016-12-13 09:20:53

code@code-virtual-machine:~/FirmAE$
code@code-virtual-machine:~/FirmAE$ ls -al | grep n604bl
-rw-rw-r-- 1 code code 3441916 1월 10 15:04 n604bl_kr_9_982.bin
drwxrwxr-x 4 code code 4096 1월 13 09:20 _n604bl_kr_9_982.bin.extracted
```

다운로드한 펌웨어는 binwalk 를 이용하여 코드와 파일을 추출한다.

```
code@code-virtual-machine:~/FirmAE/_n604bl_kr_9_982.bin.extracted/squashfs-root$ pwd
/home/code/FirmAE/_n604bl_kr_9_982.bin.extracted/squashfs-root
code@code-virtual-machine:~/FirmAE/_n604bl_kr_9_982.bin.extracted/squashfs-root$ ls
bin  cgin default dev etc etc_ro home lib linuxrc ndbin plugin proc save sbin sys tmp upgrade-bin usr var
code@code-virtual-machine:~/FirmAE/_n604bl_kr_9_982.bin.extracted/squashfs-root$
```

추출하게 되면 “/_n604bl_kr_9_982.bin.extracted/squashfs-root” 위치에 파일이 추출된 것을 확인할 수 있다.

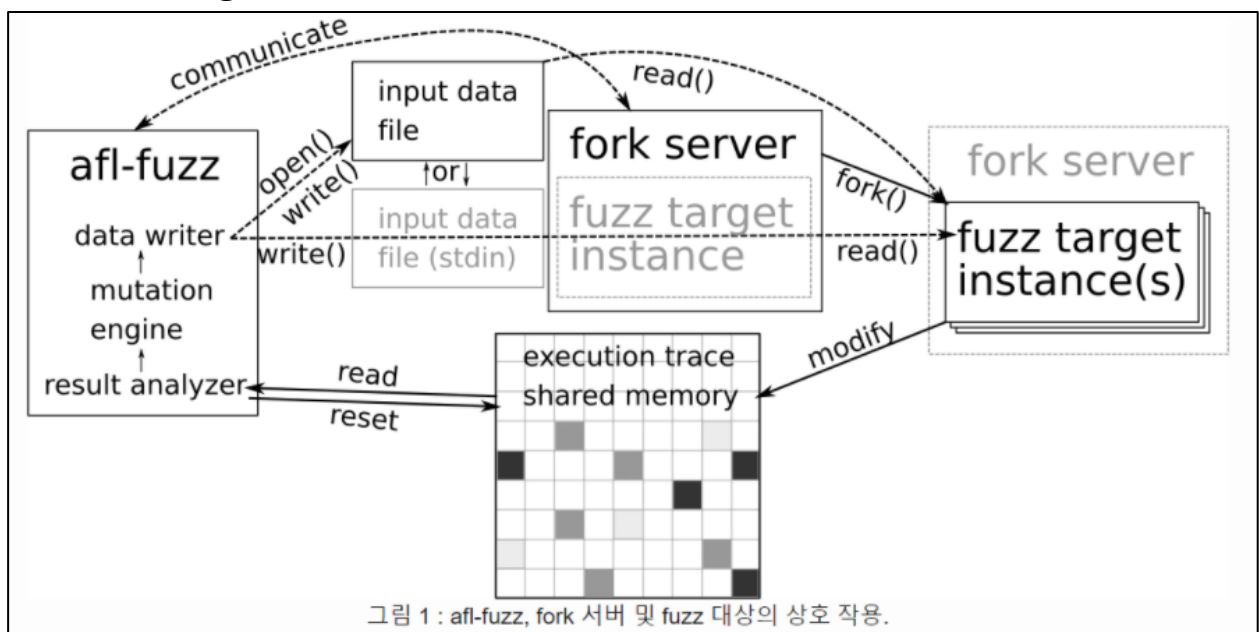


IDA 를 이용하여 /sbin/inittime 파일 분석 시 magic key 값을 "diag"로 받는 것을 확인할 수 있다.

3. AFL Fuzzer

AFL 은 American Fuzzy Lop 의 약자로, instrumentation-guided genetic algorithm 과 결합된 fuzzer 이다. 브루트포스로 입력을 받으며 거기서 끝나는 것이 아니라, 커버리지를 넓혀가며 프로그램 제어 흐름에 대한 변경사항을 기록하고, 이를 로깅하여 유니크한 크래시를 발견해 낼 수 있다. 또한 AFL Fuzzer 는 커버리지 기반 fuzzer 이기 때문에 매우 효율적인 퍼징이 가능하며, 코드 커버리지 측정을 위한 코드를 컴파일 타임에 삽입한다. QEMU 를 이용하여 컴파일 타임이 아닌, 런타임시에 코드 삽입도 가능하다.(코드 삽입을 Instrumentation 이라고 한다.)

3.1 AFL Logic



위 그림은 퍼징 로직을 대략적으로 보여준다.

1. afl-fuzz 에서 퍼징하려고 한 대상 프로그램을 실행한다.
2. 해당 프로그램이 처음으로 실행되면서 afl-fuzz 와 pipe 로 통신을 하면서 fork server 를 만들고, 새로운 타겟 인스턴스를 fork call()로 실행한다.
3. 표준 입력 or file 로 들어온 입력이 fuzz 대상 프로그램으로 전달된다.
4. 실행된 결과를 공유 메모리에 기록하여 실행을 완료한다.
5. afl-fuzz 는 공유 메모리에서 fuzzer 대상이 남긴 기록을 읽고 이전 항목을 변경하여 새로운 입력을 만든다.
6. 새롭게 만든 입력은 다시 프로그램에 들어가서 실행된다.

공유 메모리를 통해 새로운 입력을 만드는데 이 부분이 코드 커버리지를 높이기 위한 로직으로, 컴파일시 branch 에 다음과 같은 코드가 삽입된다.

```
cur_location = <COMPILE_TIME_RANDOM>;
shared_mem[cur_location ^ prev_location]++;
prev_location = cur_location >> 1;
```

3.2 AFL Fuzzer 설치 및 사용법

```

root@kali: /home/kali
File Actions Edit View Help

(root@kali)~/home/kali
└─ wget http://lcamtuf.coredump.cx/afl/releases/afl-latest.tgz
--2022-01-12 19:30:55-- http://lcamtuf.coredump.cx/afl/releases/afl-latest.tgz
Resolving lcamtuf.coredump.cx (lcamtuf.coredump.cx) ... 172.67.180.222, 104.21.83.192, 2606:4700:3036::ac43:b4de, ...
Connecting to lcamtuf.coredump.cx (lcamtuf.coredump.cx)[172.67.180.222]:80 ... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://lcamtuf.coredump.cx/afl/releases/afl-latest.tgz [following]
--2022-01-12 19:30:57-- https://lcamtuf.coredump.cx/afl/releases/afl-latest.tgz
Connecting to lcamtuf.coredump.cx (lcamtuf.coredump.cx)[172.67.180.222]:443 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 835907 (816K) [application/x-gzip]
Saving to: 'afl-latest.tgz'

afl-latest.tgz          100%[=====] 816.32K  335KB/s   in 2.4s

2022-01-12 19:31:00 (335 KB/s) - 'afl-latest.tgz' saved [835907/835907]

(root@kali)~/home/kali
└─ tar -xvf afl-latest.tgz
afl-2.52b/
afl-2.52b/afl-as.h
afl-2.52b/libtokencap/
afl-2.52b/libtokencap/libtokencap.so.c
afl-2.52b/libtokencap/README.tokencap
afl-2.52b/libtokencap/Makefile
afl-2.52b/alloc-inl.h
afl-2.52b/config.h
afl-2.52b/test-instr.c
afl-2.52b/test-instr.c

```

afl fuzzer 를 사용하기 위해 파일을 다운 받고 압축해제한다.

```

root@kali: /home/kali/afl-2.52b
File Actions Edit View Help

(root@kali)~/home/kali
└─ cd afl-2.52b

(root@kali)~/home/kali/afl-2.52b
└─ ls
afl-analyze.c  afl-fuzz.c      afl-showmap.c  config.h      experimental  llvm_mode      README
afl-as.c      afl-gcc.c      afl-tmin.c     debug.h       hash.h        Makefile       testcases
afl-as.h      afl-gotcpu.c   afl-whatshup   dictionaries  libdislocator  qemu_mode      test-instr.c
afl-cmin      afl-plot       alloc-inl.h    docs          libtokencap    QuickStartGuide.txt  types.h

(root@kali)~/home/kali/afl-2.52b
└─ make
[*] Checking for the ability to compile x86 code...
[+] Everything seems to be working, ready to compile.
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl" -DDOC_PATH="/usr/local/share/doc/afl" -DBIN_PATH="/usr/local/bin" afl-gcc.c -o afl-gcc -ldl
set -e; for i in afl-g++ afl-clang afl-clang++; do ln -sf afl-gcc $i; done
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl" -DDOC_PATH="/usr/local/share/doc/afl" -DBIN_PATH="/usr/local/bin" afl-fuzz.c -o afl-fuzz -ldl
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl" -DDOC_PATH="/usr/local/share/doc/afl" -DBIN_PATH="/usr/local/bin" afl-showmap.c -o afl-showmap -ldl
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl" -DDOC_PATH="/usr/local/share/doc/afl" -DBIN_PATH="/usr/local/bin" afl-tmin.c -o afl-tmin -ldl
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl" -DDOC_PATH="/usr/local/share/doc/afl" -DBIN_PATH="/usr/local/bin" afl-gotcpu.c -o afl-gotcpu -ldl
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl" -DDOC_PATH="/usr/local/share/doc/afl" -DBIN_PATH="/usr/local/bin" afl-analyze.c -o afl-analyze -ldl
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl" -DDOC_PATH="/usr/local/share/doc/afl" -DBIN_PATH="/usr/local/bin" afl-as.c -o afl-as -ldl
ln -sf afl-as as
[*] Testing the CC wrapper and instrumentation output...
unset AFL_USE_ASAN AFL_USE_MSAN; AFL_QUIET=1 AFL_INST_RATIO=100 AFL_PATH=. ./afl-gcc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl" -DDOC_PATH="/usr/local/share/doc/afl" -DBIN_PATH="/usr/local/bin" test-instr.c -o test-instr -ldl
echo 0 | ./afl-showmap -m none -q -o .test-instr0 ./test-instr
echo 1 | ./afl-showmap -m none -q -o .test-instr1 ./test-instr
[+] All right, the instrumentation seems to be working!
[+] LLVM users: see llvm_mode/README.llvm for a faster alternative to afl-gcc.
[+] All done! Be sure to review README - it's pretty short and useful.

```

폴더로 이동하여 소스를 컴파일한다.

```

root@kali: /home/kali/afl-2.52b
File Actions Edit View Help

root@kali: /home/kali/afl-2.52b
make install
[*] Checking for the ability to compile x86 code...
[+] Everything seems to be working, ready to compile.
[*] Testing the CC wrapper and instrumentation output...
unset AFL_USE_ASAN AFL_USE_MSAN; AFL_QUIET=1 AFL_INST_RATIO=100 AFL_PATH=. ./afl-gcc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-p
ointer-sign -DAFL_PATH="/usr/local/lib/afl/" -DDOC_PATH="/usr/local/share/doc/afl/" -DBIN_PATH="/usr/local/bin/" test-instr.c -o test-in
str -ldl
echo 0 | ./afl-showmap -m none -q -o .test-instr0 ./test-instr
echo 1 | ./afl-showmap -m none -q -o .test-instr1 ./test-instr
[+] All right, the instrumentation seems to be working!
[+] LLVM users: see llvm_mode/README.llvm for a faster alternative to afl-gcc.
[+] All done! Be sure to review README - it's pretty short and useful.

mkdir -p -m 755 ${DESTDIR}/usr/local/bin ${DESTDIR}/usr/local/lib/afl ${DESTDIR}/usr/local/share/doc/afl ${DESTDIR}/usr/local/share/afl
rm -f ${DESTDIR}/usr/local/bin/afl-plot.sh
install -m 755 afl-gcc afl-fuzz afl-showmap afl-tmin afl-gotcpu afl-analyze afl-plot afl-cmin afl-whatsup ${DESTDIR}/usr/local/bin
rm -f ${DESTDIR}/usr/local/bin/afl-as
if [ -f afl-qemu-trace ]; then install -m 755 afl-qemu-trace ${DESTDIR}/usr/local/bin; fi
if [ -f afl-clang-fast -a -f afl-llvm-pass.so -a -f afl-llvm-rt.o ]; then set -e; install -m 755 afl-clang-fast ${DESTDIR}/usr/local/bin; l
n -sf afl-clang-fast ${DESTDIR}/usr/local/bin/afl-clang-fast++; install -m 755 afl-llvm-pass.so afl-llvm-rt.o ${DESTDIR}/usr/local/lib/afl;
fi
if [ -f afl-llvm-rt-32.o ]; then set -e; install -m 755 afl-llvm-rt-32.o ${DESTDIR}/usr/local/lib/afl; fi
if [ -f afl-llvm-rt-64.o ]; then set -e; install -m 755 afl-llvm-rt-64.o ${DESTDIR}/usr/local/lib/afl; fi
set -e; for i in afl-g++ afl-clang afl-clang++; do ln -sf afl-gcc ${DESTDIR}/usr/local/bin/$i; done
install -m 755 afl-as ${DESTDIR}/usr/local/lib/afl/afl
ln -sf afl-as ${DESTDIR}/usr/local/lib/afl/as
install -m 644 docs/README docs/ChangeLog docs/*.txt ${DESTDIR}/usr/local/share/doc/afl
cp -r testcases/ ${DESTDIR}/usr/local/share/afl
cp -r dictionaries/ ${DESTDIR}/usr/local/share/afl

```

make 를 통해 만들어진 설치파일을 설치한다.

```

root@kali: /home/kali/afl-2.52b
File Actions Edit View Help

#include <stdio.h>
#include <string.h>

int main(void){
    char login[16];
    char password[16];

    printf("login : ");
    scanf("%s",login);
    printf("password : ");
    scanf("%s",password);

    if(strcmp(login,"root") == 0){
        if(strcmp(password,"toor") == 0){
            printf("Success.\n");
            return 0;
        }
    }
    printf("Fail.\n");
    return 1;
}

```

사용자로부터 id, password 를 입력받아 프로그램이 요구하는 값과 일치하면 "Success"를 출력하고, 그렇지 않으면 "Fail"을 출력하는 테스트 파일을 생성한다.

```

root@kali: /home/kali/afl-2.52b/testcase
File Actions Edit View Help

(root@kali)-[/home/kali/afl-2.52b]
# mkdir testcase

(root@kali)-[/home/kali/afl-2.52b]
# cd testcase

(root@kali)-[/home/kali/afl-2.52b/testcase]
# echo -e "a\toor" > test1.txt

(root@kali)-[/home/kali/afl-2.52b/testcase]
# echo -e "root\na" > test2.txt

(root@kali)-[/home/kali/afl-2.52b/testcase]
# echo -e "a\na" > test3.txt

(root@kali)-[/home/kali/afl-2.52b/testcase]
# echo -e "root\toor" > test4.txt

(root@kali)-[/home/kali/afl-2.52b/testcase]
#

```

퍼징에서 사용하기 위해 테스트 케이스(ID 가 틀린 경우, Password 가 틀린 경우, ID, Password 모두 틀린 경우, ID, Password 정확한 경우)를 생성한다.

```

root@kali: /home/kali/afl-2.52b
File Actions Edit View Help

(root@kali)-[/home/kali/afl-2.52b]
# afl-gcc -o test test.c
afl-cc 2.52b by <lcantuf@google.com>
afl-as 2.52b by <lcantuf@google.com>
[+] Instrumented 7 locations (64-bit, non-hardened mode, ratio 100%).

(root@kali)-[/home/kali/afl-2.52b]
# ./test
Login : root
Password : root
Fail.

(root@kali)-[/home/kali/afl-2.52b]
# ./test
Login : root
Password : toor
Success.

```

afl 에서 제공하는 컴파일러를 이용하여 빌드하며, 빌드된 파일이 정상적으로 동작하는 것을 확인한다.


```

root@kali: /home/kali/afl-2.52b
File Actions Edit View Help

(root@kali)-[/home/kali/afl-2.52b]
# echo core > /proc/sys/kernel/core_pattern

(root@kali)-[/home/kali/afl-2.52b]
# mkdir result

(root@kali)-[/home/kali/afl-2.52b]
# afl-fuzz -i testcase/ -o result/ ./test
afl-fuzz 2.52b by <lcantuf@google.com>
[+] You have 4 CPU cores and 2 runnable tasks (utilization: 50%).
[+] Try parallel jobs - see /usr/local/share/doc/afl/parallel_fuzzing.txt.
[+] Checking CPU core loadout ...
[+] Found a free CPU core, binding to #0.
[+] Checking core_pattern ...
[+] Setting up output directories ...
[+] Output directory exists but deemed OK to reuse.
[+] Deleting old session data ...
[+] Output dir cleanup successful.
[+] Scanning 'testcase/' ...
[+] No auto-generated dictionary tokens to reuse.
[+] Creating hard links for all input files ...
[+] Validating target binary ...
[+] Attempting dry run with 'id:000000,orig:test1.txt' ...
[+] Spinning up the fork server ...
[+] All right - fork server is up.
    len = 6, map size = 3, exec speed = 261 us
[+] Attempting dry run with 'id:000001,orig:test2.txt' ...
    len = 7, map size = 4, exec speed = 225 us
[+] Attempting dry run with 'id:000002,orig:test3.txt' ...
    len = 4, map size = 3, exec speed = 745 us
[!] WARNING: No new instrumentation output, test case may be useless.
[+] Attempting dry run with 'id:000003,orig:test4.txt' ...
    len = 9, map size = 4, exec speed = 203 us
[!] WARNING: No new instrumentation output, test case may be useless.
[+] All test cases processed.

```

[afl-fuzz -i testcase/ -o result/ ./test]

afl-fuzz 를 사용하여 테스트 케이스가 저장된 디렉토리 위치를 지정하고 결과가 저장될 디렉토리 위치를 지정한 후 실행할 파일을 지정한다.

```

root@kali: /home/kali/afl-2.52b
File Actions Edit View Help

american fuzzy lop 2.52b (test)

process timing
  run time : 0 days, 0 hrs, 3 min, 12 sec
  last new path : 0 days, 0 hrs, 3 min, 12 sec
  last uniq crash : 0 days, 0 hrs, 3 min, 7 sec
  last uniq hang : none seen yet
cycle progress
  now processing : 0 (0.00%)
  paths timed out : 0 (0.00%)
stage progress
  now trying : splice 11
  stage execs : 31/32 (96.88%)
  total execs : 814k
  exec speed : 4335/sec
fuzzing strategy yields
  bit flips : 0/168, 0/163, 0/153
  byte flips : 0/21, 0/16, 0/6
  arithmetics : 0/1168, 0/125, 0/0
  known ints : 0/114, 0/448, 0/264
  dictionary : 0/0, 0/0, 0/2
  havoc : 4/364k, 0/446k
  trim : 37.04%/5, 0.00%

overall results
  cycles done : 313
  total paths : 5
  uniq crashes : 3
  uniq hangs : 0
map coverage
  map density : 0.00% / 0.01%
  count coverage : 1.00 bits/tuple
findings in depth
  favored paths : 3 (60.00%)
  new edges on : 3 (60.00%)
  total crashes : 16.7k (3 unique)
  total tmoouts : 0 (0 unique)
path geometry
  levels : 2
  pending : 0
  pend fav : 0
  own finds : 1
  imported : n/a
  stability : 100.00%

[cpu000: 49%]

```

퍼징이 실행되고 있는 상태를 보여주며, findings in depth 를 보며 total crashes 를 확인 가능하며, 그 중 unique crashes 가 3 개인 것을 확인 가능하다.


```
root@kali: /home/kali/afl-2.52b/result/crashes
File Actions Edit View Help

(root@kali)~/home/kali/afl-2.52b
# cd result

(root@kali)~/home/kali/afl-2.52b/result
# cd crashes

(root@kali)~/home/kali/afl-2.52b/result/crashes
# ls -al
total 24
drwx----- 2 root root 4096 Jan 12 20:15 .
drwxr-xr-x 5 root root 4096 Jan 12 20:15 ..
-rw----- 1 root root 89 Jan 12 20:15 id:000000,sig:11,src:000000,op:havoc,rep:64
-rw----- 1 root root 81 Jan 12 20:15 id:000001,sig:11,src:000001,op:havoc,rep:128
-rw----- 1 root root 117 Jan 12 20:15 id:000002,sig:11,src:000002,op:havoc,rep:128
-rw----- 1 root root 604 Jan 12 20:15 README.txt

(root@kali)~/home/kali/afl-2.52b/result/crashes
#
```

result/crashes 폴더로 이동하면 크래시가 터진 변조된 파일들을 확인 할 수 있다.

```
root@kali: /home/kali/afl-2.52b/result/crashes
File Actions Edit View Help

(root@kali)~/home/kali/afl-2.52b/result/crashes
# ../../test < ./id:000000,sig:11,src:000000,op:havoc,rep:64
Login : Password : Fail.
zsh: segmentation fault ../../test < ./id:000000,sig:11,src:000000,op:havoc,rep:64

(root@kali)~/home/kali/afl-2.52b/result/crashes
#
```

저장된 파일을 이용하여 해당 crash 를 재현할 수 있으며, 에러 내용 확인이 가능하다.

3.3 ASAN 설치 및 사용법

ASAN 은 AddressSanitizer 의 약자로 네이티브 코드의 메모리 버그 감지를 위한 빠른 컴파일러 기반 도구이다. 스택 및 힙 버퍼 오버플로우/언더플로우, 프리 후 힙 사용, 범위를 벗어난 스택 사용, 더블 프리/와일드 프리 등을 감지한다. 또한 ASAN 을 사용하면 크래시에 대한 정보가 매우 자세하게 나온다.

```

root@kali: /home/kali/afl-2.52b
File Actions Edit View Help

(root@kali)~/home/kali/afl-2.52b
# afl-gcc -o test test.c -fsanitize=address
afl-cc 2.52b by <lcantuf@google.com>
afl-as 2.52b by <lcantuf@google.com>
[+] Instrumented 8 locations (64-bit, ASAN/MSAN mode, ratio 33%).

```

afl-gcc 를 사용하여 다시 빌드하며, 뒤에 ASAN 사용을 위해 -fsanitize=address 을 추가한다.

```

root@kali: /home/kali/afl-2.52b
File Actions Edit View Help

(root@kali)~/home/kali/afl-2.52b
# ./test < ./result/crashes/id:000000.sig:11.src:000000.op:havoc.rep:64

==940368==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffeb34d7360 at pc 0x7ff4cf686632 bp 0x7ffeb34d71d0 sp 0x7ffeb34d6980
WRITE of size 72 at 0x7ffeb34d7360 thread T0
#0 0x7ff4cf686631 in scanf_common ../../src/libsanitizer/sanitizer_common/sanitizer_common_interceptors_format.inc:342
#1 0x7ff4cf6870f7 in __interceptor__isoc99_vscanf ../../src/libsanitizer/sanitizer_common/sanitizer_common_interceptors.inc:1530
#2 0x7ff4cf6871e6 in __interceptor__isoc99_scanf ../../src/libsanitizer/sanitizer_common/sanitizer_common_interceptors.inc:1551
#3 0x55db8b5d8269 in main /home/kali/afl-2.52b/test.c:9
#4 0x7ff4cf48a7ec in __libc_start_main ../csu/libc-start.c:332
#5 0x55db8b5d8509 in _start (/home/kali/afl-2.52b/test+0x1509)

Address 0x7ffeb34d7360 is located in stack of thread T0 at offset 48 in frame
#0 0x55db8b5d81af in main /home/kali/afl-2.52b/test.c:4

This frame has 2 object(s):
[32, 48) 'login' (line 5)
[64, 80) 'password' (line 6) == Memory access at offset 48 partially underflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind mechanism, swapcontext or vfork
(Longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow ../../src/libsanitizer/sanitizer_common/sanitizer_common_interceptors_format.inc:342
in scanf_common
Shadow bytes around the buggy address:
 0x100056692e10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x100056692e20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x100056692e30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x100056692e40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x100056692e50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x100056692e60: 00 00 00 00 00 00 f1 f1 f1 f1 00 00 f2 f2 00 00
0x100056692e70: f3 f3 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x100056692e80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x100056692e90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x100056692ea0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x100056692eb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Freed heap region: fd
Stack left redzone: f1
Stack mid redzone: f2
Stack right redzone: f3
Stack after return: f5

```

ASAN 을 사용하여 빌드한 파일을 이용하여 크래시 재현 시 보다 자세한 내용 확인이 가능하며, 어떤 오류가 발생하였고 발생한 위치 등 확인 가능하다.

4. FirmAE

FirmAE 는 에뮬레이션 및 취약점 분석을 수행하는 완전 자동화된 프레임워크이다. Firmadyne 보다 에뮬레이션 성공률을 크게 높였으며, 또한 대상 펌웨어의 파일 시스템 및 커널 로그를 기반으로 웹 서비스 정보를 유추하는 0-day 검색을 위한 동적 분석 도구를 개발하였다.

4.1 설치 및 사용법

```
$ git clone --recursive https://github.com/pr0v3rbs/FirmAE
```

Github 에 접근해 폴더를 복사한다.

```
$ ./download.sh
```

download.sh 스크립트를 실행한다.

```
$ ./install.sh
```

install.sh 스크립트를 실행한다.

```
$ ./init.sh
```

설치가 끝나면 init.sh 스크립트를 실행한다.

```
$ wget http://download.ipstime.co.kr/online_upgrade/n604bl_kr_9_982.bin
```

펌웨어를 다운받는다.

```
$ sudo ./run.sh -c iptime n604bl_kr_9_982.bin
```

./run.sh -c <브랜드> <펌웨어> 로 명령어가 구성되어있으며, 사용할 펌웨어의 브랜드와 펌웨어 압축파일명을 작성한다.

```
$ sudo ./run.sh -d iptime n604bl_kr_9_982.bin
```

사용자 수준의 기본 디버깅 유틸리티로 에뮬레이트된 펌웨어가 네트워크를 사용할 경우 유용하다.

4.2 동적 디버깅

동적 디버깅이란 파일을 직접 실행시켜서 분석하는 방법으로 디버깅을 통하여 코드 흐름과 메모리 상태등을 살펴보는 방법이다. 또한 디버거를 이용하여 프로그램 내부 구조와 동작원리를 분석할 수 있다.

```

root@kali: /home/kali/FirmAE
File Actions Edit View Help

(root@kali)-[/home/kali/FirmAE]
# ./run.sh -d iptime n604bl kr 9 982.bin
[*] n604bl_kr_9_982.bin emulation start!!!
[*] extract done!!!
[*] get architecture done!!!
[*] n604bl_kr_9_982.bin already succeed emulation!!!

[IID] 3
[MODE] debug
[+] Network reachable on 192.168.0.1!
[+] Web service on 192.168.0.1
[+] Run debug!
Creating TAP device tap3_0 ...
Set 'tap3_0' persistent and owned by uid 1000
Bringing up TAP device ...
Starting emulation of firmware ... 192.168.0.1 true true 9.206096032 9.206096032
[*] firmware - n604bl_kr_9_982
[*] IP - 192.168.0.1
[*] connecting to netcat (192.168.0.1:31337)
[+] netcat connected

FirmAE Debugger

1. connect to socat
2. connect to shell
3. tcpdump
4. run gdbserver
5. file transfer
6. exit
>

```

[`./run.sh -d iptime n604bl_kr_9_982.bin`] 명령어를 실행하여 debug 모드로 실행한다.



192.168.0.1 로 접근 시 iptime 관리자 페이지에 접근이 가능하다.

```

root@kali: /home/kali/FirmAE
File Actions Edit View Help

(root@kali)-[/home/kali/FirmAE]
# binwalk -eM n604bl_kr_9_982.bin

Scan Time:      2022-01-12 23:08:29
Target File:    /home/kali/FirmAE/n604bl_kr_9_982.bin
MD5 Checksum:   77d987b5998c6ad2b2cc236ae1c85e29
Signatures:     411

DECIMAL      HEXADECIMAL    DESCRIPTION
-----
0            0x0            uImage header, header size: 64 bytes, header CRC: 0xC4EC2D86, created: 2016-12-13 09:20:56, image size: 3441
852 bytes, Data Address: 0x80000000, Entry Point: 0x8000C120, data CRC: 0x8DE743AC, OS: Linux, CPU: MIPS, image type: OS Kernel Image, com
pression type: lzma, image name: "n604bl"
64           0x40           LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes, uncompressed size: 4119628 bytes
1344764      0x1484FC       Squashfs filesystem, little endian, version 4.0, compression:xz, size: 2094554 bytes, 1261 inodes, blocksize
: 131072 bytes, created: 2016-12-13 09:20:53

Scan Time:      2022-01-12 23:08:29
Target File:    /home/kali/FirmAE/_n604bl_kr_9_982.bin.extracted/40
MD5 Checksum:   0df3cc9f574ad5ac9680c46c75727062
Signatures:     411

DECIMAL      HEXADECIMAL    DESCRIPTION
-----
782486      0xBF096        PGP RSA encrypted session key - keyid: 801000 202242C RSA Encrypt-Only 1024b
3248220     0x31905C       Linux kernel version 2.6.36
3248376     0x3190F8       CRC32 polynomial table, little endian
3300592     0x325CF0       CRC32 polynomial table, little endian
3327796     0x32C734       SHA256 hash constants, little endian
3328108     0x32C86C       AES S-Box
3328908     0x32C88C       AES Inverse S-Box
3350011     0x331DFB       Neighborly text, "NeighborRequestHandle_ActionHandle"
3350043     0x331E1B       Neighborly text, "NeighborResponseHandleM_EnqueueBcnReq"
3350135     0x331E77       Neighborly text, "NeighborRepsureReq"
3350416     0x331F90       Neighborly text, "NeighborReqActionction"

```

binwalk 툴을 사용하여 대상 펌웨어 파일시스템을 추출한다.

```

root@kali: /home/kali/FirmAE/_n604bl_kr_9_982.bin.extracted/squashfs-root/cgibin
File Actions Edit View Help

(root@kali)-[/home/kali/FirmAE]
# cd _n604bl_kr_9_982.bin.extracted

(root@kali)-[/home/kali/FirmAE/_n604bl_kr_9_982.bin.extracted]
# cd squashfs-root

(root@kali)-[/home/kali/FirmAE/_n604bl_kr_9_982.bin.extracted/squashfs-root]
# cd cgibin

(root@kali)-[/home/.../FirmAE/_n604bl_kr_9_982.bin.extracted/squashfs-root/cgibin]
# ls
captcha.cgi  download.cgi  info.cgi      login_handler.cgi  net_apply.cgi  upgrade.cgi
d.cgi        download_firewall.cgi  login.cgi     login_session.cgi  sys_apply.cgi  wireless_apply.cgi
ddns         download_portforward.cgi  login.cgi     m.cgi              timepro.cgi    wol_apply.cgi

(root@kali)-[/home/.../FirmAE/_n604bl_kr_9_982.bin.extracted/squashfs-root/cgibin]
#

```

추출한 파일시스템 디렉토리로 이동하여 동적 디버깅할 대상을 확인한다.

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment
53	http://192.168.0.1	POST	/sess-bin/login_handler.cgi		✓	200	248	HTML	cgi	E-FM Networks ipTIME ...	
52	http://192.168.0.1	GET	/sess-bin/login_session.cgi			200	5427	HTML	cgi	E-FM Networks ipTIME ...	

Request

```

1 POST /sess-bin/login_handler.cgi HTTP/1.1
2 Host: 192.168.0.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 179
9 Origin: http://192.168.0.1/sess-bin/login_session.cgi
10 Connection: close
11 Referer: http://192.168.0.1/sess-bin/login_session.cgi
12 Upgrade-Insecure-Requests: 1
13
14 init_status=1&captcha_on=1&captcha_file=0168B5931JB156hBed8i6qZ2mWI281gl&username=admin&passwd=1234&default_passwd=Password+is+%27admin%27.+Change+the+password.&captcha_code=lnkma

```

Response

```

1 HTTP/1.0 200 OK
2 Date: Thu, 13 Jan 2022 04:13:05 GMT
3 Server: Httpd/1.0
4 Connection: close
5 Content-type: text/html; charset=utf-8
6
7
8 <html>
  <script>
    parent.parent.location = "/sess-bin/login_session.cgi?noauto=1";
    //session_timeout
  </script>
</html>

```

로그인 시 입력하는 값들에 대해 오버플로우가 발생하는지 확인하기 위해 로그인 시 인자 값을 전송하는 login_handler.cgi 를 공격 대상으로 정하였다.

```

root@kali: /home/kali/FirmAE
File Actions Edit View Help
11 root      SW< [crypto]
12 root      SW< [bioset]
13 root      SW< [kblockd]
14 root      SW< [ata_sff]
15 root      SW< [cfg80211]
16 root      SW [kworker/0:1]
17 root      SW [kswapd0]
18 root      SW [fsnotify_mark]
35 root      SW [scsi_eh_0]
36 root      SW< [scsi_tmf_0]
37 root      SW [scsi_eh_1]
38 root      SW< [scsi_tmf_1]
39 root      SW [kworker/u2:1]
40 root      SW [kworker/u2:2]
41 root      SW [kworker/u2:3]
44 root      SW< [kpsmouse]
45 root      SW< [ipv6_addrconf]
46 root      SW< [deferwq]
47 root      SW< [kworker/0:1H]
52 root      1236 S  /bin/sh
57 root      1684 S  /firmadyne/sh /firmadyne/network.sh
60 root      1676 S  /firmadyne/sh /firmadyne/debug.sh
65 root      1676 S  /firmadyne/sh
66 root      1672 S  /firmadyne/busybox telnetd -p 31338 -l /firmadyne/sh
67 root      1668 S  /firmadyne/busybox sleep 36000
90 root      1992 S  httpd
2018 root    1668 S  /firmadyne/busybox sleep 5
2019 root    1236 R  ps

[+] target pid : 90
[+] gdbserver at 192.168.0.1:1337 attach on 90
[+] run target "remote 192.168.0.1:1337" in host gdb

```

run gdbserver 를 선택하여 타겟 서비스 pid 를 입력한 후 gdb server 연결 주소를 확인한다.

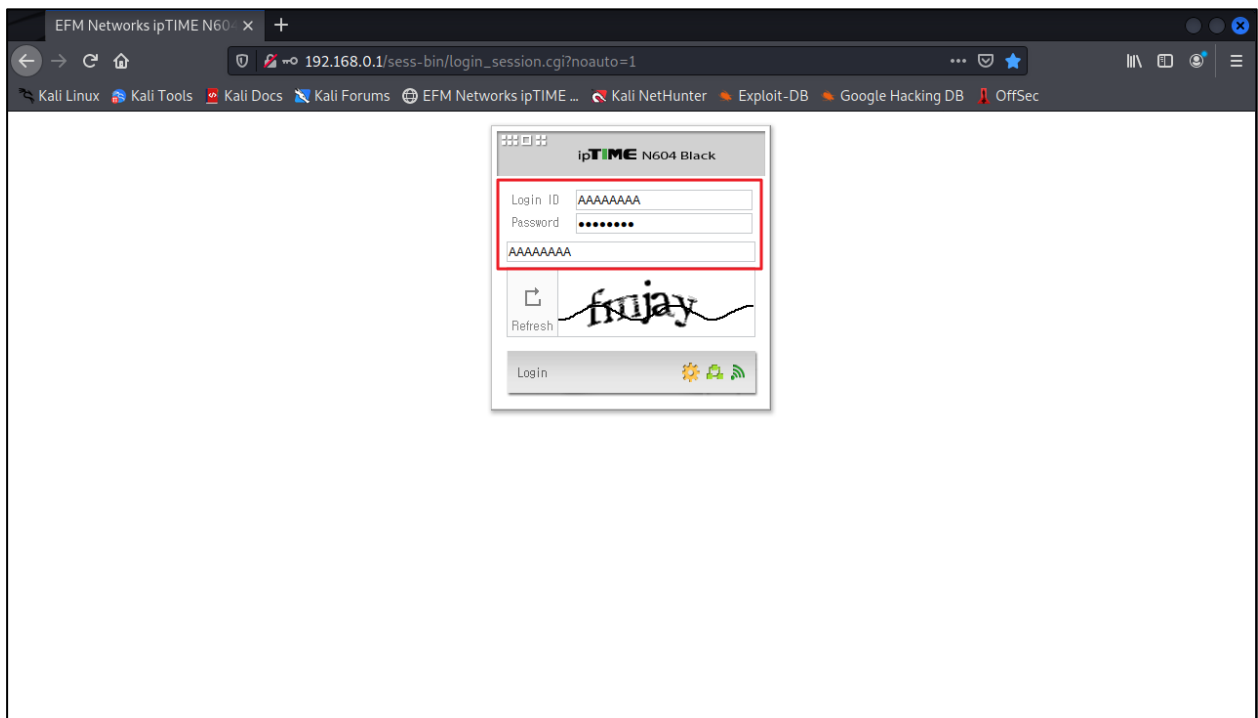

```

root@kali: /home/kali/FirmAE/_n604bl_kr_9_982.bin.extracted/squashfs-root
File Actions Edit View Help

(root@kali)-[/home/kali/FirmAE/_n604bl_kr_9_982.bin.extracted/squashfs-root]
# gdb-multiarch -q ./cgibin/login_handler.cgi
GEF for linux ready, type 'gef' to start, 'gef config' to configure
91 commands loaded for GDB 10.1.90.20210103-git using Python engine 3.9
[*] 5 commands could not be loaded, run 'gef missing' to know why.
GEF for linux ready, type 'gef' to start, 'gef config' to configure
91 commands loaded for GDB 10.1.90.20210103-git using Python engine 3.9
[*] 5 commands could not be loaded, run 'gef missing' to know why.
Reading symbols from ./cgibin/login_handler.cgi...
(No debugging symbols found in ./cgibin/login_handler.cgi)
gef> set sysroot
gef> set follow-fork-mode child
gef> set solib-search-path /home/kali/FirmAE/_n604bl_kr_9_982.bin.extracted/lib/
gef> target remote 192.168.0.1:1337

```

gdb 를 이용하여 대상 cgi 파일을 실행 및 디버깅 설정을 한 후 gdb 서버와 연결한다.



적절한 장소에 breakpoint 를 설정한 후 입력 값을 작성하여 로그인 시도를 한다.

```

root@kali: /home/kali/FirmAE/_n604bl_kr_9_982.bin.extracted/squashfs-root

File Actions Edit View Help
$V1 : 0x7fa4caa5 → 0x007fa400
$A0 : 0x6
$A1 : 0x777e18cf → 0x00000000
$A2 : 0x0
$A3 : 0x0
$T0 : 0x19999999
$T1 : 0x6
$T2 : 0x0
$T3 : 0x0
$T4 : 0x1
$T5 : 0x777e54c0 → "s_api_check_station_exist"
$T6 : 0x0
$T7 : 0xffffffff
$S0 : 0x7fa4cb18 → "192.168.0.1"
$S1 : 0x00433405 → "http://192.168.0.1/sess-bin/login_session.cgi?noau[...]"
$S2 : 0x0043345d → "init_status=16captcha_on=16captcha_file=V3Hqos24B1[ ...]"
$S3 : 0x0
$S4 : 0xa
$S5 : 0x4
$S6 : 0x1
$S7 : 0x0043351a → 0x00000000
$T8 : 0x0
$T9 : 0x7779fd0 → <close+0> lui gp, 0x7
$K0 : 0x0
$K1 : 0x0
$S8 : 0x0
$PC : 0x7787d7e8 → <check_same_subnet_with_local+56> lw gp, 16(sp)
$SP : 0x7fa4ca80 → 0x00000001
$HI : 0x2
$LO : 0x0
$FIR : 0x739300
$RA : 0x7787d7e8 → <check_same_subnet_with_local+56> lw gp, 16(sp)
$GP : 0x777e54c0 → "s_api_check_station_exist"

0x7fa4ca80 +0x0000: 0x00000001 ← $sp
0x7fa4ca84 +0x0004: 0x00000000
0x7fa4ca88 +0x0008: 0x00000000
0x7fa4ca8c +0x000c: 0x00000000
0x7fa4ca90 +0x0010: 0x778f3550 → 0x00000000
0x7fa4ca94 +0x0014: 0x00000000
0x7fa4ca98 +0x0018: "255.255.255.0"
0x7fa4ca9c +0x001c: "255.255.0"

0x7787d7dc <check_same_subnet_with_local+44> addiu a2, sp, 24
0x7787d7e0 <check_same_subnet_with_local+48> jalr t9
0x7787d7e4 <check_same_subnet_with_local+52> addiu a0, a0, 12980
→ 0x7787d7e8 <check_same_subnet_with_local+56> lw gp, 16(sp)
0x7787d7ec <check_same_subnet_with_local+60> move a0, s0
0x7787d7f0 <check_same_subnet_with_local+64> addiu a1, sp, 56
0x7787d7f4 <check_same_subnet_with_local+68> lw t9, -30836(gp)
0x7787d7f8 <check_same_subnet_with_local+72> nop
0x7787d7fc <check_same_subnet_with_local+76> jalr t9

[#0] Id 1, Name: "httpd", stopped 0x7787d7e8 in check_same_subnet_with_local (), reason: SINGLE STEP
[#0] 0x7787d7e8 → check_same_subnet_with_local()
[#1] 0x7787d7e8 → check_same_subnet_with_local()

gef>

```

레지스터에 입력한 값들이 저장된 것을 확인한다.

```

gef> x/s $s2
0x43345d: "init_status=16captcha_on=16captcha_file=V3Hqos24B1Xww4010T5XnV4lBF8c0GLA&username=AAAAAAA&passwd=AAAAAAA&default_passwd
=Password+is+%27admin%27.+Change+the+password.6captcha_code=AAAAAAA"
gef> x/20gx $s2
0x43345d: 0x6174735f74696e69 0x616326313d737574
0x43346d: 0x6e6f5f6168637470 0x637470616326313d
0x43347d: 0x3d656c696665f6168 0x3432736f71483356
0x43348d: 0x314f34777583142 0x6c34566e5835544f
0x43349d: 0x414c473063384642 0x6d616e7265737526
0x4334ad: 0x4141414141413d65 0x7773736170264141
0x4334bd: 0x4141414141413d64 0x7561666564264141
0x4334cd: 0x77737361705f746c 0x6f77737361503d64
0x4334dd: 0x32252b73692b6472 0x32256e696d646137
0x4334ed: 0x676e6168432b2e37 0x61702b6568742b65
gef>

```

입력한 값들이 정상적으로 레지스터에 존재하며, 동적 디버깅이 가능하다.