**Suricata - Bug #3883**

**Runmode Single Memory Leak**

08/17/2020 01:25 PM - Joshua Lumb

### Description

148857ERROR: LeakSanitizer: detected memory leaks

Direct leak of 176 byte(s) in 1 object(s) allocated from:

0x7f2d59623dc6 in calloc (/lib/x86_64-linux-gnu/libasan.so.5+0x10ddc6)
0x561af4dcf19b in SCCallocFunc /home/user/suricata/src/util-mem.c:57
0x561af4c63df7 in ParseAFPConfig /home/user/suricata/src/runmode-af-packet.c:131
0x561af4c683f8 in RunModeIdsAFPSingle /home/user/suricata/src/runmodes.c:382
0x561af4c75f56 in RunModeDispatch /home/user/suricata/src/runmodes.c:374
0x7f2d58e700b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)

**SUMMARY:** AddressSanitizer: 176 byte(s) leaked in 1 allocation(s).

An explanation:

During startup suricata parses config settings for af-packet threads in the function ParseAFPConfig and stores those settings in a memory-allocated AFPIfaceConfig struct. For efficiency this is only done once - but since the thread initialization function will need the AFPIfaceConfig once for each receiving thread the AFPIfaceConfig tracks the number of threads which will need to refer to it in AFPIfaceConfig->ref.

AFPIfaceConfig->ref is set based on the config file’s specified number of threads, which itself is usually determined here:

```
if (StringParseInt32(&aconf->threads, 10, 0, (const char *)threadsstr) < 0)
```

Which eventually ends up setting aconf->ref here:

```
SC_ATOMIC_RESET(aconf->ref);
(void) SC_ATOMIC_ADD(aconf->ref, aconf->threads);
```

If all is going well AFPIfaceConfig aconf is freed by the thread initialization function ReceiveAFPThreadInit's sub-function AFPDerefConfig using the following mechanism:

```
if (SC_ATOMIC_SUB(pfp->ref, 1) == 1) {
SCFree(pfp);
```

This means that as each thread finishes initialization it will decrement the counter until the last thread finally decrements it to 0, at which time AFPIfaceConfig aconf is freed.

The problem comes into play with --runmode single which overrides the number of threads to be created. There is no consideration for this currently in the assignment of aconf->ref so you can end up with only one thread calling AFPDerefConfig with aconf->ref > 1, meaning it is never freed.

---

**History**

#1 - 08/25/2020 08:37 AM - Victor Julien

- Status changed from New to In Review
- Assignee set to Joshua Lumb
- Target version set to 6.0.0rc1
https://github.com/OISF/suricata/pull/5303

#2 - 09/04/2020 05:22 AM - Victor Julien
- Status changed from In Review to Closed

https://github.com/OISF/suricata/pull/5303