Suricata - Bug #3041

snmp parsing error message

06/12/2019 12:40 PM - Eric Leblond

| Status:   | New                      |
| Priority: | Normal                   |
| Assignee: | Pierre Chifflier         |
| Category: |                         |
| Target version: | TBD                    |
| Affected Versions: | 5.0rc1            |
| Difficulty: | medium                  |

**Description**

Wit latest master (b5f3e03209922f1029b76a1a3570a3aca91659f5) on some live traffic, I'm seeing a regular message like this one:

```
[80425] 10/6/2019 -- 01:37:04 - (snmp.rs:154) <Info> (<rust>) -- parse_snmp_v2 failed: Err(Error(Code([4, 7, ..., 4, 2, 1, 4], Tag)))
[80409] 10/6/2019 -- 01:37:06 - (snmp.rs:154) <Info> (<rust>) -- parse_snmp_v1 failed: Err(Error(Code([4, 10, 11..., 5, 0], Tag)))
```

I did try to capture pcap and replay it on same branch but it was without success.

The error may be benign but we should at least have it only in debug.

**History**

**#1 - 06/14/2019 08:03 PM - Andreas Herz**
- Assignee set to OISF Dev
- Target version set to TBD

**#2 - 06/17/2019 07:38 AM - Peter Manev**

I also got this on a pcap (but cant share the pcap itself/partly due to size too)

```
[23018] 17/6/2019 -- 02:17:17 - (util-checksum.c:89) <Info> (ChecksumAutoModeCheck) -- No packets with invalid checksum, assuming checksum offloading is NOT used
[23024] 17/6/2019 -- 02:33:17 - (snmp.rs:154) <Info> (<rust>) -- parse_snmp_v2 failed: Err(Error(Code([4, 7, 12, 1, Tag])))
```

Using

```
/opt/suricagotagit/bin/suricata --build-info
This is Suricata version 5.0.0-dev (rev b5f3e0320)
Features: PCAP_SET_BUFF AF_PACKET HAVE_PACKET_FANOUT LIBCAP_NG LIBNRTL1.1 HAVE_HTP_URI_NORMALIZE_HOOK PCRE_JIT HAVE_NSS HAVE LUA HAVE LUAJIT HAVE_LIBJANSSON PROFILING TLS MAGIC RUST
SIMD support: SSE_4_2 SSE_4_1 SSE_3
Atomic intrinsics: 1 2 4 8 16 byte(s)
64-bits, Little-endian architecture
GCC version 7.4.0, C version 199901
compiled with _FORTIFY_SOURCE=2
L1 cache line size (CLS)=64
thread local storage method: __thread
compiled with LibHTP v0.5.30, linked against LibHTP v0.5.30
```
Suricata Configuration:

AF_PACKET support: yes
eBPF support: no
XDP support: no
PF_RING support: no
NFQueue support: no
NFLOG support: no
IPFW support: no
Netmap support: no
DAG enabled: no
Napatech enabled: no
WinDivert enabled: no

Unix socket enabled: yes
Detection enabled: yes

Libmagic support: yes
libnss support: yes
libnspr support: yes
libjansson support: yes
liblzma support: no
hiredis support: no
hiredis async with libevent: no
Prelude support: no
PCRE jit: yes
Lua support: yes, through luajit
libluajit: yes
libgeoip: yes
Non-bundled htp: no
Old barnyard2 support: no
Hyperscan support: yes
Libnet support: yes
liblz4 support: yes

Rust support: yes
Rust strict mode: no
Rust debug mode: no
Rust compiler: rustc 1.32.0
Rust cargo: cargo 1.32.0

Python support: yes
Python path: /usr/bin/python3
Python version: Python 3.6.7
Python distutils yes
Python yaml yes
Install suricatactl: yes
Install suricatasc: yes
Install suricata-update: not bundled

Profiling enabled: yes
Profiling locks enabled: no

Development settings:
Coccinelle / spatch: yes
Unit tests enabled: no
Debug output enabled: no
Debug validation enabled: no

Generic build parameters:
Installation prefix: /opt/suricatagit
Configuration directory: /opt/suricatagit/etc/suricata/
Log directory: /opt/suricatagit/var/log/suricata/

--prefix /opt/suricatagit
--sysconfdir /opt/suricatagit/etc
--localstatedir /opt/suricatagit/var
--datarootdir /opt/suricatagit/share

Host: x86_64-pc-linux-gnu
Compiler: gcc (exec name) / gcc (real)
GCC Protect enabled: no
GCC march native enabled: yes
GCC Profile enabled: no
Position Independent Executable enabled: no
CFLAGS -g -O2 -March=native -I${srcdir}/../rust/gen/c-headers
This one is hard to guess :/ The provided test pcap did not result in any error/warning here.

My current supposition is that there is a mismatch, and that different SNMP versions are seen in the same UDP connection. That is what the standard requests, and the current parser is a bit (over)strict with that (it checks that SNMP requests/answers are all using the same version).

I propose several changes:
- reduce verbosity (it should be Debug, and there is an additional good reason because it contains the SNMP community string)
- change restriction on version: change the parse to correctly parse any version, and maybe just add an event if versions mismatch

I think it would be good to set events. If you can set the version mismatch separately from other issues that might be in play then we can see which ones trigger in the real world traffic.

What is the status of this ticket?