Suricata - Bug #3771

Extreme performance degradation when doing IP-only rules with flow-keyword

06/15/2020 03:32 PM - Antti Tönkyrä

Status: New
Priority: Normal
Assignee:
Category:
Target version:
Affected Versions: 5.0.3
Difficulty:
Effort:
Label:

Description

I did a brief test and found out an issue when doing large sets of IP-only rules.

Following IP-only rules seem to cause a problem while same rules without flow-keyword are fine.

```
alert ip any any -> 10.0.0.0 any (msg: "test 1"; flow: stateless; sid: 1;)
alert ip any any -> 10.0.0.1 any (msg: "test 2"; flow: stateless; sid: 2;)
alert ip any any -> 10.0.0.2 any (msg: "test 3"; flow: stateless; sid: 3;)
```

Below is a test run with attached pcap and rules files. I'd expect the flow.rules would perform a bit worse than plain.rules but it turns out flow.rules takes a magnitude longer.

```
root@telakka:/usr/local/etc/suricata/prof# time suricata -r flowtest.pcap --runmode single -S flow.rules
15/6/2020 -- 15:15:13 - <Notice> - This is Suricata version 5.0.3 RELEASE running in USER mode
15/6/2020 -- 15:15:15 - <Notice> - all 1 packet processing threads, 2 management threads initialized, engine started.
15/6/2020 -- 15:15:26 - <Notice> - Signal Received. Stopping engine.
15/6/2020 -- 15:15:26 - <Notice> - Pcap-file module read 1 files, 2000 packets, 1709602 bytes
real 0m13.543s
user 0m13.493s
sys 0m0.075s
```

```
root@telakka:/usr/local/etc/suricata/prof# time suricata -r flowtest.pcap --runmode single -S plain.rules
15/6/2020 -- 15:15:34 - <Notice> - This is Suricata version 5.0.3 RELEASE running in USER mode
15/6/2020 -- 15:15:36 - <Notice> - all 1 packet processing threads, 2 management threads initialized, engine started.
15/6/2020 -- 15:15:36 - <Notice> - Signal Received. Stopping engine.
15/6/2020 -- 15:15:36 - <Notice> - Pcap-file module read 1 files, 2000 packets, 1709602 bytes
real 0m1.581s
user 0m1.301s
sys 0m0.277s
```

I have attached relevant pcap and rule files. Extract the rules and pcap into any directory and run:

```
time suricata -r flowtest.pcap --runmode single -S flow.rules
time suricata -r flowtest.pcap --runmode single -S plain.rules
```

History

#1 - 06/15/2020 09:10 PM - Peter Manev

I can confirm similar results with latest git

time /opt/suritest/bin/suricata -S flow.rules -l logs/ -k none -r flowtest.pcap --runmode=single
[1013645] 15/6/2020 -- 23:02:00 - (suricata.c:1062) <Notice> [LogVersion] -- This is Suricata version 6.0.0-de
what I also found interesting is that if mpm: auto is enabled, it takes even bigger perf hit

```
1376  prefILTER:
1377  # default prefiltering setting. "mpm" only creates MPM/fast_pattern
1378  # engines. "auto" also sets up prefilter engines for other keywords.
1379  # Use --list-keywords=all to see which keywords support prefiltering.
1380  default: auto #mpm
```

```
time /opt/suritest/bin/suricata -S flow.rules -l logs/ -k none -r flowtest.pcap --runmode=single
15/6/2020 -- 23:05:21 - (suricata.c:1062) <Notice> (LogVersion) -- This is Suricata version 6.0.0-dev
v (79681bf65 2020-06-09) running in USER mode
[1013904] 15/6/2020 -- 23:05:24 - (tm-threads.c:1887) <Notice> (TmThreadWaitOnThreadInit) -- all 1 packet processing threads, 4 management threads initialized, engine started.

real    0m15.879s
user    0m15.859s
sys     0m0.144s
```

Using default configs and

```
/opt/suritest/bin/suricata --build-info
This is Suricata version 6.0.0-dev (79681bf65 2020-06-09)
Features: PCAP_SET_BUFF AF_PACKET HAVE_PACKET_FANOUT LIBCAP_NG LIBNHTI.1 HAVE_HTP_URI_NORMALIZE_HOOK PCRE_JIT HAVE_NSS HAVE_LUA HAVE LUAJIT HAVE_LIBJANSSON TLS TLS_C11 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 9.3.0, C version 201112 compiled with _FORTIFY_SOURCE=0
L1 cache line size (CLSIZE)=64
thread local storage method: _Thread_local
compiled with LibHTP v0.5.33, linked against LibHTP v0.5.33
```
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
hiredis support: no
hiredis async with libevent: no
Prelude support: no
PCRE jit: yes
LUA support: yes, through luajit
libluajit: yes
GeoIP2 support: yes
Non-bundled htp: no
Old barnyard2 support: yes
Hyperscan support: yes
Libnet support: yes
liblz4 support: yes

Rust support: yes
Rust strict mode: no
Rust compiler path: /home/pevma/.cargo/bin/rustc
Rust compiler version: rustc 1.44.0 (49cae5576 2020-06-01)
Cargo path: /home/pevma/.cargo/bin/cargo
Cargo version: cargo 1.44.0 (05d080f0a 2020-05-06)
Cargo vendor: yes

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

Profiling enabled: no
Profiling locks enabled: no

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

Generic build parameters:
Installation prefix: /opt/suritest
Configuration directory: /opt/suritest/etc/suricata/
Log directory: /opt/suritest/var/log/suricata/
--prefix /opt/suritest
--sysconfdir /opt/suritest/etc
--localstatedir /opt/suritest/var
--datarootdir /opt/suritest/share

Host: x86_64-pc-linux-gnu
Compiler: gcc (exec name) / g++ (real)
GCC Protect enabled: no
GCC march native enabled: yes
GCC Profile enabled: no
Position Independent Executable enabled: no
CFLAGS -g -O2 -std=cl1 -march=native -I${srcdir}/../rust/gen
PCAP_CFLAGS -I/usr/include
SECCFLAGS

Profiling enabled: no
Profiling locks enabled: no

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

Generic build parameters:
Installation prefix: /opt/suritest
Configuration directory: /opt/suritest/etc/suricata/
Log directory: /opt/suritest/var/log/suricata/
--prefix /opt/suritest
--sysconfdir /opt/suritest/etc
--localstatedir /opt/suritest/var
--datarootdir /opt/suritest/share

Host: x86_64-pc-linux-gnu
Compiler: gcc (exec name) / g++ (real)
GCC Protect enabled: no
GCC march native enabled: yes
GCC Profile enabled: no
Position Independent Executable enabled: no
CFLAGS -g -O2 -std=cl1 -march=native -I${srcdir}/../rust/gen
PCAP_CFLAGS -I/usr/include
SECCFLAGS

07/16/2022
My observations (which can be false, please double-check; I'm not really an expert on detect code :)

With "almost IP-only rules" suricata ends up in a situation where you have $O(n)$ complexity iteration at DetectRulePacketRules for every packet and $n$ being number of rules. The fact that mpm-auto increases the load is because in prefilter stage the flow matches are done first time and then second time in DetectRulePacketRules.

I'm not sure how one would go about fixing this. Should flow prefilter check for IP addresses like IPonly filter does or should there be a separate keyword which adds IPonly-like prefilter stage?

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow.rules</td>
<td>8.76 MB</td>
<td>06/15/2020</td>
<td>Antti Tönkyrä</td>
</tr>
<tr>
<td>plain.rules</td>
<td>7.14 MB</td>
<td>06/15/2020</td>
<td>Antti Tönkyrä</td>
</tr>
<tr>
<td>flowtest.pcap</td>
<td>1.66 MB</td>
<td>06/15/2020</td>
<td>Antti Tönkyrä</td>
</tr>
</tbody>
</table>