Suricata - Feature #4226
bsize: apply as depth to patterns
12/18/2020 04:56 AM - Brandon Murphy

Status: Assigned
Priority: Normal
Assignee: Jeff Lucovsky
Category: Target version: 7.0rc1
Effort: Label: Needs backport

Description
When reviewing rule profiling output of comparing the speed of bsize:x; checks to using depth:x; isdataat:!1,relative; in effort to achieve an "exact match" of a buffer, it has been observed that bsize variants of rules often consume considerably more ticks than alternative methods.

Tested on 7.0.0-dev (372fc2673 2020-12-11) with the default suricata.yaml, with minor adjustments for non-json output of the rule profiling output.

command used for testing

```
suricata -c suricata.yaml -S test.rules -r a564fbcf-e41d-494c-9ad8-7e44e22a03d9.pcap -l /tmp/perf_test/perf_1/
```

cpcap from https://app.any.run/tasks/a564fbcf-e41d-494c-9ad8-7e44e22a03d9/

test signatures

```
alert dns any any -> any any (msg:"bsize test"; dns.query; content:"yundol0727.kro.kr"; bsize:17; sid:1;)
alert dns any any -> any any (msg:"depth and isdataat test"; dns.query; content:"yundol0727.kro.kr"; depth:17; isdataat:!1,relative; sid:2;)
alert dns any any -> any any (msg:"depth and endswith test"; dns.query; content:"yundol0727.kro.kr"; depth:17; endswith; sid:3;)
```

It's been observed on other buffers as well, just using dns.query as a quick example.

I ran the above signatures through on the same pcap 3 times and have recorded the output of the

First Run

```
Date: 12/18/2020 -- 04:12:47. Sorted by: ticks.
```

<table>
<thead>
<tr>
<th>Num</th>
<th>Rule</th>
<th>Gid</th>
<th>Rev</th>
<th>Ticks</th>
<th>%</th>
<th>Checks</th>
<th>Matches</th>
<th>Max Ticks</th>
<th>Avg Ticks</th>
<th>Avg Match</th>
<th>Avg No Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>45045</td>
<td>66.10</td>
<td>1</td>
<td>1</td>
<td>45045</td>
<td>45045</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>12123</td>
<td>17.79</td>
<td>1</td>
<td>1</td>
<td>12123</td>
<td>12123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>10983</td>
<td>16.12</td>
<td>1</td>
<td>1</td>
<td>10983</td>
<td>10983</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>15045.00</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>12123.00</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>10983.00</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Second Run
### Third Run

<table>
<thead>
<tr>
<th>Num</th>
<th>Rule</th>
<th>Gid</th>
<th>Rev</th>
<th>Ticks</th>
<th>%</th>
<th>Checks</th>
<th>Matches</th>
<th>Max Ticks</th>
<th>Avg Ticks</th>
<th>Avg Match</th>
<th>Avg No Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>26637</td>
<td>53.87</td>
<td>1</td>
<td>1</td>
<td>26637</td>
<td>26637</td>
<td>26637</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>11772</td>
<td>23.81</td>
<td>1</td>
<td>1</td>
<td>11772</td>
<td>11772</td>
<td>11772</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>11040</td>
<td>22.33</td>
<td>1</td>
<td>1</td>
<td>11040</td>
<td>11040</td>
<td>11040</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Build Info**

uricata --build-info
This is Suricata version 7.0.0-dev (372fc2673 2020-12-11)
Features: PCAP_SET_BUFF AF_PACKET HAVE_PACKET_FANOUT LIBCAP_NG LIBNET1.1 HAVE_HTP_URI_NORMALIZE_HOST OK PCRE_JIT HAVE_NSS HAVE LUA HAVE LUAJIT HAVE_LIBJANNSON PROFILING 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 7.5.0, C version 201112
compiled with _FORTIFY_SOURCE=2
L1 cache line size (CLS)=64
thread local storage method: _Thread_local
compiled with LibHTP v0.5.36, linked against LibHTP v0.5.36

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
**hiredis support:** no  
**hiredis async with libevent:** no  
**Prelude support:** no  
**PCRE jit:** yes  
**LUA support:** yes, through luajit  
**libluajit:** yes  
**GeoIP2 support:** no  
**Non-bundled htp:** no  
**Hyperscan support:** no  
**Libnet support:** yes  
**liblz4 support:** yes  

**Rust support:** yes  
**Rust strict mode:** no  
**Rust compiler path:** /usr/bin/rustc  
**Rust compiler version:** rustc 1.43.0  
**Cargo path:** /usr/bin/cargo  
**Cargo version:** cargo 1.43.0  
**Cargo vendor:** yes  

**Python support:** no  
**Python path:** not set  
**Python distutils**  
**Python yaml**  

**Install suricatactl:** requires python  
**Install suricatasc:** requires python  
**Install suricata-update:** not bundled  

**Profiling enabled:** yes  
**Profiling locks enabled:** no  

**Plugin support (experimental):** yes  

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

**Generic build parameters:**  
**Installation prefix:** /usr/local  
**Configuration directory:** /usr/local/etc/suricata/  
**Log directory:** /usr/local/var/log/suricata/  

```
--prefix /usr/local  
--sysconfdir /usr/local/etc  
--localstatedir /usr/local/var  
--datarootdir /usr/local/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**  
**PCAP_CFLAGS**  
**SECCFLAGS**

---

**History**  
**#1 - 03/04/2022 06:07 AM - Victor Julien**  
- Tracker changed from Bug to Feature  
- Subject changed from bsize is considerably slower than depth:x; isdataat:!1,relative to bsize: apply as depth to patterns  
- Status changed from New to Assigned
If bsize setting is the exact length of a pattern, apply startwith/endswith logic. Otherwise, apply it as depth. For dsize (and urilen?) we already do this IIRC.

I think this could be backported as well. Technically its not a bug, but it would be a virtually “free” optimization that should be low risk.

Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>a564fbcf-e41d-494c-9ad8-7e44e22a03d9.pcap</td>
<td>12.1 KB</td>
<td>12/18/2020</td>
<td>Brandon Murphy</td>
</tr>
</tbody>
</table>