### Description

Currently protocol detection is a mix of port independent pattern based matching and port depended 'probing parsers'. This has served reasonably well, but also has serious limitations. Protocols that are similar in structure (e.g. smtp, ftp and irc) are not well supported.

The goal of the improvement is to fix these limitations.

### Related issues:

- Related to Feature #273: IRC protocol detection support
  - New
- Related to Feature #511: Port independent protocol identification
  - New
- Related to Task #2685: SuriCon 2018 brainstorm
  - Assigned
- Related to Bug #2978: IRC traffic parsed by FTP
  - New
- Related to Feature #2713: protocol detection w/o protocol parsing
  - Feedback
- Related to Task #3392: Tracking: protocol detection evasions
  - New
- Related to Feature #1125: smtp: improve protocol detection
  - In Review

### History

1. **12/21/2018 12:07 PM - Victor Julien**  
   - Related to Feature #273: IRC protocol detection support added

2. **12/21/2018 12:07 PM - Victor Julien**  
   - Related to Feature #511: Port independent protocol identification added

3. **12/21/2018 12:07 PM - Victor Julien**  
   - Related to Feature #1125: smtp: improve protocol detection added

4. **12/21/2018 12:07 PM - Victor Julien**  
   - Related to Task #2685: SuriCon 2018 brainstorm added

5. **02/18/2019 10:49 AM - Victor Julien**  
   - Related to Bug #2393: One way TLS traffic not properly identified added

6. **03/11/2019 01:52 PM - Victor Julien**  
   - Status changed from New to Assigned
   - Assignee set to Victor Julien

7. **05/10/2019 07:59 AM - Victor Julien**  
   - Related to Bug #2978: IRC traffic parsed by FTP added

8. **06/15/2019 09:52 PM - Andreas Herz**  
   - Target version set to TBD

9. **06/20/2019 07:14 PM - Victor Julien**
We want to support detection
- unidirectional (for both client and server sides)
- midstream
- port independent
- similar FTP-IRC-SMTP
- performant

What about evasion techniques?
Such as my evil FTP client connects to a FTP server, but then first sends a EHLO request (rejected by the server) and then speaks regular FTP

Correct. It should be robust against evasions too. One way we do this (for TCP) is to check if client and server protos match. If not, an event is set.

While port independent is the goal, it might not always be achievable. I could imagine that in some ambiguous cases, like SMTP/FTP, we would ultimately fall back to ports or at least assign some weight to the ports.

So, I see multiple things to do:

- There is clearly the server and client protos to match (and not stop on first detection as is done now calling TCPProtoDetect)
- Support when multiple protocols have patterns matching (such as USER for FTP and IRC)
- Check completeness for the current protocols in terms of patterns, probes, both client and server side
- Add a configurable behavior to run all probing parsers on unrecognized traffic whatever the port (and then add lots of protocols in terms of recognition only)

There is clearly the server and client protos to match (and not stop on first detection as is done now calling TCPProtoDetect)

In facts, it seems to be done already for TCP (whatever the comment says)
But it seems not done over UDP

- Related to deleted (Bug #2393: One way TLS traffic not properly identified)

About completeness, here are the protocols referenced in app-layer-protos.h but not implemented:
- IMAP
- MSN
- JABBER
- IRC

Should we implement at least detection for them?

And here are the other partially (ie TO_CLIENT) missing protocol detections (and current pull requests):
SMTP : https://github.com/OISF/suricata/pull/3832
DNS : https://github.com/OISF/suricata/pull/4058
MODBUS : https://github.com/OISF/suricata/pull/4082
DN3P : https://github.com/OISF/suricata/pull/4082
TFTP : https://github.com/OISF/suricata/pull/4082

For all the other protocols and directions, we have have at least one mechanism (whose relevance I did not check here):
HTTP,TOSERVER,PM
HTTP,TOCLIENT,PM
FTP,TOSERVER,PM
FTP,TOCLIENT,PM
FTPDATA,FLOW,EXPECTATION
SMTP,TOSERVER,PM
TLS,TOSERVER,PM,PP
TLS,TOCLIENT,PM
SSH,TOSERVER,PM
SSH,TOCLIENT,PM
SMB, TOSERVER, PM+PP  
SMB, TOCLIENT, PM+PP  
DCERP, TOSERVER, PM+PP+hack?  
DCERP, TOCLIENT, PM+hack?  
DNS, TOSERVER, PP  
MODBUS, TOSERVER, PP  
ENIP, TOSERVER, PP  
ENIP, TOCLIENT, PP  
DNP3, TOSERVER, PP  
NFS, TOSERVER, PP  
NFS, TOCLIENT, PP  
TFTP, TOSERVER, PP  
DHCP, TOSERVER, PP  
DHCP, TOCLIENT, PP  
IKEV2, TOSERVER, PP  
IKEV2, TOCLIENT, PP  
KRBS, TOSERVER, PP  
KRBS, TOCLIENT, PP  
NTP, TOSERVER, PP  
NTP, TOCLIENT, PP

#16 - 09/02/2019 01:03 PM - Philippe Antoine

About TCP in the case of client and server protocol mismatch, the one taken into consideration is the first one to have sent data...
Here is a problematic case:
https://github.com/OISF/suricata-verify/pull/117

I think that, in the case of a protocol mismatch between client and server, we should parse further to confirm which is the protocol...
Any ideas on this?

#17 - 09/02/2019 02:36 PM - Philippe Antoine

To sum up a bit my point of view, we need:
- completeness: what is wished? (ie should we add IRC, MSN and such?)
- match client and server protocols: for UDP https://github.com/OISF/suricata/pull/4058
- match client and server protocols: do not always trust the server in case of mismatch cf https://github.com/OISF/suricata-verify/pull/117
- multiple protocol recognition: discussion in https://github.com/OISF/suricata/pull/4152

#18 - 11/01/2019 03:08 PM - Philippe Antoine
- Related to Feature #2713: protocol detection w/o protocol parsing added

#19 - 12/10/2019 09:58 AM - Victor Julien
- Related to Task #3392: Tracking: protocol detection evasions added

#20 - 02/18/2020 12:50 PM - Philippe Antoine
- Status changed from Assigned to In Review

Multiple parallel PRs:
https://github.com/OISF/suricata/pull/4058
https://github.com/OISF/suricata/pull/4152
https://github.com/OISF/suricata/pull/4217

#21 - 02/18/2020 12:52 PM - Philippe Antoine
- Related to deleted (Feature #1125: smtp: improve protocol detection)

#22 - 02/18/2020 12:53 PM - Philippe Antoine
- Blocks Feature #1125: smtp: improve protocol detection added

#23 - 04/15/2020 08:20 AM - Philippe Antoine
- Target version changed from TBD to 6.0.0beta1

#24 - 07/28/2020 09:24 AM - Philippe Antoine
- File sshsmb.pcap added

Here is a sample pcap with both SSH and SMB patterns
#25 - 08/06/2020 07:26 AM - Victor Julien
- Target version changed from 6.0.0beta1 to 6.0.0rc1

#26 - 09/04/2020 07:21 AM - Philippe Antoine
- Target version changed from 6.0.0rc1 to 7.0.0-beta1

#27 - 01/23/2021 01:41 PM - Philippe Antoine
Merged https://github.com/OISF/suricata/pull/5773: debug validation that we cannot have multiple patterns matching at once

#28 - 05/24/2022 06:48 AM - Philippe Antoine
Now that https://github.com/OISF/suricata/pull/7320 got merged, this ticket requires more discussion about what needs to be done
- Code cleaning ?
- match client and server protocols : do not always trust the server in case of mismatch (retry multiple times) ?
- completeness (detect more protocols) ?

#29 - 07/06/2022 12:38 PM - Philippe Antoine
- Blocks deleted (Feature #1125: smtp: improve protocol detection)

#30 - 07/06/2022 12:38 PM - Philippe Antoine
- Related to Feature #1125: smtp: improve protocol detection added

#31 - 07/06/2022 12:39 PM - Philippe Antoine
1125 is next

#32 - 09/12/2022 02:11 PM - Philippe Antoine
- Target version changed from 7.0.0-beta1 to TBD

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>sshsmb.pcap</td>
<td>1.84 KB</td>
<td>07/28/2020</td>
</tr>
</tbody>
</table>

Philippe Antoine