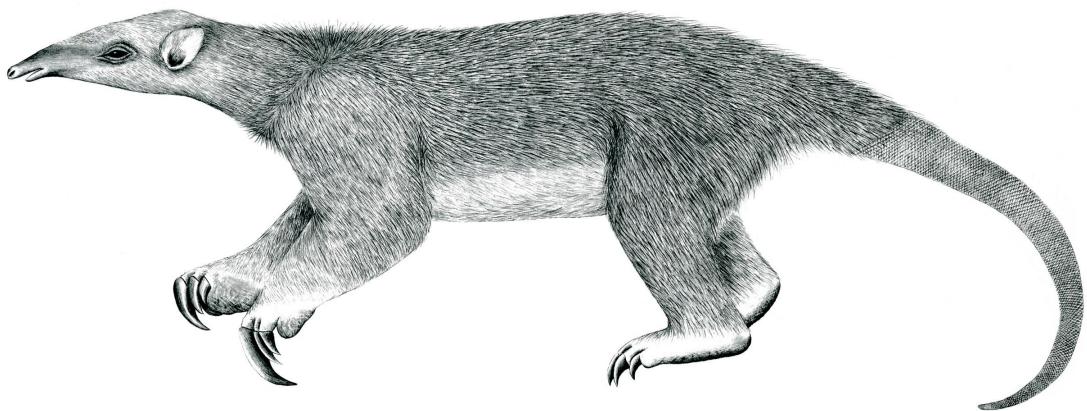

Tranalyzer2

Version 0.9.3 (Cobra)



Flow based forensic and network troubleshooting traffic analyzer



Tranalyzer Development Team

Contents

1	Introduction	1
1.1	Getting Tranalyzer	1
1.2	Dependencies	1
1.3	Compilation	2
1.4	Installation	3
1.5	Getting Started	3
1.6	Getting Help	3
2	Tranalyzer2	5
2.1	Supported Link-Layer Header Types	5
2.2	Enabling/Disabling Plugins	5
2.3	Man Page	7
2.4	Invoking Tranalyzer	7
2.5	hashTable.h	13
2.6	ioBuffer.h	14
2.7	loadPlugins.h	14
2.8	main.h	14
2.9	networkHeaders.h	16
2.10	proto/capwap.h	16
2.11	proto/ethertype.h	16
2.12	proto/linktype.h	16
2.13	proto/lwapp.h	17
2.14	packetCapture.h	17
2.15	tranalyzer.h	17
2.16	bin2txt.h	28
2.17	gz2txt.h	28
2.18	outputBuffer.h	28
2.19	rbTree.h	29
2.20	subnetHL.h	29
2.21	t2log.h	29
2.22	Tranalyzer2 Output	29
2.23	Final Report	30
2.24	Monitoring Modes During Runtime	33
2.25	Cancellation of the Sniffing Process	39
3	arpDecode	40
3.1	Description	40
3.2	Dependencies	40
3.3	Configuration Flags	40
3.4	Flow File Output	40
3.5	Packet File Output	42
3.6	Monitoring Output	42
3.7	Plugin Report Output	42

4 basicFlow	43
4.1 Description	43
4.2 Configuration Flags	43
4.3 Flow File Output	44
4.4 Packet File Output	54
4.5 Post-Processing	55
5 basicStats	56
5.1 Description	56
5.2 Configuration Flags	56
5.3 Flow File Output	57
5.4 Packet File Output	58
5.5 Plugin Report Output	58
6 bayesClassifier	59
6.1 Description	59
6.2 Dependencies	59
6.3 Configuration Flags	59
6.4 Flow File Output	59
6.5 Known Bugs and Limitations	60
7 bgpDecode	61
7.1 Description	61
7.2 Dependencies	61
7.3 Configuration Flags	61
7.4 Flow File Output	62
7.5 Additional Output	67
7.6 Plugin Report Output	68
7.7 Post-Processing	68
7.8 Anomalies	72
7.9 Examples	72
7.10 References	73
8 binSink	74
8.1 Description	74
8.2 Dependencies	74
8.3 Configuration Flags	74
8.4 Post-Processing	75
8.5 t2b2t	75
8.6 Custom File Output	75
9 bitForensic	76
9.1 Description	76
9.2 Dependencies	76
9.3 Configuration Flags	76
9.4 Flow File Output	77
9.5 Packet File Output	77
9.6 Plugin Report Output	77

10 cdpDecode	78
10.1 Description	78
10.2 Dependencies	78
10.3 Configuration Flags	78
10.4 Flow File Output	78
10.5 Packet File Output	80
10.6 Monitoring Output	81
10.7 Plugin Report Output	81
11 centrality	82
11.1 Description	82
11.2 Dependencies	82
11.3 Eigenvector centrality	82
11.4 Configuration Flags	82
11.5 Flow File Output	83
11.6 Centrality File Output	83
11.7 Matrix File Output	83
11.8 TODO	83
12 clickhouseSink	84
12.1 Description	84
12.2 Dependencies	84
12.3 Configuration Flags	85
12.4 Example	86
13 connStat	88
13.1 Description	88
13.2 Configuration Flags	88
13.3 Flow File Output	88
13.4 Monitoring Output	88
13.5 Plugin Report Output	89
14 covertChannels	90
14.1 Description	90
14.2 Required Files	90
14.3 Configuration Flags	91
14.4 Flow File Output	91
14.5 Plugin Report Output	91
14.6 TODO	92
15 descriptiveStats	93
15.1 Description	93
15.2 Dependencies	93
15.3 Configuration Flags	93
15.4 Flow File Output	93
15.5 Known Bugs and Limitations	94

16 dfft	95
16.1 Description	95
16.2 Dependencies	95
16.3 Configuration Flags	95
16.4 Flow File Output	95
16.5 Plugin Report Output	95
17 dhcpDecode	96
17.1 Description	96
17.2 Configuration Flags	96
17.3 Flow File Output	96
17.4 Packet File Output	106
17.5 Plugin Report Output	106
17.6 TODO	106
17.7 References	106
18 dnsDecode	107
18.1 Description	107
18.2 Configuration Flags	107
18.3 Flow File Output	108
18.4 Packet File Output	114
18.5 Monitoring Output	115
18.6 Plugin Report Output	115
18.7 Example Output	115
18.8 TODO	115
19 entropy	116
19.1 Description	116
19.2 Configuration Flags	116
19.3 Flow File Output	116
19.4 Plugin Report Output	117
20 findexer	118
20.1 Description	118
20.2 Configuration Flags	118
20.3 fextractor	118
20.4 Example scenario	119
20.5 Additional Output (findexer v2)	120
20.6 Limitations	122
20.7 Old format (findexer v1)	122
21 fnameLabel	124
21.1 Description	124
21.2 Configuration Flags	124
21.3 Flow File Output	124
21.4 Packet File Output	124

22 ftpDecode	125
22.1 Description	125
22.2 Configuration Flags	125
22.3 Flow File Output	125
22.4 Packet File Output	128
22.5 Monitoring Output	128
22.6 Plugin Report Output	128
23 geoip	129
23.1 Description	129
23.2 Dependencies	129
23.3 Configuration Flags	129
23.4 Flow File Output	131
23.5 Post-Processing	133
24 gquicDecode	134
24.1 Description	134
24.2 Configuration Flags	134
24.3 Flow File Output	134
24.4 Packet File Output	135
24.5 Plugin Report Output	136
24.6 Known Bugs and Limitations	136
25 gsmDecode	137
25.1 Description	137
25.2 Dependencies	137
25.3 Configuration Flags	137
25.4 Flow File Output	140
25.5 Packet File Output	141
25.6 Plugin Report Output	142
25.7 Additional Output	142
25.8 GSM Mobile Terminating and Mobile Originating Call Call Flow Procedures	144
25.9 Post-Processing	144
25.10 Acronyms	144
25.11 References	146
26 httpSniffer	147
26.1 Description	147
26.2 Configuration Flags	147
26.3 Flow File Output	149
26.4 Packet File Output	153
26.5 Monitoring Output	153
26.6 Plugin Report Output	153
27 icmpDecode	154
27.1 Description	154
27.2 Configuration Flags	154
27.3 Flow File Output	154
27.4 Packet File Output	158

27.5 Monitoring Output	159
27.6 Plugin Report Output	159
27.7 Additional Output	159
27.8 Post-Processing	161
28 igmpDecode	163
28.1 Description	163
28.2 Required Files	163
28.3 Configuration Flags	163
28.4 Flow File Output	163
28.5 Plugin Report Output	164
28.6 Additional Output	164
28.7 Post-Processing	164
29 ircDecode	165
29.1 Description	165
29.2 Configuration Flags	165
29.3 Flow File Output	165
29.4 Plugin Report Output	168
30 jsonSink	170
30.1 Description	170
30.2 Dependencies	170
30.3 Configuration Flags	170
30.4 Plugin Report Output	171
30.5 Custom File Output	171
30.6 Output Selected Fields Only	171
30.7 Example	172
31 kafkaSink	173
31.1 Description	173
31.2 Dependencies	173
31.3 Services Initialization	173
31.4 Configuration Flags	174
31.5 Plugin Report Output	174
31.6 Example	174
32 liveXtr	176
32.1 Description	176
32.2 Dependencies	176
32.3 Configuration Flags	176
32.4 Plugin Report Output	176
32.5 Additional Output	176
33 lldpDecode	177
33.1 Description	177
33.2 Dependencies	177
33.3 Configuration Flags	177
33.4 Flow File Output	177

33.5	Packet File Output	179
33.6	Monitoring Output	179
33.7	Plugin Report Output	179
34	macRecorder	180
34.1	Description	180
34.2	Dependencies	180
34.3	Configuration Flags	180
34.4	Flow File Output	180
34.5	Packet File Output	181
34.6	Plugin Report Output	181
34.7	Example Output	181
35	mndpDecode	182
35.1	Description	182
35.2	Configuration Flags	182
35.3	Flow File Output	182
35.4	Packet File Output	183
35.5	Plugin Report Output	183
36	modbus	184
36.1	Description	184
36.2	Configuration Flags	184
36.3	Flow File Output	184
36.4	Packet File Output	186
36.5	Monitoring Output	187
36.6	Plugin Report Output	187
37	mongoSink	188
37.1	Description	188
37.2	Dependencies	188
37.3	Configuration Flags	188
37.4	Insertion of Selected Fields Only	189
37.5	Working with Timestamps (ISODate)	189
37.6	Example	189
38	mqttDecode	191
38.1	Description	191
38.2	Configuration Flags	191
38.3	Flow File Output	191
38.4	Packet File Output	193
38.5	Monitoring Output	193
38.6	Plugin Report Output	193
38.7	Additional Output	193

39 mysqlSink	194
39.1 Description	194
39.2 Dependencies	194
39.3 Database Setup	194
39.4 Configuration Flags	194
39.5 Insertion of Selected Fields Only	195
39.6 Example	196
40 nDPI	197
40.1 Description	197
40.2 Dependencies	197
40.3 Configuration Flags	197
40.4 Flow File Output	197
40.5 Packet File Output	198
40.6 nDPIMstrProto	198
40.7 Plugin Report Output	203
40.8 Additional Output	203
40.9 Post-Processing	203
40.10 How to Update nDPI to New Version	203
41 nFrstPkts	204
41.1 Description	204
41.2 Configuration Flags	204
41.3 Flow File Output	204
41.4 Post-Processing	205
42 netflowSink	206
42.1 Description	206
42.2 Dependencies	206
42.3 Configuration Flags	206
42.4 Example	206
43 ntlmsspDecode	207
43.1 Description	207
43.2 Configuration Flags	207
43.3 Flow File Output	207
43.4 Plugin Report Output	209
43.5 Additional Output	210
43.6 Post-Processing	211
43.7 References	211
44 ntpDecode	212
44.1 Description	212
44.2 Configuration Flags	212
44.3 Flow File Output	212
44.4 Monitoring Output	214
44.5 Plugin Report Output	214
44.6 Examples	214

45 ospfDecode	216
45.1 Description	216
45.2 Configuration Flags	216
45.3 Flow File Output	217
45.4 Packet File Output	219
45.5 Plugin Report Output	219
45.6 Additional Output	219
45.7 Post-Processing	220
46 p0f	222
46.1 Description	222
46.2 Dependencies	222
46.3 Configuration Flags	222
46.4 Flow File Output	222
46.5 References	223
47 payloadDumper	224
47.1 Description	224
47.2 Configuration Flags	224
47.3 Flow File Output	225
47.4 Packet File Output	225
47.5 Plugin Report Output	225
47.6 Additional Output	226
48 pcapd	227
48.1 Description	227
48.2 Dependencies	227
48.3 Configuration Flags	227
48.4 Plugin Report Output	229
48.5 Additional Output	229
48.6 Examples	229
49 pktSIATHisto	230
49.1 Description	230
49.2 Configuration Flags	230
49.3 Flow File Output	231
49.4 Post-Processing	232
49.5 Example Output	232
50 popDecode	233
50.1 Description	233
50.2 Configuration Flags	233
50.3 Flow File Output	233
50.4 Packet File Output	235
50.5 Plugin Report Output	235
50.6 TODO	235

51 portClassifier	236
51.1 Description	236
51.2 Dependencies	236
51.3 Configuration Flags	236
51.4 Flow File Output	236
51.5 Packet File Output	236
52 protoStats	237
52.1 Description	237
52.2 Dependencies	237
52.3 Configuration Flags	237
52.4 Flow File Output	237
52.5 Additional Output	237
52.6 Post-Processing	238
53 psqlSink	239
53.1 Description	239
53.2 Dependencies	239
53.3 Database Initialization	239
53.4 Configuration Flags	239
53.5 Insertion of Selected Fields Only	240
53.6 Post-Processing	241
53.7 Example	241
54 pwX	242
54.1 Description	242
54.2 Configuration Flags	242
54.3 Flow File Output	242
54.4 Plugin Report Output	243
55 quicDecode	244
55.1 Description	244
55.2 Dependencies	244
55.3 Configuration Flags	244
55.4 Flow File Output	244
55.5 Packet File Output	246
55.6 Plugin Report Output	246
55.7 Known Bugs and Limitations	247
55.8 References	247
56 radiusDecode	248
56.1 Description	248
56.2 Configuration Flags	248
56.3 Flow File Output	248
56.4 Packet File Output	254
56.5 Monitoring Output	255
56.6 Plugin Report Output	255
56.7 References	255

57 regexHyperscan	256
57.1 Description	256
57.2 Dependencies	256
57.3 Hyperscan regex format	256
57.4 Configuration Flags	257
57.5 Flow File Output	257
58 regex_pcre	258
58.1 Description	258
58.2 Dependencies	258
58.3 Configuration Flags	258
58.4 Flow File Output	262
58.5 Packet File Output	262
58.6 Plugin Report Output	262
59 regex_re2	263
59.1 Description	263
59.2 Dependencies	263
59.3 Configuration Flags	263
59.4 Flow File Output	264
59.5 Plugin Report Output	264
59.6 Known Bugs and Limitations	264
60 sctpDecode	265
60.1 Description	265
60.2 Configuration Flags	265
60.3 Flow File Output	265
60.4 Packet File Output	268
60.5 Plugin Report Output	269
61 smbDecode	270
61.1 Description	270
61.2 Configuration Flags	270
61.3 Flow File Output	271
61.4 Plugin Report Output	277
61.5 Post-Processing	277
61.6 References	277
62 smtpDecode	279
62.1 Description	279
62.2 Configuration Flags	279
62.3 Flow File Output	279
62.4 Packet File Output	280
62.5 Plugin Report Output	281
62.6 TODO	281

63 snmpDecode	282
63.1 Description	282
63.2 Configuration Flags	282
63.3 Flow File Output	282
63.4 Packet File Output	283
63.5 Plugin Report Output	283
64 socketSink	285
64.1 Description	285
64.2 Dependencies	285
64.3 Configuration Flags	285
64.4 Additional Output	286
64.5 Example	287
64.6 Post-Processing	287
64.7 t2b2t	287
65 sqliteSink	288
65.1 Description	288
65.2 Dependencies	288
65.3 Configuration Flags	288
65.4 Insertion of Selected Fields Only	289
65.5 Plugin Report Output	289
65.6 Example	289
66 sshDecode	291
66.1 Description	291
66.2 Dependencies	291
66.3 Configuration Flags	291
66.4 Flow File Output	292
66.5 Packet File Output	293
66.6 Monitoring Output	294
66.7 Plugin Report Output	294
67 sslDecode	295
67.1 Description	295
67.2 Dependencies	295
67.3 Configuration Flags	295
67.4 Flow File Output	297
67.5 Plugin Report Output	309
68 stpDecode	310
68.1 Description	310
68.2 Dependencies	310
68.3 Configuration Flags	310
68.4 Flow File Output	310
68.5 Packet File Output	312
68.6 Monitoring Output	313
68.7 Plugin Report Output	313

69 stunDecode	314
69.1 Configuration Flags	314
69.2 Flow File Output	314
69.3 Plugin Report Output	316
69.4 TODO	316
70 syslogDecode	317
70.1 Description	317
70.2 Configuration Flags	317
70.3 Flow File Output	317
70.4 Packet File Output	317
70.5 Plugin Report Output	318
70.6 TODO	318
70.7 References	318
71 tcpFlags	319
71.1 Description	319
71.2 Configuration Flags	319
71.3 Flow File Output	319
71.4 Packet File Output	326
71.5 Monitoring Output	328
71.6 Plugin Report Output	328
71.7 References	328
72 tcpStates	329
72.1 Description	329
72.2 Configuration Flags	329
72.3 Flow File Output	329
72.4 Packet File Output	330
72.5 Plugin Report Output	330
73 telegram	332
73.1 Description	332
73.2 Configuration Flags	332
73.3 Flow File Output	332
73.4 Packet File Output	333
73.5 Plugin Report Output	333
73.6 TODO	333
74 telnetDecode	334
74.1 Description	334
74.2 Configuration Flags	334
74.3 Flow File Output	334
74.4 Packet File Output	337
74.5 Plugin Report Output	337
74.6 TODO	338

75 tftpDecode	339
75.1 Description	339
75.2 Configuration Flags	339
75.3 Flow File Output	339
75.4 Packet File Output	342
75.5 Plugin Report Output	342
76 torDetector	343
76.1 Description	343
76.2 Dependencies	343
76.3 Configuration Flags	343
76.4 Flow File Output	343
76.5 Packet File Output	344
76.6 Plugin Report Output	344
76.7 Plugin Detection Method	344
76.8 Other Detection Methods	345
77 tp0f	348
77.1 Description	348
77.2 Configuration Flags	348
77.3 Flow File Output	348
77.4 Plugin Report Output	349
77.5 TODO	349
77.6 References	349
78 txtSink	350
78.1 Description	350
78.2 Dependencies	350
78.3 Configuration Flags	350
78.4 Additional Output	351
79 voipDetector	354
79.1 Description	354
79.2 Configuration Flags	354
79.3 Flow File Output	355
79.4 Packet File Output	357
79.5 Monitoring Output	357
79.6 Plugin Report Output	357
79.7 TODO	358
80 vrrpDecode	359
80.1 Description	359
80.2 Configuration Flags	359
80.3 Flow File Output	359
80.4 Monitoring Output	361
80.5 Plugin Report Output	361
80.6 Additional Output	361
80.7 Post-Processing	361

81 vtpDecode	362
81.1 Description	362
81.2 Dependencies	362
81.3 Configuration Flags	362
81.4 Flow File Output	362
81.5 Packet File Output	364
81.6 Plugin Report Output	364
81.7 Additional Output	365
81.8 References	365
82 wavelet	366
82.1 Description	366
82.2 Configuration Flags	366
82.3 Flow File Output	366
83 wechatDecode	367
83.1 Description	367
83.2 Dependencies	367
83.3 Configuration Flags	367
83.4 Flow File Output	367
83.5 Custom File Output	368
84 scripts	369
84.1 b64ex	369
84.2 fpsGplt	369
84.3 gpq3x	369
84.4 osStat	370
84.5 protStat	370
84.6 statGplt	370
84.7 t2_aliases	370
84.8 t2alive	374
84.9 t2b2t	375
84.10t2caplist	375
84.11t2conf	375
84.12t2dmon	377
84.13t2doc	378
84.14t2docker	378
84.15t2dpdk	379
84.16t2flowstat	379
84.17t2fm	379
84.18t2fuzz	379
84.19t2locate	379
84.20t2netID	379
84.21t2plot	380
84.22t2plugin	380
84.23t2rrd	380
84.24t2stat	380
84.25t2timeline	381

84.26t2topcap	381
84.27t2utils.sh	381
84.28t2viz	387
84.29t2voipconv	387
84.30t2whois	387
84.31topNStat	388
85 PDF Report Generation from PCAP using t2fm	389
85.1 Introduction	389
85.2 Prerequisites	389
85.3 PCAP to PDF in One Command	390
85.4 Step-by-Step Instructions (PCAP to PDF)	390
85.5 Step-by-Step Instructions (flow file to PDF)	390
85.6 Step-by-Step Instructions (ClickHouse / MongoDB / PostgreSQL to PDF)	391
85.7 NetFlow Records to PDF in One Command	391
85.8 Conclusion	391
86 tawk	392
86.1 Description	392
86.2 Dependencies	392
86.3 Installation	392
86.4 Usage	392
86.5 -s and -N Options	393
86.6 Related Utilities	394
86.7 Functions	394
86.8 Examples	400
86.9 t2nfdump	401
86.10t2custom	402
86.11Writing a tawk Function	402
86.12Using tawk Within Scripts	403
86.13Using tawk With Non-Tranalyzer Files	404
86.14Awk Cheat Sheet	404
86.15Awk Templates	405
86.16Examples	408
86.17FAQ	409
A Creating a Custom Plugin	412
A.1 Plugin Name	412
A.2 Plugin Number	412
A.3 Plugin Creation	412
A.4 Compilation	413
A.5 Plugin Structure	413
A.6 Error, warning, and informational messages	415
A.7 Naming Conventions	415
A.8 Generating Output	416
A.9 Accessible structures	417
A.10 Important structures	418
A.11 Generating output (advanced)	418

A.12 Writing repeated output	425
A.13 Important notes	425
A.14 Administrative functions	425
A.15 Processing functions	426
A.16 Timeout handlers	427
B Importing Tranalyzer Flows in Splunk	430
B.1 Prerequisites	430
B.2 Select Network Interface	430
B.3 Configure Tranalyzer jsonSink Plugin	430
B.4 Recompile the jsonSink Plugin	430
B.5 Start Tranalyzer2	430
B.6 Start Splunk	431
B.7 Login to Splunk, Import and Search Data	431
C Advanced Performance Enhancements with PF_RING	441
D Status	443
D.1 Global Plugins	443
D.2 Basic Plugins	443
D.3 Layer 2 Protocol Plugins	443
D.4 Layer 3/4 Protocol Plugins	443
D.5 Layer 7 Protocol Plugins	444
D.6 Application Plugins	444
D.7 Math Plugins	445
D.8 Classifier Plugins	446
D.9 Output Plugins	446
E TODO	447
E.1 Features	447
E.2 Plugins	447
F FAQ	449
F.1 After ./setup.sh, I receive the following error: -bash: t2: command not found	449
F.2 If the hashtable is full, how much memory do I need to add?	449
F.3 Can I change the timeout of a specific flow in my plugin?	449
F.4 Can I reduce the maximal flow length?	449
F.5 How can I change the separation character in the flow file?	449
F.6 How can I build all the plugins?	450
F.7 T2 failed to compile: What can I do?	450
F.8 T2 segfaults: What can I do?	450
F.9 socketSink plugin aborts with “could not connect to socket: Connection refused”	451
F.10 T2 stalls after USR1 interrupt: What can I do?	451
F.11 Can I reuse my configuration between different machines or Tranalyzer versions?	451
F.12 How to contribute code, submit a bug or request a feature?	451

1 Introduction

Tralyzer2 is a lightweight flow generator and packet analyzer designed for simplicity, performance and scalability. The program is written in C and built upon the *libpcap* library. It provides functionality to pre- and post-process IPv4/IPv6 data into flows and enables a trained user to see anomalies and network defects even in very large datasets. It supports analysis with special bit coded fields and generates statistics from key parameters of IPv4/IPv6 tcpdump traces either being live-captured from an Ethernet interface or one or several pcap files. The quantity of binary and text based output of Tralyzer2 depends on enabled modules, herein denoted as **plugins**. Hence, users have the possibility to tailor the output according to their needs and developers can develop additional plugins independent of the functionality of other plugins.

1.1 Getting Tralyzer

Tralyzer can be downloaded from: <https://tralyzer.com/downloads.html>

1.2 Dependencies

Tralyzer2 requires the following tools and libraries:

Kali/Ubuntu:

```
sudo apt-get install autoconf autoconf-archive automake libbsd-dev libpcap-dev  
libreadline-dev libtool make meson zlib1g-dev
```

Arch/Manjaro:

```
sudo pacman -S autoconf autoconf-archive automake bash-completion gcc libpcap  
libtool make meson pkgconf zlib
```

CentOS/Fedora/Red Hat:

```
sudo dnf install autoconf autoconf-archive automake bzip2 libbsd-devel  
libpcap-devel libtool meson readline-devel zlib-devel1
```

Gentoo:

```
sudo emerge autoconf autoconf-archive automake bash-completion libpcap libtool meson zlib
```

openSUSE:

```
sudo zypper install autoconf autoconf-archive automake gcc libbsd-devel  
libpcap-devel libtool meson readline-devel zlib-devel
```

macOS:

```
brew install autoconf autoconf-archive automake libpcap libtool meson readline zlib2
```

Note that `meson` is optional, but recommended as it is much faster than the autotools (`autoconf`, `automake`, ...).

¹If the `dnf` command could not be found, try with `yum` instead

²Brew is a packet manager for macOS that can be found here: <https://brew.sh>

1.3 Compilation

To build Tranalyzer2 and the plugins, run one of the following commands:

- Tranalyzer2 only:
cd "\$T2HOME"; ./autogen.sh tranalyzer2
(alternative: cd "\$T2HOME/tranalyzer2"; ./autogen.sh)
- A specific plugin only, e.g., myPlugin:
cd "\$T2HOME"; ./autogen.sh myPlugin
(alternative 1: cd "\$T2PLHOME/myPlugin"; ./autogen.sh)
(alternative 2: cd "\$T2HOME/plugins/myPlugin"; ./autogen.sh)
- Tranalyzer2 and a default set of plugins:
cd "\$T2HOME"; ./autogen.sh
- Tranalyzer2 and all the plugins in T2HOME:
cd "\$T2HOME"; ./autogen.sh -a
- Tranalyzer2 and a custom set of plugins (listed in `plugins.build`) (Section 1.3.1):
cd "\$T2HOME"; ./autogen.sh -b

where `T2HOME` points to the root folder of Tranalyzer, i.e., where the file `README.md` is located.

For finer control of which plugins to load, refer to [Enabling/Disabling Plugins](#).

Note that if `t2_aliases` is installed, the `t2build` command can be used instead of `autogen.sh`. The command can be run from anywhere, so just replace the above commands with `t2build tranalyzer2`, `t2build myPlugin`, `t2build -a` and `t2build -b`. Run `t2build --help` for the full list of options accepted by the script.

1.3.1 Custom Build

The `-b` option of the `autogen.sh` script takes an optional file name as argument. If none is provided, then the default `plugins.build` is used. The format of the file is as follows:

- Empty lines and lines starting with a '#' are ignored (can be used to prevent a plugin from being built)
- One plugin name per row
- Example:

```
# Do not build the tcpStates plugin
#tcpStates

# Build the txtSink plugin
txtSink
```

A `plugins.ignore` file can also be used to prevent specific plugins from being built. A different filename can be used with the `-I` option.

1.4 Installation

The `-i` option of the `autogen.sh` script installs Tranalyzer in `/usr/local/bin` (as `tranalyzer`) and the man page in `/usr/local/man/man1`. Note that root rights are required for the installation.

Alternatively, use the file `t2_aliases` or add the following alias to your `~/.bash_aliases`:

```
alias tranalyzer="$T2HOME/tranalyzer2/src/tranalyzer"
```

where `T2HOME` points to the root folder of Tranalyzer, i.e., where the file `README.md` is located.

The man page can also be installed manually, by calling (as root):

```
mkdir -p /usr/local/man/man1 && gzip -c man/tranalyzer.1 > /usr/local/man/man1/tranalyzer.1.gz
```

1.4.1 Aliases

The file `t2_aliases` documented in `$T2HOME/scripts/doc/scripts.pdf` contains a set of aliases and functions to facilitate working with Tranalyzer. To install it, append the following code to `~/.bashrc` or `~/.bash_aliases` (make sure to replace `$T2HOME` with the actual path, e.g., `$HOME/tranalyzer2-0.9.3`):

```
if [ -f "$T2HOME/scripts/t2_aliases" ]; then
    . "$T2HOME/scripts/t2_aliases"           # Note the leading '.'
fi
```

1.5 Getting Started

Run Tranalyzer as follows:

```
tranalyzer -r file.pcap -w outfolder/outprefix
```

For a full list of options, use Tranalyzer `-h` option: `tranalyzer -h` or refer to the complete documentation.

1.6 Getting Help

1.6.1 Documentation

Tranalyzer and every plugin come with their own documentation, which can be found in the `doc` subfolder. The complete documentation of Tranalyzer2 and all the locally available plugins can be generated by running `make` in `$T2HOME/doc`. The file `t2_aliases` provides the function `t2doc` to allow easy access to the different parts of the documentation from anywhere.

1.6.2 Man Page

If the man page was installed (Section 1.4), then accessing the man page is as simple as calling

```
man tranalyzer
```

If it was not installed, then the man page can be invoked by calling

```
man $T2HOME/tranalyzer2/man/tranalyzer.1
```

1.6.3 Help

For a full list of options, use Tranalyzer -h option: `tranalyzer -h`

1.6.4 FAQ

Refer to the complete documentation in `$T2HOME/doc` for a list of frequently asked questions.

1.6.5 Contact

Any feedback, feature requests and questions are welcome and can be sent to the development team via email at:

andy@tranalyzer.com

2 Tranalyzer2

Tranalyzer2 is designed in a modular way. Thus, the packet flow aggregation and the flow statistics are separated. While the main program performs the header dissection and flow organization, the plugins produce specialized output such as packet statistics, mathematical transformations, signal analysis and result file generation.

2.1 Supported Link-Layer Header Types

Tranalyzer handles most PCAP link-layer header types automatically. Some specific types can be analyzed by switching on flags in `linktypes.h`. The following table summarizes the link-layer header types handled by Tranalyzer:

Linktype	Description	Flags
DLT_C_HDLC	Cisco PPP with HDLC framing	
DLT_C_HDLC_WITH_DIR	Cisco PPP with HDLC framing preceded by one byte direction	
DLT_EN10MB	IEEE 802.3 Ethernet (10Mb, 100Mb, 1000Mb and up)	
DLT_FRELAY	Frame Relay	
DLT_FRELAY_WITH_DIR	Frame Relay preceded by one byte direction	
DLT_IEEE802_11	IEEE802.11 wireless LAN	
DLT_IEEE802_11_RADIO	Radiotap link-layer information followed by an 802.11 header	
DLT_IPV4	Raw IPv4	
DLT_IPV6	Raw IPv6	
DLT_JUNIPER_ATM1	Juniper ATM1 PIC (experimental)	LINKTYPE_JUNIPER=1
DLT_JUNIPER_ETHER	Juniper Ethernet (experimental)	LINKTYPE_JUNIPER=1
DLT_JUNIPER_PPPOE	Juniper PPPoE PIC (experimental)	LINKTYPE_JUNIPER=1
DLT_LAPD	Raw LAPD	LAPD_ACTIVATE=1
DLT_LINUX_LAPD	LAPD (Q.921) frames, with a DLT_LINUX_SLL header	LAPD_ACTIVATE=1
DLT_LINUX_SLL	Linux “cooked” capture encapsulation	
DLT_NULL	BSD loopback encapsulation	
DLT_PPI	Per-Packet Information	
DLT_PPP	Point-to-Point Protocol	
DLT_PPP_SERIAL	PPP in HDLC-like framing	
DLT_PPP_WITH_DIR	PPP preceded by one byte direction	
DLT_PRISM_HEADER	Prism monitor mode information followed by an 802.11 header	
DLT_RAW	Raw IP	
DLT_SYMANTEC_FIREWALL	Symantec Enterprise Firewall	

2.2 Enabling/Disabling Plugins

The plugins are stored under `~/.tranalyzer/plugins`. This folder can be changed with the `-p` option.

By default, all the plugins found in the plugin folder are loaded. This behavior can be changed by altering the value of `USE_PLIST` in `loadPlugins.h:35`. The valid options are:

USE_PLIST	Description
0	disable <code>-b</code> option and load all plugins from the plugin folder (default)
1	only load plugins present in the list (whitelist)

USE_PLIST	Description
2	do not load plugins present in the list (blacklist)

This following sections discuss the various ways to selectively enable/disable plugins.

2.2.1 Default

By default, all the files in the plugin folder named according to the following pattern are loaded:

```
^ [0-9] {3}_[a-zA-Z0-9]+.so$
```

To disable a plugin, it must be removed from the plugin folder. A subfolder, e.g., *disabled*, can be used to store unused plugins.

2.2.2 Whitelisting Plugins

If USE_PLIST=1, the whitelist (loading list) is searched under the plugins folder with the name `plugins.txt`. The name can be changed by adapting the value `PLIST` in `loadPlugins.h:36`. If the file is stored somewhere else, Tranalyzer2 `-b` option can be used.

The format of the whitelist is as follows (empty lines and lines starting with a '#' are ignored):

```
# This is a comment

# This plugin is whitelisted (will be loaded)
001_protoStats.so

# This plugin is NOT whitelisted (will NOT be loaded)
#010_basicFlow.so
```

Note that if a plugin is not present in the list, it will **NOT** be loaded.

2.2.3 Blacklisting Plugins

If USE_PLIST=2, the blacklist is searched under the plugins folder with the name `plugins.txt`. The name can be changed by adapting the value `PLIST` in `loadPlugins.h:36`. If the file is stored somewhere else, Tranalyzer2 `-b` option can be used.

The format of the blacklist is as follows (empty lines and lines starting with a '#' are ignored):

```
# This is a comment

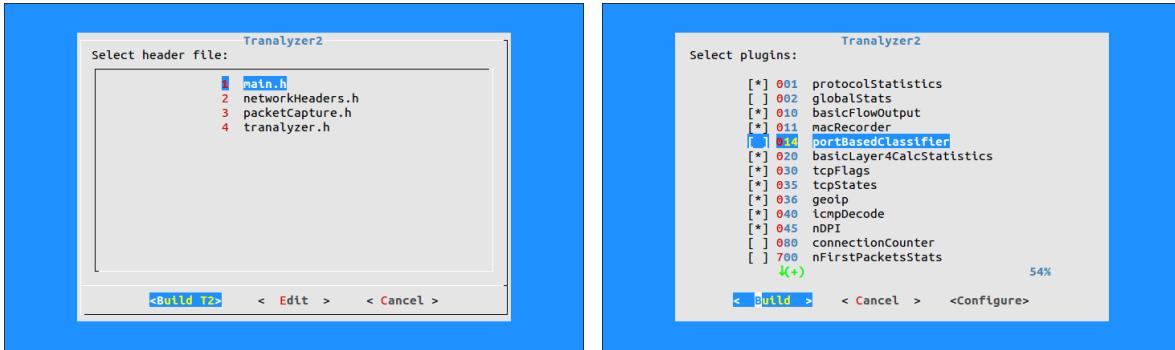
# This plugin is blacklisted (will NOT be loaded)
001_protoStats.so

# This plugin is NOT blacklisted (will be loaded)
#010_basicFlow.so
```

Note that if a plugin is not present in the list, it will be loaded.

2.2.4 Graphical Configuration and Building of T2 and Plugins

Tranalyzer2 comes with a script named `t2conf` allowing easy configuration of all the plugins through a command line based graphical menu:



Use the arrows on your keyboard to navigate up and down and between the buttons. The first window is only displayed if the `-t2` option is used. The `Edit` and `Configure` buttons will launch a text editor (`$EDITOR` or `vim`³ if the environment variable is not defined). The second window can be used to activate and deactivate plugins (toggle the active/inactive state with the space key).

To access the script from anywhere, use the provided `install.sh` script, install `t2_aliases` or manually add the following alias to `~/.bash_aliases`:

```
alias t2conf="$T2HOME/scripts/t2conf/t2conf"
```

Where `$T2HOME` is the folder containing the source code of Tranalyzer2 and its plugins.

A man page for `t2conf` is also provided and can be installed with the `install.sh` script.

2.3 Man Page

If the man page was installed (Section 1.4), then accessing the man page is as simple as calling

```
man tranalyzer
```

If it was not installed, then the man page can be invoked by calling

```
man $T2HOME/tranalyzer2/man/tranalyzer.1
```

2.4 Invoking Tranalyzer

As stated earlier Tranalyzer2 either operates on Ethernet/DAG interfaces or pcap files. It may be invoked using a BPF if only certain flows are interesting. The required arguments are listed below. Note that the `-i`, `-r`, `-R` and `-D` options cannot be used at the same time.

³The default editor can be changed by editing the variable `DEFAULT_EDITOR` (line 7)

2.4.1 Help

For a full list of options, use the `-h` option: `tranalyzer -h`

```
Tranalyzer 0.9.2 - High performance flow based network traffic analyzer
```

Usage:

```
tranalyzer [OPTION...] <INPUT>
```

Input arguments:

<code>-i IFACE</code>	Listen on interface IFACE
<code>-r PCAP</code>	Read packets from PCAP file or from stdin if PCAP is "-"
<code>-R FILE</code>	Process every PCAP file listed in FILE
<code>-D EXPR[:SCHR][,STOP]</code>	Process every PCAP file whose name matches EXPR, up to an optional last index STOP. If STOP is omitted, then Tranalyzer never stops. EXPR can be a filename, e.g., file.pcap0, or an expression, such as "dump*.pcap00", where the star matches anything (note the quotes to prevent the shell from interpreting the expression). SCHR can be used to specify the last character before the index (default: 'p')

Output arguments:

<code>-w PREFIX</code>	Append PREFIX to any output file produced. If the option is omitted, derive PREFIX from the input. Use ' <code>-w -</code> ' to output the flow file to stdout (other files will be saved as if the ' <code>-w</code> ' option had been omitted and the ' <code>-l</code> ' and ' <code>-m</code> ' options used)
<code>-W PREFIX[:SIZE][,START]</code>	Like <code>-w</code> , but fragment flow files according to SIZE, producing files starting with index START. SIZE can be specified in bytes (default), KB ('K'), MB ('M') or GB ('G'). Scientific notation, i.e., <code>1e5</code> or <code>1E5</code> (<code>=100000</code>), can be used as well. If a ' <code>f</code> ' is appended, e.g., <code>10Kf</code> , then SIZE denotes the number of flows.
<code>-l</code>	Print end report in <code>PREFIX_log.txt</code> instead of stdout
<code>-m</code>	Print monitoring output in <code>PREFIX_monitoring.txt</code> instead of stdout
<code>-s</code>	Packet forensics mode

Interface capture arguments:

<code>-S SNAPLEN</code>	Set the snapshot length (used with <code>-i</code> option)
<code>-B BUFSIZE</code>	Set the live Rx buffer size (used with <code>-i</code> option)

Optional arguments:

<code>-p PATH</code>	Load plugins from PATH instead of <code>~/.tranalyzer/plugins</code>
<code>-b FILE</code>	Use plugin list FILE instead of <code>plugin_folder/plugins.txt</code>
<code>-e FILE</code>	Create a PCAP file by extracting all packets belonging to flow indexes listed in FILE (require <code>pcapd</code> plugin)
<code>-f FACTOR</code>	Set hash multiplication factor
<code>-x ID</code>	Sensor ID
<code>-c CPU</code>	Bind tranalyzer to one core. If CPU is 0 then OS selects the

```

core to bind
-P PRIO      Set tranalyzer priority to PRIO (int) instead of 0
               (PRIO [highest, lowest]: [-20, 20] (root), [0, 20] (user))
-M FLT       Set monitoring interval to FLT seconds
-F FILE      Read BPF filter from FILE

```

Help and documentation arguments:

```

-V           Show the version of the program and exit
-h           Show help options and exit

```

Remaining arguments:

```
BPF          Berkeley Packet Filter command, as in tcpdump
```

2.4.2 -i INTERFACE

Capture data from an Ethernet interface INTERFACE (requires *root* privileges). If high volume of traffic is expected, then enable internal buffering in [ioBuffer.h](#).

```
sudo tranalyzer -i eth0
```

2.4.3 -r FILE

Capture data from a pcap file FILE.

```
tranalyzer -r file.pcap
```

The special file '-' can be used to read data from *stdin*. This can be used, e.g., to process compressed pcap files, e.g., *file.pcap.gz*, using the following command:

```
zcat file.pcap.gz | tranalyzer -r - -w out
```

2.4.4 -R FILE

Process all the pcap files listed in FILE. All files are being treated as one large file. The life time of a flow can extend over many files. The processing order is defined by the location of the filenames in the text file. The absolute path has to be specified. The t2caplist script documented in \$T2HOME/scripts/scripts.pdf can be used to generate such a list. All lines starting with a '#' are considered as comments and thus ignored.

```

$ t2caplist directory > pcap_list.txt
$ tranalyzer -R pcap_list.txt

```

2.4.5 -D FILE[*][.ext]#1[:SCHR][,#2]

Process files in a directory using file start and stop index, defined by #1 and #2 respectively. ext can be anything, e.g., .pcap, and can be omitted. If #2 is omitted and not in round robin mode, then Tranalyzer2 never stops and waits until the next file in the increment is available. If leading zeroes are used, #2 defaults to $10^{\text{number_length}} - 1$. Note that only the last occurrence of SCHR is considered, e.g., if SCHR='p', then out.pcap001 will work, but out001pcap, will not. with the : [SCHR] option a new separation character can be set, superseding SCHR defined in [tranalyzer.h](#).

The following variables in [tranalyzer.h](#) can be used to configure this mode:

Name	Default	Description
RROP	0	Activate round robin operations WARNING: if set to 1, then finDEXER will not work anymore
POLLTM	5	Poll timing (in seconds) for files
MFPTMOUT	0	> 0: timeout for poll > POLLTM, 0: no poll timeout
SCHR	'p'	Separating character for file number

For example, when using `tcpdump` to capture traffic from an interface (`eth0`) and produce 100MB files as follows:

```
sudo tcpdump -C 100 -i eth0 -w out.pcap
```

The following files are generated: `out.pcap`, `out.pcap1`, `out.pcap2`, ..., `out.pcap10`, ...

Then `SCHR` must be set to '`p`', i.e., the last character before the file number (`out.pcapNUM`) and Tranalyzer must be run as follows:

```
tranalyzer -D out.pcap
```

Or to process files 10 to 100:

```
tranalyzer -D out.pcap10,100
```

Or to process files 10 to 100 in another format:

```
tranalyzer -D out10.pcap,100 -w out
```

Or to process files from 0 to $2^{32} - 1$ using regex characters:

```
tranalyzer -D "out*.pcap" -w out
```

The last command can be shortened further, the only requirement being the presence of `SCHR` (the last character before the file number) in the pattern:

```
tranalyzer -D "*p" -w out
```

Note the quotes ("") which are necessary to avoid preemptive interpretation of regex characters and `SCHR` which MUST appear in the pattern. The same configuration can be used for filenames using one or more leading zeros, e.g., `out.pcap000`, `out.pcap001`, `out.pcap002`, ..., `out.pcap010`, ...

The following table summarizes the supported naming patterns and the configuration required:

Filenames	SCHR	Command
<code>out.pcap</code> , <code>out.pcap1</code> , <code>out.pcap2</code> , ...	'p'	<code>tranalyzer -D out.pcap -w out</code>
<code>out.pcap00</code> , <code>out.pcap01</code> , <code>out.pcap02</code> , ...	'p'	<code>tranalyzer -D out.pcap00 -w out</code>
<code>out0.pcap</code> , <code>out1.pcap</code> , <code>out2.pcap</code> , ...	't'	<code>tranalyzer -D out0.pcap -w out</code>
<code>out00.pcap</code> , <code>out01.pcap</code> , <code>out02.pcap</code> , ...	't'	<code>tranalyzer -D out00.pcap -w out</code>
<code>out_24.04.2016.20h00.pcap</code> , <code>out_24.04.2016.20h00.pcap1</code> , ...	'p'	<code>tranalyzer -D "out*.pcap" -w out</code>

Filenames	SCHR	Command
out_24.04.2016.20h00.pcap00, out_24.04.2016.20h00.pcap01, ...	'p'	tranalyzer -D "out*.pcap00" -w out
out0.pcap, out1.pcap, out2.pcap, ...	't'	tranalyzer -D out0.pcap:t -w out
out.pcap00, out.pcap01, out.pcap02, ...	'p'	tranalyzer -D out.pcap00:p -w out

2.4.6 -w PREFIX

Use a PREFIX for all output file types. The number of files being produced vary with the number of activated plugins. The file suffixes are defined in the file [tranalyzer.h](#) (see Section 2.15.13) or in the header files for the plugins. If you forget to specify an output file, Tranalyzer will use the input interface name or the file name as file prefix and print the flows to *stdout*. Thus, Tranalyzer output can be piped into other command line tools, such as netcat in order to produce centralized logging to another host or an AWK script for further post processing without intermediate writing to a slow disk storage.

2.4.7 -W PREFIX[:SIZE][,START]

This option allows the fragmentation of flow files produced by Tranalyzer independent of the input mode. The expression before the `:' is the output prefix, the expression after the `:' denotes the maximal file size for each fragment and the number after the `,' denotes the start index of the first file. If omitted it defaults to 0. The size of the files can be specified in bytes (default), KB ('K'), MB ('M') or GB ('G'). Scientific notation, i.e., 1e5 or 1E5 (=100000), can be used as well. If no size is specified, the default value of 500MB, defined by OFRWFILELN in [tranalyzer.h](#) is used. If no size is specified, then the `:' can be omitted. The same happens if no start index is specified. If an additional 'f' is appended the unit is flow count. This enables the user to produce file chunks containing the same amount of flows. Some typical examples are shown below.

Command	Fragment Size	Start Index	Output Files
tranalyzer -r nudel.pcap -W out:1.5E9,10	1.5GB	10	out10, out11, ...
tranalyzer -r nudel.pcap -W out:1.5e9,5	1.5GB	5	out5, out6, ...
tranalyzer -r nudel.pcap -W out:1.5G,1	1.5GB	1	out1, out2, ...
tranalyzer -r nudel.pcap -W out:5000K	0.5MB	0	out0, out1, ...
tranalyzer -r nudel.pcap -W out:5Kf	5000 Flows	0	out0, out1, ...
tranalyzer -r nudel.pcap -W out:180M	180MB	0	out0, out1, ...
tranalyzer -r nudel.pcap -W out:2.5G	2.5GB	0	out0, out1, ...
tranalyzer -r nudel.pcap -W out,6	OFRWFILELN	0	out6, out7, ...
tranalyzer -r nudel.pcap -W out	OFRWFILELN	0	out0, out1, ...

2.4.8 -l

All Tranalyzer command line and report output is diverted to the log file: PREFIX_log.txt. Fatal error messages still appear on the command line.

2.4.9 -m

All Tranalyzer monitoring output is diverted to the monitoring file: PREFIX_monitoring.txt.

2.4.10 -s

Initiates the packet mode where a file with the suffix `PREFIX_packets.txt` is created. The content of the file depends on the plugins loaded. The display of the packet number (first column) is controlled by `SPKTMD_PKTNO` in `main.h`. The payload can be displayed in hexadecimal and/or as characters by using the `SPKTMD_PCNTH` and `SPKTMD_PCNTC` respectively. The start of the payload (full packet, L2/3/4/7) to print is controlled by `SPKTMD_PCNTL`. A tab separated header description line is printed at the beginning of the packet file. The first two lines then read as follows:

```
%pktNo      time      pktIAT      duration      flowInd      flowStat      numHdrDesc      hdrDesc      vlanID
ethType      srcMac      dstMac      srcIP4      srcPort      dstIP4      dstPort      14Proto      ipTOS
ipID      ipIDDiff      ipFrag      ipTTL      ipHdrChkSum      ipCalChkSum      14HdrChkSum
14CalChkSum      ipFlags      pktLen      ipOptLen      ipOpts      seq      ack      seqDiff      ackDiff
seqPktLen      ackPktLen      tcpStat      tcpFlags      tcpAnomaly      tcpWin      tcpOptLen      tcpOpts
17Content
...
25      1291753225.446846      0.000000      0.000000      23      0x00006000      6      eth:vlan:mpls{2}:ipv4:
tcp      20      0x0800      00:11:22:33:44:55      66:77:88:99:aa:bb      X.Y.Z.U      62701      M.N.O.P
80      6      0x00      0x26f6      0      0x4000      62      0x6ca6      0x6ca6      0x0247      0x0247      0
x0040      460      0      0xb2a08909      0x90314073      0      0      0      0x59      0x18      0
x0000      65535      12      0x01      0x01      0x08      0x0a      0x29      0x2d      0xc3      0x6e      0x83      0x63      0xc5      0x76      GET /
images/I/01TnJ0+mhnL.png HTTP/1.1\r\nHost: ecx.images-amazon.com\r\nUser-Agent: Mozilla/5.0 (
Macintosh; U; Intel Mac OS X 10.6; de; rv:1.9.2.8) Gecko/20100722 Firefox/3.6.8\r\nAccept:
image/png,image/*;q=0.8,*/*;q=0.5\r\nAccept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3\r\n
nAccept-Encoding: gzip,deflate\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\nKeep-Alive
: 115\r\nConnection: keep-alive\r\nReferer: http://z-ecx.images-amazon.com/images/I/11
J5cf408UL.css\r\n\r\n
...
```

2.4.11 -p FOLDER

Changes the plugin folder from standard `~/.tranalyzer/plugins` to FOLDER.

2.4.12 -b FILE

Changes the plugin blacklist file from `plugin_folder/plugin_blacklist.txt` to FILE, where `plugin_folder` is either `~/.tranalyzer/plugins` or the folder specified with the `-p` option.

2.4.13 -e FLOWINDEXFILE

Denotes the filename and path of the flow index file when the `pcapd` plugin is loaded. The path and name of the pcap file depends on `FLOWINDEXFILE`. If omitted the default names for the PCAP file are defined in `pcapd.h`. The format of the `FLOWINDEXFILE` is a list of 64 bit flow indices which define the packets to be extracted from the pcap being read by the `-r` option. In general the user should use a plain file with the format displayed below:

```
# Comments (ignored)
% Flow file info (ignored)
30
3467
656697
5596
```

For more information on the `pcapd` plugin please refer to its [documentation](#).

2.4.14 -f HASHFACTOR

Sets and supersedes the HASHFACTOR constant in [tranalyzer.h](#).

2.4.15 -x SENSORID

Each T2 can have a separate sensor ID which can be listed in a flow file in order to differentiate flows originating from several interfaces during post processing, e.g., in a DB. If not specified T2_SENSORID (666), defined in [tranalyzer.h](#), will be the default value.

2.4.16 -c CPU

Bind Tranalyzer to core number CPU; if CPU == 0 then the operating system selects the core to bind.

2.4.17 -P PRIO

Set Tranalyzer priority to PRIO instead of 0 (int). PRIO MUST belong in [-20, 20] for root and in [0, 20] for standard users. The lower the number, the higher the priority, i.e., -20 has higher priority than 20.

2.4.18 -M FLT

Set monitoring interval to FLT seconds.

2.4.19 -F FILE

Read BPF filter from FILE. A filter can span multiple lines and can be commented using the '#' character (everything following a '#' is ignored).

2.4.20 BPF Filter

A Berkeley Packet Filter (BPF) can be specified at any time in order to reduce the amount of flows being produced and to increase speed during life capture ops. All rules of pcap BPF apply.

2.5 hashTable.h

Name	Default	Description
T2_HASH_FUNC	10	Hash function to use: 0: standard 1: Murmur3 32-bits 2: Murmur3 128-bits (truncated to 64-bits) 3: xxHash 32-bits 4: xxHash 64-bits 5: XXH3 64-bits 6: XXH3 128-bits (truncated to lower 64-bits) 7: CityHash64 8: MUM-hash version 3 64-bits 9: hashlittle 32-bits 10: wyhash 64-bits

Name	Default	Description
		11: FastHash32 12: FastHash64 13: t1ha0 (Linux only) [meson build backend only] 14: t1ha2 [meson build backend only]
HASHTABLE_DEBUG	0	Print debug information
HASHTABLE_NAME_LEN	7	Maximum length of a hashTable's name

2.6 ioBuffer.h

Name	Default	Description
IO_BUFFERING	0	Enables buffering of the packets in a queue
If IO_BUFFERING == 1, the following flags are available:		
IO_BUFFER_FULL_WAIT_MS	200	Number of milliseconds to wait if queue is full
IO_BUFFER_SIZE	8192	Maximum number of packets that can be stored in the buffer (power of 2)
IO_BUFFER_MAX_MTU	2048	Maximum size of a packet (divisible by 4)

2.7 loadPlugins.h

Name	Default	Description
USE_PLLIST	1	Behavior of -b option (plugin loading list): 0: disable -b option and load all plugins from the plugin folder 1: only load plugins present in the list (whitelist) 2: do not load plugins present in the list (blacklist)

2.8 main.h

Name	Default	Description	Flags
The following four flags apply to the packet mode (-s option):			
SPKTMD_PKTNO	1	Print the packet number	
SPKTMD_PCNTC	1	Print packet payload as characters	
SPKTMD_PCNTH	0	Print packet payload as hex	
SPKTMD_PCNTL	4	Content field: 0: Print the full payload of the packet 1: Print payload from L2 2: Print payload from L3 3: Print payload from L4	

Name	Default	Description	Flags
SPKTMD_BOPS	0x00	4: Print payload from L7 Bit operations on content: 0x00: MSB, no bit inverse, no shift 0x01: LSB, bit inverse 0x02: Nibble swap 0x10: Shift right 0x20: shift from last byte into extra trailing byte (requires SPKTMD_BOPS & 0x10)	
SPKTMD_BSHFT_POS	5	Shift SPKTMD_BSHFT_POS-1 at 8-SPKTMD_BSHFT into following bytes	
SPKTMD_BSHFT	2	Bitshift	SPKTMD_BOPS&0x10
SPKTMD_PCNTH_PREF	"0x"	Prefix to add to every byte in packet mode as hex (" → ab cd instead of 0xab 0xcd)	SPKTMD_PCNTH=1
SPKTMD_PCNTH_SEP	" "	Byte separator in packet mode as hex (", " → 0xab, 0xcd instead of 0xab 0xcd)	SPKTMD_PCNTH=1
MIN_MAX_ESTIMATE	0	Min/Max bandwidth statistics	
MMXLAGTMS	0.1	Min Max interval [s]	MIN_MAX_ESTIMATE=1
MMXNO0	0	Suppress 0 in MIN estimation	MIN_MAX_ESTIMATE=1

The following flags control the monitoring mode:

MONINTTHRD	1	Activate threaded interrupt handling
MONINTBLK	0	Block interrupts in main loop during packet processing (disables MONINTTHRD)
MONINTPSYNC	1	0: Interrupt printing, 1: pcap main loop synchronized printing
MONINTTMRPCP	0	0: real time base, 1: pcap time base
MONINTTMRPCP_ON	0	Startup monitoring. 0: off, 1: on
MONINTV	1.0	≥ 1 sec interval of monitoring output if USR2 is sent or MONINTTMRPCP=0
POLLENV	0	Change monitoring interval via env var \$T2MTIME
MONPROTMD	1	0: report protocol numbers, 1: report protocol names
MONPROTFL	"proto.txt"	proto file

The following flags control the DPDK multi-process mode:

DPDK_MP	0	Use DPDK multi-process mode instead of libpcap
---------	---	--

The MONPROTL2 and MONPROTL3 flags can be used to configure the L2 and L3 protocols to monitor. Their default values are

- MONPROTL2: ETHERTYPE_ARP, ETHERTYPE_RARP
- MONPROTL3: L3_TCP, L3_UDP, L3_ICMP, L3_ICMP6, L3_SCTP

2.9 networkHeaders.h

Name	Default	Description	Flags
IPV6_ACTIVATE	2	0: IPv4 only 1: IPv6 only 2: Dual mode	
ETH_ACTIVATE	1	0: No Ethernet flows 1: Activate Ethernet flows generation 2: Also use Ethernet addresses for IPv4/6 flows	
LAPD_ACTIVATE	0	0: No LAPD/Q.931 flows 1: Activate LAPD/Q.931 flow generation	
LAPD_OVER_UDP	0	0: Do not try dissecting LAPD over UDP 1: Dissect LAPD over UDP (experimental)	LAPD_ACTIVATE=1
SCTP_ACTIVATE	0	1: standard flows 1: activate SCTP chunk streams → flow 2: activate SCTP association → flow 3: activate SCTP chunk & association → flow	
SCTP_STATFINDEX	0	1: finDEX constant for all SCTP streams in a packet 0: finDEX increments	SCTP_ACTIVATE=1
MULTIPKTSUP	0	Multi-packet suppression (discard duplicated packets)	IPV6_ACTIVATE=0
T2_PRI_HDRDESC	1	1: keep track of the headers traversed	
T2_HDRDESC_AGGR	1	1: aggregate repetitive headers, e.g., vlan{2}	T2_PRI_HDRDESC=1
T2_HDRDESC_LEN	128	max length of the headers description	T2_PRI_HDRDESC=1

2.10 proto/capwap.h

Name	Default	Description	Flags
CAPWAP_SWAP_FC	1	Swap CAPWAP frame control (required for Cisco)	CAPWAP=1

2.11 proto/ethertype.h

Name	Default	Description	Flags
PW_ETH_CW	1	Detect Pseudowire (PW) Ethernet Control Word (Heuristic, experimental)	

2.12 proto/linktype.h

Name	Default	Description	Flags
LINKTYPE_JUNIPER	1	Dissect PCAP with Juniper linktypes (Experimental)	

2.13 proto/lwapp.h

Name	Default	Description	Flags
LWAPP_SWAP_FC	1	Swap LWAPP frame control (required for Cisco)	LWAPP=1

2.14 packetCapture.h

The config file *packetCapture.h* provides control about the packet capture and packet structure process of Tranalyzer2. The most important fields are described below. Please note that after changing any value in define statements a rebuild is required. Note that the PACKETLENGTH switch controls the `len` variable in the packet structure, from where the packet length is measured from. So statistical plugins such as `basicStats` can have a layer dependent output. If only L7 length is needed, use the `l7Len` variable in the packet structure.

Name	Default	Description	Flags
PACKETLENGTH	3	0: including L2, L3 and L4 header 1: including L3 and L4 header 2: including L4 header 3: only higher layer payload (Layer 7)	
FRGIPPKTLENVIEW	1	0: IP header stays with 2nd++ fragmented packets 1: IP header stripped from 2nd++ fragmented packets	PACKETLENGTH=1
NOLAYER2	0	0: Automatic L3 header discovery 1: Manual L3 header positioning	
NOL2_L3HDOFFSET	0	Offset of L3 header	NOLAYER2=1
MAXHDRCNT	5	Maximal header count (MUST be ≥ 3)	IPV6_ACTIVATE=1
SALRM	0	1: enable sending <code>FL_ALARM</code> bit for <code>pcapd</code>	
SALRMINV	0	1: invert selection	SALRM=1

2.15 tranalyzer.h

Name	Default	Description	Flags
T2_SENSORID	666	Sensor ID (can be overwritten with <code>t2 -x</code> option)	
ENVCNTRL	2	Plugins configuration mode: 0: Values from header file during compilation 1: Values from header file at runtime 2: Values from environment if defined, otherwise from header file at runtime	
REPSUP	0	Activate alive mode	
PID_FNM_ACT	0	Save the PID into a file <code>PID_FNM</code> (default: "tranalyzer.pid")	

Name	Default	Description	Flags
PKT_CB_STATS	0	Compute stats about time spent in <code>perPacketCallback()</code>	
DEBUG	0	0: no debug output 1: debug output which occurs only once or very seldom 2: + debug output which occurs in special situations, but not regularly 3: + debug output which occurs regularly (every packet)	
VERBOSE	2	0: no output 1: basic pcap report 2: + full traffic statistics 3: + info about fragmentation anomalies	
MEMORY_DEBUG	0	0: no memory debug 1: detect leaks and overflows (see <i>utils/memdebug.h</i>)	
NO_PKTS_DELAY_US	1000	If no packets are available, sleep for <i>n</i> microseconds	
NON_BLOCKING_MODE	1	Non-blocking mode	
MAIN_OUTBUF_SIZE	1000000	Size of the main output buffer	
SNAPLEN	BUFSIZ	Snapshot length (live capture)	
CAPTURE_TIMEOUT	1000	Read timeout in milliseconds (live capture)	
BPF_OPTIMIZE	0	1: Optimize BPF filters	
TSTAMP_PREC	1	Timestamp precision: 0: microseconds, 1: nanoseconds	
TSTAMP_UTC	1	Time representation: 0: localtime, 1: UTC	
TSTAMP_R_UTC	0	Time report representation: 0: localtime, 1: UTC	
ALARM_MODE	0	Only output flow if an alarm-based plugin fires	
ALARM_AND	0	0: logical OR, 1: logical AND	ALARM_MODE=1
FORCE_MODE	0	Parameter induced flow termination (NetFlow mode)	
BLOCK_BUF	0	Block unnecessary buffer output when non Tranalyzer format event based plugins are active	
USE_T2BUS	0	Use t2Bus communication backend (experimental)	
PLUGIN_REPORT	1	Enable plugins to contribute to Tranalyzer end report	
DIFF_REPORT	0	0: absolute Tranalyzer command line USR1 report 1: differential report	
MACHINE_REPORT	0	USR1 report: 0: human compliant, 1: machine compliant	
REPORT_HIST	0	Store statistical report history in <code>REPORT_HIST_FILE</code> after shutdown and reload it when restarted	
ESOM_DEP	0	Allow plugins to globally access other plugins variables	
AYIYA	1	Process AYIYA	
GENEVE	1	Process GENEVE	
TEREDO	1	Process TEREDO	
L2TP	1	Process L2TP	
GRE	1	Process GRE	
GTP	1	Process GTP (GPRS Tunneling Protocol)	
VXLAN	1	Process VXLAN	
IPIP	1	Process IPv4/6 in IPv4/6	
ETHIP	1	Process Ethernet within IP	
CAPWAP	1	Process CAPWAP	

Name	Default	Description	Flags
LWAPP	1	Process LWAPP	
DTLS	1	Process DTLS	
FRAGMENTATION	1	Activate fragmentation processing	
FRAG_HLST_CRF	1	Enables crafted packet processing	FRAGMENTATION=1
FRAG_ERROR_DUMP	0	Dumps flawed fragmented packet to stdout	FRAGMENTATION=1
IPVX_INTERPRET	0	Interpret bogus IPvX packets	
ANONYM_IP	0	1: No output of IP information	
ETH_STAT_MODE	0	0: use the innermost layer 2 type for the statistics 1: use the outermost layer 2 type for the statistics	
SUBNET_ON	1	Core control of subnet function for plugins	
RELTIME	0	0: absolute time 1: relative time	
FDURLIMIT	0	If > 0, force flow life span to $n \pm 1$ seconds	
FDLSFINDEX	0	Findex for flows of a superflow 0: different findex 1: same findex	FDURLIMIT=1
FLOW_TIMEOUT	182	Flow timeout after a packet is not seen after n seconds	
NOFLWCRT	1	SIGINT 1 create no flow, SIGINT 2 release all flows, end report	
ZPKTITMUPD	1	0: update if packets received 1: Zero Pkt actTime update active	
ZPKTTMO	1500	Number of loops until actTime update	ZPKTITMUPD=1
HASHFACTOR	1	default multiplication factor for HASHTABLE_BASE_SIZE (can be overwritten with -f option)	
HASH_CHAIN_FACTOR	2	default multiplication factor for HASHCHAIANTABLE_BASE_SIZE	
HASH_AUTOPILOT	1	When main hash map is full: 0: terminate 1: flushes oldest NUMFLWRM flow(s)	
NUMFLWRM	1	Number of flows to flush when main hash map is full	HASH_AUTOPILOT=1

Note that the PLUGIN_FOLDER flag (".*tranalyzer/plugins/*") can be either set in this file or set at runtime with `t2 -p` option.

Although not recommended, the suffix for the generated files can also be changed by editing the PACKETS_SUFFIX ("`_packets.txt`"), LOG_SUFFIX ("`_log.txt`") and MON_SUFFIX ("`_monitoring.txt`") flags.

2.15.1 -D constants

the following constants influence the file name convention:

Name	Default	Description
RROP	0	round robin operations

Name	Default	Description
POLLTM	5	poll timing for files
SCHR	'p'	separating character for file number

2.15.2 alive signal

The alive signal is a derivative of the passive monitoring mode by the USR1 signal, where the report is deactivated. If REPSUP=1 then only the command defined by REPCMDAS/W is sent to the control program defined by ALVPROG as defined below:

Name	Default	Description
REPSUP	0	0: alive mode off, 1: alive mode on, monitoring report suppressed
ALVPROG	"t2alive"	name of control program
REPCMDAS	"a='pgrep " ALVPROG "'; \ if [\$a]; then kill -USR1 \$a; fi"	alive and stall USR1 signal (no packets)
REPCMDAW	"a='pgrep " ALVPROG "'; \ if [\$a]; then kill -USR2 \$a; fi"	alive and well USR2 signal (working)

If T2 crashes or is stopped a syslog message is issued by the `t2alive` daemon. Same if T2 gets started.

2.15.3 FORCE_MODE

A 1 enables the force mode which enables any plugin to force the output of flows independent of the timeout value. Hence, Cisco NetFlow similar periodic output can be produced or overflows of counters can produce a flow and restart a new one. The macro which has to be present in the `t2OnLayer4()` function is shown below:

```
T2_RM_FLOW(flowP);
```

Figure 1: Force code line in the `t2OnFlowTerminate()` plugin routine

2.15.4 ALARM_MODE

A 1 enables the alarm mode which differs from the default flow mode by the plugin based control of the Tranalyzer core flow output. It is useful for classification plugins generating alarms, thus emulating alarm based SW such as Snort, etc. The default value is 0. The plugin sets the global output suppress variable `supOut=1` in the `t2OnFlowTerminate()` function before any output is generated. This mode also allows multiple classification plugins producing an 'AND' or an 'OR' operation if many alarm generating plugins are loaded. The variable `ALARM_AND` controls the logical alarm operation. The macro which has to be present in the `t2OnFlowTerminate()` function is shown below:

```
T2_REPORT_ALARMS(tcpWinFlowP->winThCnt);
```

Figure 2: Alarm code line in the `t2OnFlowTerminate()` plugin routine

2.15.5 BLOCK_BUF

if set to ‘1’ unnecessary buffered output from all plugins is blocked when non Tranalyzer format event based plugins are active, e.g., text-based or binary output plugins are not loaded.

2.15.6 Report Modes

Tranalyzer provides a user interrupt based report and a final report. The interrupt based mode can be configured in a variety of ways being defined below.

Name	Default	Description
PLUGIN_REPORT	0	enable plugins to contribute to the tranalyzer command line end report
DIFF_REPORT	0	1: differential, 0: Absolute tranalyzer command line USR1 report
MACHINE_REPORT	0	USR1 Report 1: machine compliant; 0: human compliant

The following interrupts are being caught by Tranalyzer2:

Signal Name	Description
SIGINT	like ^C terminates new flow production ⁴
SIGTERM	terminates tranalyzer
SIGUSR1	prints statistics report
SIGUSR2	toggles repetitive statistics report

2.15.7 State and statistical save mode

T2 is capable to preserve its internal statistical state and certain viable global variables, such as the findex.

Name	Default	Description
REPORT_HIST	0	Store statistical report history after shutdown, reload it upon restart
REPORT_HIST_FILE	"stat_hist.txt"	default statistical report history filename

The history file is stored by default under `./tranalyzer/plugins` or under the directory defined by a `-p` option.

2.15.8 L2TP

A '1' activates the L2TP processing of the Tranalyzer2 core. All L2TP headers either encapsulated in MPLS or not will be processed and followed down via PPP headers to the IP header and then passed to the IP processing. The default value of the variable is '0'. Then the stack will be parsed until the first IP header is detected. So all L2TP UDP headers having source and destination port 1701 will be processed as normal UDP packets.

2.15.9 GRE

A '1' activates the L3 General Routing Encapsulation (GRE, `l4proto=47`) processing of the Tranalyzer2 core. All GRE headers either encapsulated in MPLS or not will be processed and followed down via PPP headers to the IP header and then passed to the IP processing. The default value of the variable is 0. Then the stack will be parsed until the first IP header is detected. If the following content is not existing or compressed the flow will contain only `l4Proto=47` information.

⁴If two SIGINT interrupts are being sent in short order Tranalyzer will be terminated instantly.

2.15.10 FRAGMENTATION

A ‘1’ activates the fragmentation processing of the Tranalyzer2 core. All packets following the header packet will be assembled in the same flow. The core and the plugin [tcpFlags](#) will provide special flags for fragmentation anomalies. If FRAGMENTATION is set to 0 only the initial fragment will be processed; all later fragments will be ignored.

2.15.11 FRAG_HLST_CRFT

A ‘1’ enables crafted packet processing even when the lead fragment is missing or packets contain senseless flags as being used in attacks or equipment failure.

2.15.12 FRAG_ERROR_DUMP

A ‘1’ activates the dump of packet information on the command line for time based identification of ill-fated or crafted fragments in [tcpdump](#) or Wireshark. It provides the Unix timestamp, the six tuple, IPID and fragID as outlined in figure below.

```
MsgType    msg    time    vlan    srcIP    srcPort    dstIP    dstPort    proto    fragID    fragOffset
[WRN]  packetCapture: 1. frag not found @ 1291753225.449690 20 X.Y.Z.U 42968 M.N.O.P 52027 17 - 0
      x191F 0x00A0
[WRN]  packetCapture: 1. frag not found @ 1291753225.482611 20 X.Y.Z.U 43044 M.N.O.P 1719 17 - 0
      x1922 0x00A0
[WRN]  packetCapture: 1. frag not found @ 1291753225.492830 20 X.Y.Z.U 55841 M.N.O.P 28463 17 - 0
      x1923 0x00A0
[WRN]  packetCapture: 1. frag not found @ 1291753225.503955 20 X.Y.Z.U 25668 M.N.O.P 8137 17 - 0
      x1924 0x00A0
[WRN]  packetCapture: 1. frag not found @ 1291753225.551094 20 X.W.Z.T 41494 T.V.W.Z 27796 17 - 0
      x5A21 0x00A0
[WRN]  packetCapture: 1. frag not found @ 1291753225.639627 20 A.B.C.D 38824 E.F.G.H 55133 17 - 0
      x0DAE 0x00AC
```

Figure 3: A sample report on *stdout* for packets with an elusive first fragment

WARNING: If `FRAG_HLST_CRFT == 1` then every fragmented headerless packet will be reported!

2.15.13 *_SUFFIX

This constant defines the suffix of all plugin output files. For example if you specify the output `foo.foo` (with the `-w` option), the generated file for the per-packet output will be in the default setting `foo.foo_packets`.

2.15.14 RELTIME

`RELTIME` renders all time based plugin output into relative to the beginning of the pcap or start of packet capture. In `-D` or `-R` read operation the first file defines the start time.

2.15.15 FLOW_TIMEOUT

This constant specifies the default time in seconds (182) after which a flow will be considered as terminated since the last packet is captured. Note: Plugins are able to change the timeout values of a flow. For example the [tcpStates](#) plugin adjusts the timeout of a flow according to the TCP state machine. A reduction of the flow timeout has an effect on the necessary flow memory defined in `HASHCHAIANTABLE_SIZE`, see below.

2.15.16 FDURLIMIT

FDURLIMIT defines the maximum flow duration in seconds which is then forced to be released. It is a special force mode for the duration of flows and a special feature for Dalhousie University. If FDURLIMIT > 0 then FLOW_TIMEOUT is overwritten if FURLIMIT seconds are reached.

2.15.17 HASHFACTOR

A factor to be multiplied with the HASHTABLE_SIZE described below. It facilitates the correct setting of the hash space. Moreover, if T2 runs out of hash it will give an upper estimate the user can choose for HASHFACTOR. Set it to this value, recompile and rerun T2. This constant is superseded by the [-f option](#).

2.15.18 HASHTABLE_SIZE

The number of buckets in the hash table. As a separate chaining hashing method is used, this value does not denote the amount of elements the hash table is able to manage! The larger, the less likely are hash collisions. The current default value is 2^{18} . Its value should be selected at least two times larger as the value of HASHCHAIANTABLE_SIZE discussed in the following chapter.

2.15.19 HASH_CHAIN_FACTOR

A factor to be multiplied with the HASHCHAIANTABLE_SIZE described below.

2.15.20 HASHCHAIANTABLE_SIZE

Specifies the amount of flows the main hash table is able to manage. The default value is 2^{19} , so roughly half the size of HASHTABLE_SIZE. T2 supplies information about the hash space in memory in: Max number of IPv4 flows in memory: 113244 (50.220%). Together with the amount of traffic already processed the total value can be computed. An example is given in Figure 1.

2.15.21 HASH_AUTOPILOT

Default 1. Avoids overrun of main hash, flushes oldest flow on every flow insert if hash map is full. 0 disables hash overrun protection. If speed is an issue avoid overruns by invoking T2 with the [-f option](#) set to the value recommended by T2.

2.15.22 SUBNET_ON

Since the version 0.8.8 the core controls the subnet functions instead of [basicFlow](#) as now the aggregation mode according to countries or organization is possible.

2.15.23 utils.h

The following flags can be used to configure the subnet files and the output of the plugins:

Name	Default	Description
SUBRNG	0	Subnet definition: 0: CIDR only 1: Begin-End

Name	Default	Description
CNTYCTY	0	1: Add the county and city
CNTYLEN	14	length of County record
CTYLEN	14	length of City record
WHOLEN	27	length of Organization record

If any of those flags is changed, `tranalyzer` **MUST** be recompiled with `t2build -f` in order to generate a new binary subnet file, as we only load the info the user needs.

2.15.24 Format of the Subnet Files

The text format of the `subnets4.txt` and `subnets6.txt` files is defined as follows:

- A '-' in the first column (prefix/mask) denotes a non-CIDR range. In this case, Tranalyzer reads the 2nd column instead of the 1st when `SUBRNG=1` in `utils.h`.
- If `SUBRNG=0`, the 2nd column is ignored and only CIDR ranges are accepted.
- Country and Whois 32 bit hex code: cccc cccc cTww wwww wwww wwww wwww where c: Country, T Tor address bit, w: Organization
- Autonomous System Number (ASN)
- Uncertainty of location in km
- Latitude
- Longitude
- Country
- County
- City
- Organization

An extraction of `subnets4.txt` is depicted below:

The text files `subnets4.txt` and `subnets6.txt` can be edited and manually converted, just move to `utils/subnet` directory and invoke the following command:

```
./subconv subnets4.txt and ./subconv subnets6.txt
```

2.15.25 Tor Information

Since 0.8.8 Tor information is also available for IPv6. The conversion programs from the raw files to T2 format can be found under `utils/subnet/tor`. It can be controlled also by `subconv`, see options below:

#	5	16122020	IPCIDR	IPRange CtryWhoCode	ASN	Uncert	Latitude	Longitude	Country
County	City	Org							
# Begin IPv4 private address space									
10.0.0.0/8	10.0.0.0-10.255.255.255	0x0301c2a7	0	-1.0	666.000000	666.000000			
04	- -	Private network							
14.0.0.0/8	14.0.0.0-14.255.255.255	0x00000000	0	-1.0	666.000000	666.000000			
03	- -	Public data networks							
24.0.0.0/8	24.0.0.0-24.255.255.255	0x00000000	0	-1.0	666.000000	666.000000			
09	- -	Cable television networks							
127.0.0.0/8	127.0.0.0-127.255.255.255	0x01014fe7	0	-1.0	666.000000	666.000000			
666.000000	01 - -	Loopback							
100.64.0.0/10	100.64.0.0-100.127.255.255	0x0702041f	0	-1.0	666.000000	666.000000			
666.000000	20 - -	Shared address space							
169.254.0.0/16	169.254.0.0-169.254.255.255	0x02014965	0	-1.0	666.000000	666.000000			
666.000000	02 - -	Link-local							
172.16.0.0/12	172.16.0.0-172.31.255.255	0x0381c2a7	0	-1.0	666.000000	666.000000			
666.000000	05 - -	Private network							
192.0.0.0/24	192.0.0.0-192.0.0.255	0x0401c2a7	0	-1.0	666.000000	666.000000			
06	- -	Private network							
192.0.2.0/24	192.0.2.0-192.0.2.255	0x07823fb8	0	-1.0	666.000000	666.000000			
21	- -	TEST-NET-1							
192.88.99.0/24	192.88.99.0-192.88.99.255	0x0b011c29	0	-1.0	666.000000	666.000000			
666.000000	60 - -	IPv6 to IPv4 relay							
192.168.0.0/16	192.168.0.0-192.168.255.255	0x0481c2a7	0	-1.0	666.000000	666.000000			
666.000000	07 - -	Private network							
198.18.0.0/15	198.18.0.0-198.19.255.255	0x0501c2a7	0	-1.0	666.000000	666.000000			
666.000000	08 - -	Private network							
198.51.100.0/24	198.51.100.0-198.51.100.255	0x08023fb9	0	-1.0	666.000000	666.000000			
666.000000	22 - -	TEST-NET-2							
203.0.113.0/24	203.0.113.0-203.0.113.255	0x08823fb8	0	-1.0	666.000000	666.000000			
666.000000	23 - -	TEST-NET-3							
224.0.0.0/4	224.0.0.0-239.255.255.255	0x06017598	0	-1.0	666.000000	666.000000			
666.000000	10 - -	Multicast							
...									
240.0.0.0/4	240.0.0.0-255.255.255.254	0x0901dd04	0	-1.0	666.000000	666.000000			
666.000000	24 - -	Reserved							
255.255.255.255/32	255.255.255.255-255.255.255.255	0x06804ca0	0	-1.0	666.000000	666.000000			
666.000000	11 - -	Broadcast							
# End IPv4 private address space									
1.0.0.0/24	1.0.0.0-1.0.0.255	0x8480205e	13335	80.000000	34.052231				
-118.243683	us California	Los Angeles		APNIC Research and Development					
1.0.1.0/24	1.0.1.0-1.0.1.255	0x260062cc	0	80.000000	26.061390				
119.306107	cn Fujian	Fuzhou CHINANET FUJIAN PROVINCE NETWORK							
1.0.2.0/23	1.0.2.0-1.0.3.255	0x260062cc	0	80.000000	26.061390				
119.306107	cn Fujian	Fuzhou CHINANET FUJIAN PROVINCE NETWORK							
1.0.4.0/24	1.0.4.0-1.0.4.255	0x148284d8	56203	80.000000	-37.813999				
144.963318	au Victoria	Melbourne Wirefreebroadband Pty Ltd							
...									

\$./subconv -h

Usage:

subconv [OPTION...] <subnets.txt>

Optional arguments:

-4 Generate subnet file for IPv4

```

-6           Generate subnet file for IPv6

-t           Add Tor info to subnet file
-a           Download and add Tor info to subnet file
-c           Convert from Tor JSON info to subnet file

-h, --help    Show this help, then exit

```

So to convert the IPv4 subnet file and add existing Tor info invoke the following command:

```
./subconv -t subnets4.txt
```

2.15.26 Aggregation Mode

The aggregation mode enables the user to confine certain IP, port or protocol ranges into a single flow. The variable AGGREGATIONFLAG in [tranalyzer.h](#) defines a bit field which enables specific aggregation modes according to the six tuple values listed below.

Aggregation Flag	Value
L4PROT	0x01
DSTPORT	0x02
SRCPORT	0x04
DSTIP	0x08
SRCIP	0x10
VLANID	0x20
SUBNET	0x80

If a certain aggregation mode is enabled the following variables in [tranalyzer.h](#) define the aggregation range.

Aggregation Flag	Type	Description
SRCIP4CMSK	uint8_t	src IPv4 aggregation CIDR mask
DSTIP4CMSK	uint8_t	dst IPv4 aggregation CIDR mask
SRCIP6CMSK	uint8_t	src IPv6 aggregation CIDR mask
DSTIP6CMSK	uint8_t	dst IPv6 aggregation CIDR mask
SRCPORTLW	uint16_t	src port lower bound
SRCPORTHW	uint16_t	src port upper bound
DSTPORTLW	uint16_t	dst port lower bound
DSTPORTHW	uint16_t	dst port upper bound

If SUBNET is chosen then all flows are aggregated according to a 32-bit hex subnet mask of the organization country hex code.

```
// SUBNET mode: IP flow aggregation network masks
#define CNTRY_MSK 0xff800000
#define TOR_MSK   0x00400000
#define ORG_MSK   0x003fffff
#define NETIDMSK  (CNTRY_MSK | ORG_MSK) // netID mask
```

2.16 bin2txt.h

Name	Default	Description
IP4_FORMAT	0	IPv4 addresses representation: 0: normal, 1: normalized (padded with zeros), 2: one 32-bits hex number 3: one 32-bits unsigned number
IP6_FORMAT	0	IPv6 addresses representation: 0: compressed, 1: uncompressed, 2: one 128-bits hex number, 3: two 64-bits hex numbers
MAC_FORMAT	0	MAC addresses representation: 0: normal (edit MAC_SEP to change the separator), 1: one 64-bits hex number
MAC_SEP	" ":"	Separator to use in MAC addresses: 11:22:33:44:55:66
B2T_NON_IP_STR	" - "	Representation of non-IPv4/IPv6 addresses in IP columns
HEX_CAPITAL	0	Hex output: 0: lower case; 1: upper case
TFS_EXTENDED_HEADER	0	Extended header in flow file
TFS_NC_TYPE	2	Types in header file: 0: none, 1: numbers, 2: C types
TFS_SAN_UTF8	1	Activates the UTF-8 sanitizer for strings
B2T_TIMESTR	0	Print Unix timestamps as human readable dates
HDR_CHR	" % "	start character(s) of comments
SEP_CHR	" \t "	column separator in the flow file ";", ".", "_" and "\ " should not be used
JSON_KEEP_EMPTY	0	Output empty fields
JSON_PRETTY	0	Add spaces to make the output more readable

2.17 gz2txt.h

Name	Default	Description
USE_ZLIB	1	Activate code for gzip-(de)compression

2.18 outputBuffer.h

Name	Default	Description	Flags
BUF_DATA_SHFT	0	Adds for each binary output record the length and shifts the record by n <code>uint32_t</code> words to the right (see <code>binSink</code> and <code>socketSink</code> plugin)	
OUTBUF_AUTOPILOT	1	Automatically increase the output buffer when required	
OUTBUF_MAXSIZE_F	5	Maximal factor to increase the output buffer size to	OUTBUF_AUTOPILOT=1

2.19 rbTree.h

Name	Default	Description
RBT_DEBUG	0	Enable debug output
RBT_ROTATION	0	Activate the Red-Black Tree feature of rotating an unbalanced tree

2.20 subnetHL.h

Name	Default	Description
SUBRNG	0	IP range definition: 0: CIDR only, 1: Begin-End
CNTYCTY	0	Output the county and the city
WHOADDR	0	Add whois address info
SUB_MAP	1	Use mmap to load the subnet file
CNTYLEN	14	Length of County record
CTYLEN	14	Length of City record
WHOLEN	30	Length of Organization record
ADDRLEN	30	Length of Address record
SUBNET_UNK	"-"	Representation of unknown locations

2.21 t2log.h

Name	Default	Description
T2_LOG_COLOR	1	Whether or not to color messages

2.22 Tranalyzer2 Output

As stated before, the functionality and output of Tranalyzer2 is defined by the activated plugins. Basically, there are two ways a plugin can generate output. First, it can generate its own output file and write any arbitrary content into any stream. The second way is called standard output or per-flow output. After flow termination Tranalyzer2 provides an output buffer and appends the direction of the flow to it. For example, in case of textual output, an "A" flow is normally followed by a "B" flow or if the "B" flow does not exist it is followed by the next "A" flow. Then, the output buffer is passed to the plugins providing their per-flow output. Finally the buffer is sent to the activated output plugins. This process repeats itself for the "B" flow. For detailed explanation about the functionality of the output plugins refer to the section plugins.

2.22.1 Hierarchical Ordering of Numerical or Text Output

Tranalyzer2 provides a hierarchical ordering of each output. Each plugin controls the:

- volume of its output
- number of values or bins

- hierarchical ordering of the data
- repetition of data substructures

Thus, complex structures such as lists or matrices can be presented in a single line.

The following sample of text output shows the hierarchical ordering for four data outputs, separated by tabulators:

A	0.3	2.0_3.4_2.1	2;4;2;1	(1_2_9)_ (1_3_1)_ (7_5_3)_ (2_3_7)
---	-----	-------------	---------	------------------------------------

The A indicates the direction of the flow; in this case it is the initial flow. The next number denotes a singular descriptive statistical result. Output number two consists of three values separated by “_” characters. Output number three consists of one value, that can be repeated, indicated by the character “;”. Output number four is a more complex example: It consists of four values containing three subvalues indicated by the braces. This could be interpreted as a matrix of size 4×3 .

2.23 Final Report

Standard configuration of Tranalyzer2 produces a statistical report to *stdout* about timing, packets, protocol encapsulation type, average bandwidth, dump length, etc. A sample report including some current protocol relevant warnings is depicted in the figure below. Warnings are not fatal hence are listed at the end of the statistical report when Tranalyzer2 terminates naturally. The *Average total Bandwidth* estimation refers to the processed bandwidth during the data acquisition process. It is only equivalent to the actual bandwidth if the total packet length including all encapsulations is not truncated and all traffic is IP. The *Average IP Traffic Bandwidth* is an estimate comprising all IP traffic actually present on the wire. Plugins can report extra information when `PLUGIN_REPORT` is activated. This report can be saved in a file, by using one of the following command:

```
tranalyzer -r file.pcap -w out -l (See Section 2.4.8)
tranalyzer -r file.pcap -w out | tee out_stdout.txt
tranalyzer -r file.pcap -w out > out_stdout.txt
```

Both commands will create a file `out_stdout.txt` containing the report. The only difference between those two commands is that the first one still outputs the report to *stdout*.

Fatal errors regarding the invocation, configuration and operation of Tranalyzer2 are printed to *stderr* after the plugins are loaded, thus before the processing is activated, see the *Hash table error* example in Listing 1. These errors terminate Tranalyzer2 immediately and are located before the final statistical report as being indicated by the “*Shutting down...*” key phrase. If the final report is to be used in a following script a pipe can be appended and certain lines can be filtered using `grep` or `awk`.

\$./tranalyzer -r ~/data/knoedel.pcap -w ~/results/ ===== Tranalyzer 0.8.9 (Anteater), Tarantula. PID: 3426 ===== [INF] Creating flows for L2, IPv4, IPv6 Active plugins: 01: protoStats, 0.8.9 02: basicFlow, 0.8.9 03: macRecorder, 0.8.9 04: portClassifier, 0.8.9 05: basicStats, 0.8.9 06: tcpFlags, 0.8.9 07: tcpStates, 0.8.9 08: icmpDecode, 0.8.9 09: dnsDecode, 0.8.9
--

```

10: httpSniffer, 0.8.9
11: connStat, 0.8.9
12: txtSink, 0.8.9
[INF] IPv4 Ver: 5, Rev: 16122020, Range Mode: 0, subnet ranges loaded: 406027 (406.03 K)
[INF] IPv6 Ver: 5, Rev: 17122020, Range Mode: 0, subnet ranges loaded: 50974 (50.97 K)
Processing file: /home/wurst/knoedel.pcap
Link layer type: Ethernet [EN10MB/1]
Dump start: 1291753225.446732 sec (Tue 07 Dec 2010 20:20:25 GMT)
[WRN] snapL2Length: 1550 - snapL3Length: 1484 - IP length in header: 1492
[WRN] Hash Autopilot: main HashMap full: flushing 1 oldest flow(s)
[INF] Hash Autopilot: Fix: Invoke Tranalyzer with '-f 5'
Dump stop : 1291753452.373884 sec (Tue 07 Dec 2010 20:24:12 GMT)
Total dump duration: 226.927152 sec (3m 46s)
Finished processing. Elapsed time: 171.835756 sec (2m 51s)
Finished unloading flow memory. Time: 182.101116 sec (3m 2s)
Percentage completed: 100.00%
Number of processed packets: 53982409 (53.98 M)
Number of processed bytes: 42085954664 (42.09 G)
Number of raw bytes: 42101578296 (42.10 G)
Number of pad bytes: 25023 (25.02 K)
Number of pcap bytes: 42949673232 (42.95 G)
Number of IPv4 packets: 53768017 (53.77 M) [99.60%]
Number of IPv6 packets: 214107 (214.11 K) [0.40%]
Number of A packets: 31005806 (31.01 M) [57.44%]
Number of B packets: 22976603 (22.98 M) [42.56%]
Number of A bytes: 14211017503 (14.21 G) [33.77%]
Number of B bytes: 27874937161 (27.87 G) [66.23%]
Average A packet load: 458.33
Average B packet load: 1213.19 (1.21 K)
-----
macRecorder: MAC pairs per flow: min: 1, max: 3, average: 1.00
basicStats: Biggest L2 talker: xx:xx:xx:xx:xx:xx: 68 [0.00%] packets
basicStats: Biggest L2 talker: xx:xx:xx:xx:xx:xx: 85480 (85.48 K) [0.00%] bytes
basicStats: Biggest L3 talker: v.v.v.v (GB): 154922 (154.92 K) [0.29%] packets
basicStats: Biggest L3 talker: w.w.w.w (GB): 236308004 (236.31 M) [0.56%] bytes
tcpFlags: Aggregated ipFlags: 0x3d6e
tcpFlags: Aggregated tcpAnomaly: 0xfe07
tcpFlags: Number of TCP scans attempted, successful: 88849 (88.85 K), 140408 (140.41 K) [158.03%]
tcpFlags: Number of TCP SYN retries, seq retries: 65369 (65.37 K), 15783 (15.78 K)
tcpFlags: Number WinSz below 1: 162725 (162.72 K) [0.38%]
tcpFlags: Number of MPTCP packets: 6 [0.00%]
tcpFlags: Aggregated MPTCP types: 0x0008 and flags: 0x00
tcpStates: Aggregated tcpStates anomalies: 0xdf
icmpDecode: Aggregated icmpStat: 0x31
icmpDecode: Number of ICMP echo request packets: 14979 (14.98 K) [12.00%]
icmpDecode: Number of ICMP echo reply packets: 2690 (2.69 K) [2.16%]
icmpDecode: ICMP echo reply / request ratio: 0.18
icmpDecode: Number of ICMPv6 echo request packets: 1440 (1.44 K) [9.63%]
icmpDecode: Number of ICMPv6 echo reply packets: 951 [6.36%]
icmpDecode: ICMPv6 echo reply / request ratio: 0.66
dnsDecode: Number of DNS packets: 237597 (237.60 K) [0.44%]
dnsDecode: Number of DNS Q packets: 125260 (125.26 K) [52.72%]
dnsDecode: Number of DNS R packets: 112337 (112.34 K) [47.28%]
dnsDecode: Aggregated dnsStat: 0xf72f
httpSniffer: Number of HTTP packets: 36623492 (36.62 M) [67.84%]
httpSniffer: Number of HTTP GET requests: 426825 (426.82 K) [1.17%]
httpSniffer: Number of HTTP POST requests: 39134 (39.13 K) [0.11%]
httpSniffer: HTTP GET/POST ratio: 10.91
httpSniffer: Aggregated httpStat : 0x003f
httpSniffer: Aggregated httpAFlags : 0x5143

```

```

httpSniffer: Aggregated httpCFlags : 0x007a
httpSniffer: Aggregated httpHeadMimes: 0x80ef
httpSniffer: Number of files img_vid_aud_msg_txt_app_unk: 192890_5262_401_95_149772_91634_1958
connStat: Number of unique source IPs: 275311 (275.31 K)
connStat: Number of unique destination IPs: 301003 (301.00 K)
connStat: Number of unique source/destination IPs connections: 1242 (1.24 K)
connStat: Max unique number of source IP / destination port connections: 1521 (1.52 K)
connStat: IP prtcon/sdcon, prtcon/scon: 1.224638, 0.005525
connStat: Source IP with max connections: X.Y.Z.U (HU): 1515 (1.51 K) connections
connStat: Destination IP with max connections: L.M.N.O (FI): 3241 (3.24 K) connections
-----
Headers count: min: 4, max: 13, average: 7.10
Max VLAN header count: 1
Max MPLS header count: 2
Number of LLC packets: 285 [0.00%]
Number of GRE packets: 285 [0.00%]
Number of Teredo packets: 213876 (213.88 K) [0.40%]
Number of AYIYA packets: 231 [0.00%]
Number of IGMP packets: 401 [0.00%]
Number of ICMP packets: 124800 (124.80 K) [0.23%]
Number of ICMPv6 packets: 14946 (14.95 K) [0.03%]
Number of TCP packets: 43273341 (43.27 M) [80.16%]
Number of TCP bytes: 36129271238 (36.13 G) [85.85%]
Number of UDP packets: 10311931 (10.31 M) [19.10%]
Number of UDP bytes: 5832263590 (5.83 G) [13.86%]
Number of IPv4 fragmented packets: 19155 (19.16 K) [0.04%]
Number of IPv6 fragmented packets: 9950 (9.95 K) [4.65%]
~~~~~
Number of processed flows: 1438758 (1.44 M)
Number of processed A flows: 1209354 (1.21 M) [84.06%]
Number of processed B flows: 229404 (229.40 K) [15.94%]
Number of request flows: 930585 (930.59 K) [64.68%]
Number of reply flows: 508173 (508.17 K) [35.32%]
Total A/B flow asymmetry: 0.68
Total req/rply flow asymmetry: 0.29
Number of processed packets/flows: 37.52
Number of processed A packets/flows: 25.64
Number of processed B packets/flows: 100.16
Number of processed total packets/s: 237884.31 (237.88 K)
Number of processed A+B packets/s: 237884.31 (237.88 K)
Number of processed A packets/s: 136633.30 (136.63 K)
Number of processed B packets/s: 101251.01 (101.25 K)
~~~~~
Number of average processed flows/s: 6340.18 (6.34 K)
Average full raw bandwidth: 1484232320 b/s (1.48 Gb/s)
Average snapped bandwidth : 1483681536 b/s (1.48 Gb/s)
Average full bandwidth : 1483899136 b/s (1.48 Gb/s)
Max number of flows in memory: 262144 (262.14 K) [100.00%]
Number of flows terminated by autopilot: 815749 (815.75 K) [56.70%]
Memory usage: 2.49 GB [3.70%]
Aggregate flow status: 0x0c00bcfad298fb04
[WRN] L3 SnapLength < Length in IP header
[WRN] L4 header snapped
[WRN] Consecutive duplicate IP ID
[WRN] IPv4/6 payload length > framing length
[WRN] IPv4/6 fragmentation header packet missing
[WRN] IPv4/6 packet fragmentation sequence not finished
[INF] Ethernet flows
[INF] IPv4 flows
[INF] IPv6 flows

```

```
[INF] VLAN encapsulation
[INF] IPv4/6 fragmentation
[INF] MPLS encapsulation
[INF] L2TP encapsulation
[INF] PPP/HDLC encapsulation
[INF] GRE encapsulation
[INF] AYIYA tunnel
[INF] Teredo tunnel
[INF] CAPWAP/LWAPP tunnel
[INF] IPsec AH
[INF] IPsec ESP
[INF] SSDP/UPnP
[INF] SIP/RTP
```

Listing 1: A sample Tranalyzer2 final report including encapsulation warning, Hash Autopilot engagement when hash table full

T2 runs in IPv4 mode, but warns the user that there is IPv6 encapsulated. Note that the new Hash Autopilot warns you when the main hash map is full. T2 then removes the oldest flow and continues processing your pcap. To avoid that, run T2 again, but this time, use the `-f 5` option as indicated in the warning message:

```
[INF] Hash Autopilot: Fix: Invoke Tranalyzer with '-f 5'
$ t2 -r ~/wurst/data/knoedel.pcap -w ~/results -f 5
```

or just let it run to the finish.

2.24 Monitoring Modes During Runtime

If debugging is deactivated or the verbose level is zero (see Section 2.15), Tranalyzer2 prints no status information or end report. Since 0.8.8 the control of T2 can be achieved without the knowledge of the PID via `t2stat`:

```
t2stat -h
Usage:
  t2stat [OPTION...]

Optional arguments:
  -INTERVAL      Send a signal to Tranalyzer every INTERVAL seconds
  -SIGNAME       Send SIGNAME signal instead of USR1
  -s             Run the command as root (with sudo)
  -p             Print Tranalyzer PID(s) and exit
  -l             List Tranalyzer PID(s), commands, running time and exit
  -i             Interactively cycle through all Tranalyzer processes

Help and documentation arguments:
  -h             Show help options and exit
```

Here are some examples

Command	Description
<code>t2stat</code>	T2 sends configured monitoring report to stdout
<code>t2stat 5</code>	T2 sends configured monitoring report to stdout every 5 seconds
<code>t2stat -USR2</code>	T2 toggles between on demand and continuous monitoring operation
<code>t2stat -SIGINT</code>	stop flow creation (like <code>^C</code> in the shell)
<code>t2stat -TERM</code>	terminate T2 immediately (Like two times <code>^C</code> in the shell)

The script t2stat has the same function as kill -USR1 PID. An example of a typical signal requested report (MACHINE_REPORT=0) is shown in Listing 2.

```

@      @
|      |
=====vVv==(a      a)==vVv=====
=====\\ /=====
=====\\ /=====
oo
USR1 A type report: Tranalyzer 0.8.9 (Anteater), Tarantula. PID: 3434
PCAP time: 1291753338.831347 sec (Tue 07 Dec 2010 20:22:18 GMT)
PCAP duration: 113.384615 sec (1m 53s)
Time: 1613737828.648459 sec (Fri 19 Feb 2021 13:30:28 CET)
Elapsed time: 66.478134 sec (1m 6s)
Processing file: /home/wurst/knoedel.pcap
Total bytes to process: 42949673232 (42.95 G)
Percentage completed: 50.60%
Total bytes processed so far: 21730424832 (21.73 G)
Remaining time: 64.914335 sec (1m 4s)
ETF: 1613737893.562794 sec (Fri 19 Feb 2021 13:31:33 CET)
Number of processed packets: 27154806 (27.15 M)
Number of processed bytes: 21295948035 (21.30 G)
Number of raw bytes: 21303536835 (21.30 G)
Number of pad bytes: 12158 (12.16 K)
Number of IPv4 packets: 27051228 (27.05 M) [99.62%]
Number of IPv6 packets: 103450 (103.45 K) [0.38%]
Number of A packets: 15590982 (15.59 M) [57.42%]
Number of B packets: 11563824 (11.56 M) [42.58%]
Number of A bytes: 7188873480 (7.19 G) [33.76%]
Number of B bytes: 14107074555 (14.11 G) [66.24%]
Average A packet load: 461.09
Average B packet load: 1219.93 (1.22 K)
-----
tcpFlags: Number of TCP scans attempted, successful: 29501 (29.50 K), 51309 (51.31 K) [173.92%]
tcpFlags: Number of TCP SYN retries, seq retries: 32048 (32.05 K), 7843 (7.84 K)
icmpDecode: Aggregated icmpStat: 0x21
icmpDecode: Number of ICMP echo request packets: 8244 (8.24 K) [11.58%]
icmpDecode: Number of ICMP echo reply packets: 1948 (1.95 K) [2.74%]
dnsDecode: Number of DNS packets: 122201 (122.20 K) [0.45%]
dnsDecode: Number of DNS Q packets: 64102 (64.10 K) [52.46%]
dnsDecode: Number of DNS R packets: 58099 (58.10 K) [47.54%]
dnsDecode: Aggregated dnsStat: 0x0201
httpSniffer: Number of HTTP packets: 18450263 (18.45 M) [67.94%]
connStat: Number of unique source IPs: 136530 (136.53 K)
connStat: Number of unique destination IPs: 151775 (151.78 K)
connStat: Number of unique source/destination IPs connections: 1456 (1.46 K)
connStat: Max unique number of source IP / destination port connections: 2019 (2.02 K)
connStat: IP prtcon/sdcon, prtcon/scon: 1.386676, 0.014788
connStat: Source IP with max connections: x.x.x.x (CH): 1165 (1.17 K) connections
connStat: Destination IP with max connections: y.y.y.y (CH): 5028 (5.03 K) connections
-----
Headers count: min: 4, max: 13, average: 7.10
Max VLAN header count: 1
Max MPLS header count: 2
Number of LLC packets: 128 [0.00%]
Number of GRE packets: 128 [0.00%]
Number of Teredo packets: 103318 (103.32 K) [0.38%]
Number of AYIYA packets: 132 [0.00%]
Number of IGMP packets: 168 [0.00%]
Number of ICMP packets: 63517 (63.52 K) [0.23%]
```

```

Number of ICMPv6 packets: 7691 (7.69 K) [0.03%]
Number of TCP packets: 21820890 (21.82 M) [80.36%]
Number of TCP bytes: 18347271411 (18.35 G) [86.15%]
Number of UDP packets: 5137175 (5.14 M) [18.92%]
Number of UDP bytes: 2890294860 (2.89 G) [13.57%]
Number of IPv4 fragmented packets: 8295 (8.29 K) [0.03%]
Number of IPv6 fragmented packets: 3238 (3.24 K) [3.13%]
=====
Number of processed flows: 707800 (707.80 K)
Number of processed A flows: 593821 (593.82 K) [83.90%]
Number of processed B flows: 113979 (113.98 K) [16.10%]
Number of request flows: 582056 (582.06 K) [82.23%]
Number of reply flows: 125744 (125.74 K) [17.77%]
Total A/B flow asymmetry: 0.68
Total req/rply flow asymmetry: 0.64
Number of processed packets/flows: 38.37
Number of processed A packets/flows: 26.26
Number of processed B packets/flows: 101.46
Number of processed total packets/s: 239492.87 (239.49 K)
Number of processed A+B packets/s: 239492.87 (239.49 K)
Number of processed A packets/s: 137505.27 (137.50 K)
Number of processed B packets/s: 101987.60 (101.99 K)
=====
Number of average processed flows/s: 6242.47 (6.24 K)
Average full raw bandwidth: 1503099136 b/s (1.50 Gb/s)
Average snapped bandwidth : 1502563456 b/s (1.50 Gb/s)
Average full bandwidth : 1502764416 b/s (1.50 Gb/s)
Fill size of main hash map: 582178 [44.42%]
Max number of flows in memory: 582178 (582.18 K) [44.42%]
Memory usage: 5.61 GB [8.32%]
Aggregate flow status: 0x0c00b872d298fb04
[WRN] L3 SnapLength < Length in IP header
[WRN] Consecutive duplicate IP ID
[WRN] IPv4/6 payload length > framing length
[WRN] IPv4/6 fragmentation header packet missing
[INF] Ethernet flows
[INF] IPv4 flows
[INF] IPv6 flows
[INF] VLAN encapsulation
[INF] IPv4/6 fragmentation
[INF] MPLS encapsulation
[INF] L2TP encapsulation
[INF] PPP/HDLC encapsulation
[INF] GRE encapsulation
[INF] AYIYA tunnel
[INF] Teredo tunnel
[INF] CAPWAP/LWAPP tunnel
[INF] IPsec AH
[INF] IPsec ESP
[INF] SSDP/UPnP
[INF] SIP/RTP
=====

```

Listing 2: A sample Tranalyzer2 human readable report aggregate mode

Note at the beginning USR1 A type denotes that there was a signal received and all reporting is aggregated from the beginning of T2 operation, almost like the end report. Note, that the reaction to signals depends to internal core configuration, discussed in a later chapter below. For the time being we stick with the default. If you require now a report every 30s then type t2stat 30. Then a signal is sent to T2 every 30s, hence a remote control. Nevertheless, he can do it

also by himself, more later. Note that the report does not only tell you all about packet statistics but also the

1. pcap duration
2. Elapsed time
3. Total bytes to process
4. Percentage completed
5. Total bytes processed so far
6. Remaining time
7. Estimated Time Finish (ETF)

Especially the ETF is very valuable for large or multiple large pcap operations, e.g., multiple 10TB files. So the Anteater tells you when to come back from your coffee break.

[Listing 3](#) illustrates the output of the header line and subsequent data lines generated when MACHINE_REPORT=1.

%repTyp	time	sensorID	dur	memUsageKB	fillSzHashMap	numFlows	...
USR1MR_A	1022171702.125000	666	0.308953000	31686	2152	2152	...
USR1MR_A	1022171703.027000	666	1.308855000	33914	3919	3919	...
USR1MR_A	1022171704.334000	666	2.309162000	34750	5031	5031	...
USR1MR_A	1022171705.030000	666	3.308858000	35258	5847	5847	...

Listing 3: A sample Tranalyzer2 machine report aggregate mode

2.24.1 Configuration for Monitoring Mode on interface real time mode

To enable monitoring mode, configure Tranalyzer as follows:

main.h	tranalyzer.h
#define MONINTTMPCP 0	#define DIFF_REPORT 1
#define MONINTTHRD 1	#define MACHINE_REPORT 1

The following plugins contribute to the output:

- arpDecode
- basicStats
- cdpDecode
- connStat
- dnsDecode
- ftpDecode
- httpSniffer
- icmpDecode
- lldpDecode
- modbus
- mqttDecode
- ntpDecode
- radiusDecode
- sshDecode
- stpDecode
- t2PSkel
- tcpFlags
- vrrpDecode

The generated output is illustrated in Figure 3. The columns are as follows:

1. repType
2. sensorID
3. procID (t2 -i option and DPDK_MP=1 only)
4. time
5. duration
6. pktsRec (t2 -i option only)
7. pktsDrp (t2 -i option only)
8. ifDrp (t2 -i option and DPDK_MP=0 only)
9. pktsErr (t2 -i option and DPDK_MP=1 only)
10. bytesRec (t2 -i option and DPDK_MP=1 only)
11. memUsageKB
12. fillSzHashMap
13. numFlows
14. numAFlows
15. numBFlows
16. numPkts
17. numAPkts
18. numBPkts
19. numL2Pkts
20. numV4Pkts
21. numV6Pkts
22. numVxPkts
23. numBytes
24. numABytes
25. numBBytes
26. numFrgV4Pkts
27. numFrgV6Pkts
28. numAlarms
29. rawBandwidth
30. globalWarn
31. Layer 2 protocols stats (see MONPROTL2 in *main.h*):
 - 0x0806Pkts (ARP)
 - 0x0806Bytes (ARP)
 - 0x8035Pkts (RARP)
 - 0x8035Bytes (RARP)
32. Layer 3 protocols stats (see MONPROTL3 in *main.h*):
 - TCPPkts
 - TCPBytes
 - UDPPkts
 - UDPBytes
 - ICMPPkts
 - ICMPBytes
 - ICMPv6Pkts
 - ICMPv6Bytes
 - SCTPPkts
 - SCTPBytes

2.24.2 Monitoring Mode to Syslog

In order to send monitoring info to a syslog server T2 must be configured in machine mode as indicated above. Then the output has to be piped into the following script:

```
t2 -D ... -w ... | gawk -F"\t" '{ print "<25> ", strftime("%b %d %T"), "Monitoring: " $0 }' | \
nc -u w.x.y.z 514
```

Netcat will send it to the syslog server at address `w.x.y.z`. Specific columns from the monitoring output can be selected in the awk script.

2.24.3 RRD Graphing of Monitoring Output

The monitoring output can be stored in a RRD database using the `t2rrd` script. To start creating a RRD database, launch Tranalyzer2 (in monitoring mode) as follows:

```
t2 -r file.pcap -l | t2rrd -m
```

Or for monitoring from a live interface:

```
st2 -i eth0 -l | t2rrd -m
```

Plots for the various fields can then be generated using the same `t2rrd` script:

```
t2rrd -i 30s numAFlows numBFlows
```

To specify intervals, use `s` (seconds), `m` (minutes), `h` (hour), `d` (day), `w` (week), `mo` (month), `y` (year). For example, to plot the data from the last two weeks, use `-i 2w` or `-s -2w`.

An example graph is depicted in Figure 4.

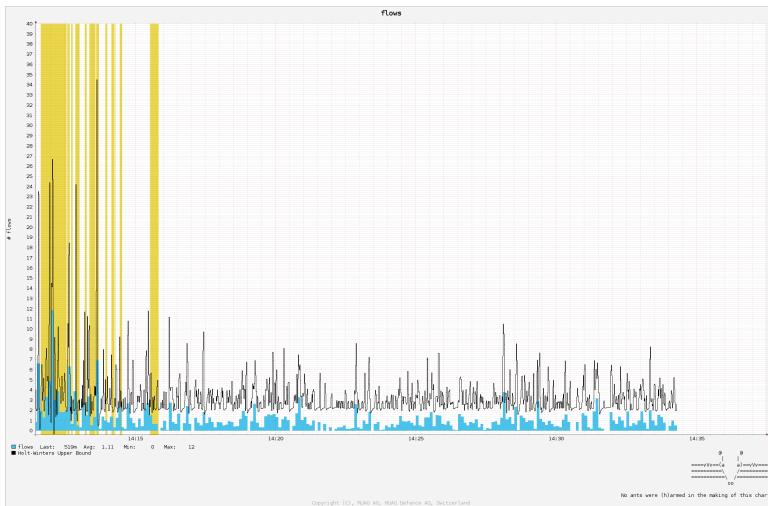


Figure 4: T2 monitoring using RRD

2.25 Cancellation of the Sniffing Process

Processing of a pcap file stops upon end of file. In case of live capture from an interface Tralyzer2 stops upon `CTRL+C` interrupt or a `kill -9` PID signal. The disconnection of the interface cable will stop Tralyzer2 also after a timeout of 182 seconds. The console based `CTRL+C` interrupt does not immediately terminate the program to avoid corrupted entries in the output files. It stops creating additional flows and finishes only currently active flows. Note that waiting the termination of active flow depends on the activity or the lifetime of a connection and can take a very long time. In order to mitigate that problem the user can issue the `CTRL+C` for `GI_TERM_THRESHOLD` times to immediately terminate the program.

3 arpDecode

3.1 Description

The arpDecode plugin analyzes ARP traffic.

3.2 Dependencies

3.2.1 Core Configuration

This plugin requires the following core configuration:

- `$T2HOME/tranalyzer2/src/networkHeaders.h`:
 - `ETH_ACTIVATE>0`

3.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
<code>ARP_MAX_IP</code>	10	Max. number of MAC/IP pairs to list [1 - 255]

3.4 Flow File Output

The arpDecode plugin outputs the following columns:

Column	Type	Description
<code>arpStat</code>	H8	Status
<code>arpHwType</code>	U16	Hardware type
<code>arpOpcode</code>	H16	Operational code
<code>arpIpMacCnt</code>	U16	Number of distinct MAC / IP pairs
<code>arpMac_Ip_Cnt</code>	R(MAC_IP4_U16)	MAC/IP pairs found and number of times the pair appeared (a count of zero may appear in case of ARP spoofing and indicate the pair was discovered in a different flow)

3.4.1 arpStat

The `arpStat` column is to be interpreted as follows:

arpStat	Description
0x01	ARP detected
0x02	Gratuitous ARP (sender IP same as target IP)
0x04	ARP Probe
0x08	ARP Announcement
0x10	—

arpStat	Description
0x20	MAC/IP list truncated... increase ARP_MAX_IP
0x40	—
0x80	ARP spoofing (same MAC assigned to multiple IPs)

3.4.2 arpHwType

The arpHwType column is to be interpreted as follows:

Type	Description
1	Ethernet
2	Experimental Ethernet
3	Amateur Radio AX.25
4	Proteon ProNET Token Ring
5	Chaos
6	IEEE 802
7	ARCNET
8	Hyperchannel
9	Lanstar
10	Autonet Short Address
11	LocalTalk
12	LocalNet (IBM PCNet or SYTEK LocalNET)
13	Ultra link
14	SMDS
15	Frame Relay
16	ATM (Asynchronous Transmission Mode)
17	HDLC
18	Fibre Channel

Type	Description
19	ATM (Asynchronous Transmission Mode)
20	Serial Line
21	ATM (Asynchronous Transmission Mode)
22	MIL-STD-188-220
23	Metricom
24	IEEE 1394.1995
25	MAPOS
26	Twinaxial
27	EUI-64
28	HIPARP
29	IP and ARP over ISO 7816-3
30	ARPSec
31	IPsec tunnel
32	Infiniband
33	CAI (TIA-102 Project 25 Common Air Interface)
34	Wiegand Interface
35	Pure IP

3.4.3 arpOpcode

The arpOpcode column is to be interpreted as follows:

arpOpcode	Description
2 ⁰ (=0x0001)	—
2 ¹ (=0x0002)	ARP Request
2 ² (=0x0004)	ARP Reply
2 ³ (=0x0008)	Reverse ARP (RARP) Request
2 ⁴ (=0x0010)	Reverse ARP (RARP) Reply
2 ⁵ (=0x0020)	Dynamic RARP (DRARP) Request
2 ⁶ (=0x0040)	Dynamic RARP (DRARP) Reply
2 ⁷ (=0x0080)	Dynamic RARP (DRARP) Error

arpOpcode	Description
2 ⁸ (=0x0100)	Inverse ARP (InARP) Request
2 ⁹ (=0x0200)	Inverse ARP (InARP) Reply
2 ¹⁰ (=0x0400)	ARP NAK
2 ¹¹ (=0x0800)	—
2 ¹² (=0x1000)	—
2 ¹³ (=0x2000)	—
2 ¹⁴ (=0x4000)	—
2 ¹⁵ (=0x8000)	—

3.5 Packet File Output

In packet mode (-s option), the arpDecode plugin outputs the following columns:

Column	Type	Description
arpStat	H8	Status
arpHwType	U16	Hardware type
arpProtoType	H16	Protocol type
arpHwSize	U8	Hardware size
arpProtoSize	U8	Protocol size
arpOpcode	U16	Operational code
arpSenderMAC	MAC	Sender MAC address
arpSenderIP	IP4	Sender IP address
arpTargetMAC	MAC	Target MAC address
arpTargetIP	IP4	Target IP address

3.6 Monitoring Output

In monitoring mode, the arpDecode plugin outputs the following columns:

Column	Type	Description
arpStat	H8	Status

3.7 Plugin Report Output

The following information is reported:

- Aggregated arpStat

4 basicFlow

4.1 Description

The basicFlow plugin provides host identification fields and timing information.

4.2 Configuration Flags

4.2.1 basicFlow.h

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
BFO_SENSORID	0	Output sensorID	
BFO_HDRDESC_PKTCNT	0	Include packet count for header description	
BFO_MAC	1	Output MAC addresses	
BFO_ETHERTYPE	1	Output EtherType	IPV6_ACTIVATE=2 ETH_ACTIVATE>0
BFO_VLAN	1	0: Do not output VLAN information, 1: Output VLAN numbers, 2: Output VLAN headers as hex 3: Output decoded VLAN headers as TPID_PCP_DEI_VID	
BFO_MPLS	0	0: Do not output MPLS information, 1: Output MPLS labels, 2: Output MPLS labels as hex, 3: Output MPLS headers as hex, 4: Output decoded MPLS headers	
BFO_GRE	0	Enable GRE output	
BFO_L2TP	0	Enable L2TP output	
BFO PPP	0	Enable PPP output	
BFO LAPD	0	Enable LAPD output	LAPD_ACTIVATE=1
BFO_TEREDO	0	Enable Teredo output	
BFO_SUBNET_IPLIST	0	Display a list of IP aggregated	
BFO_SUBNET_TEST	1	Enable subnet test	
BFO_SUBNET_TEST_GRE	0	Enable subnet test on GRE addresses	IPV6_ACTIVATE!=1
BFO_SUBNET_TEST_L2TP	0	Enable subnet test on L2TP addresses	IPV6_ACTIVATE!=1
BFO_SUBNET_TEST_TEREDO	0	Enable subnet test on Teredo addresses	
BFO_SUBNET_HEX	0	Output the country code and organization as one 32-bit hex number	
BFO_SUBNET ASN	0	Output Autonomous System Numbers (ASN)	

Name	Default	Description	Flags
BFO_SUBNET_LL	0	Output position (latitude, longitude and reliability)	
BFO_MAX_HDRDESC	4	Max. number of headers descriptions to store	T2_PRI_HDRDESC=1
BFO_MAX_MAC	2	Max. different MAC addresses to store	
BFO_MAX_IP	2	Max. different IP addresses to store	
BFO_MAX MPLS	3	Max. MPLS headers/tags to store	
BFO_MAX_VLAN	3	Max. VLAN headers/numbers to store	

4.2.2 bin2txt.h

Additional configuration options can be found in \$T2HOME/utils/bin2txt.h. Refer to [Tranalyzer2](#) documentation for more details.

4.2.3 subnetHL.h

Additional configuration options can be found in \$T2HOME/utils/subnetHL.h. Refer to [Tranalyzer2](#) documentation for more details.

4.3 Flow File Output

The basicFlow plugin outputs the following columns:

Column	Type	Description	Flags
dir	C	Flow direction A / B	
flowInd	U64	Flow index	
sensorID	U32	Sensor ID	BFO_SENSORID=1
flowStat	H64	Flow status and warnings	
timeFirst	TS	Date time of first packet	
timeLast	TS	Date time of last packet	
duration	U64.U32	Flow duration	

If T2_PRI_HDRDESC=1 and BFO_MAX_HDRDESC>0, the following columns are displayed:

numHdrDesc	U8	Number of different headers descriptions	
numHdrs	R(U16)	Number of headers (depth) in <code>hdrDesc</code>	
hdrDesc	RS	Headers description	BFO_HDRDESC_PKT_CNT=0
hdrDesc_pktCnt	R(S_U64)	Headers description and packet count	BFO_HDRDESC_PKT_CNT=1
srcMac	R(MAC)	Source MAC address	BFO_MAC=1
dstMac	R(MAC)	Destination MAC address	BFO_MAC=1
ethType	H16	Ethernet type	BFO_ETHERTYPE=1 && (ETH_ACTIVATE>0 IPV6_ACTIVATE=2)

Column	Type	Description	Flags
If <code>BFO_VLAN>0</code> and <code>BFO_MAX_VLAN>0</code> , the columns described in Section 4.3.1 are displayed here.			
If <code>BFO_MPLS>0</code> and <code>BFO_MAX_MPLS>0</code> , the columns described in Section 4.3.2 are displayed here.			
If <code>BFO_PPP=1</code> , the columns described in Section 4.3.3 are displayed here.			
If <code>LAPD_ACTIVATE=1</code> and <code>BFO_LAPD=1</code> , the columns described in Section 4.3.4 are displayed here.			
If <code>BFO_L2TP=1</code> , the columns described in Section 4.3.5 are displayed here.			
If <code>BFO_GRE=1</code> , the columns described in Section 4.3.6 are displayed here.			
If <code>BFO_TEREDO=1</code> , the columns described in Section 4.3.7 are displayed here.			

Standard five tuple output including geo-information

<code>srcIP</code>	IP	Source IP address	<code>BFO_SUBNET_IPLIST=0</code>
<code>srcIP</code>	R(IP)	Source IP addresses	<code>BFO_SUBNET_IPLIST=1</code>

If `BFO_SUBNET_TEST=1`, the following columns are displayed:

<code>srcIPASN</code>	U32	Source IP ASN	<code>BFO_SUBNET ASN=1</code>
<code>srcIPCOC</code>	H32	Source IP country organization code	<code>BFO_SUBNET HEX=1</code>
<code>srcIPCC</code>	SC	Source IP country	
<code>srcIPCnty</code>	S	Source IP county	<code>CNTYCTY=1</code>
<code>srcIPCity</code>	S	Source IP city	<code>CNTYCTY=1</code>
<code>srcIPOrg</code>	S	Source IP organization	<code>BFO_SUBNET ORG=1</code>
<code>srcIPLat_Lng_relP</code>	F_F_F	Source IP lat., long. and reliability	<code>BFO_SUBNET LL=1</code>

`srcPort` U16 Source Port

<code>dstIP</code>	IP	Destination IP address	<code>BFO_SUBNET_IPLIST=0</code>
<code>dstIP</code>	R(IP)	Destination IP addresses	<code>BFO_SUBNET_IPLIST=1</code>

If `BFO_SUBNET_TEST=1`, the following columns are displayed:

<code>dstIPASN</code>	U32	Dest. IP ASN	<code>BFO_SUBNET ASN=1</code>
<code>dstIPCOC</code>	H32	Dest. IP country organization code	<code>BFO_SUBNET HEX=1</code>
<code>dstIPCC</code>	SC	Dest. IP country	
<code>dstIPCnty</code>	S	Dest. IP county	<code>CNTYCTY=1</code>
<code>dstIPCity</code>	S	Dest. IP city	<code>CNTYCTY=1</code>
<code>dstIPOrg</code>	S	Dest. IP organization	<code>BFO_SUBNET ORG=1</code>

Column	Type	Description	Flags
dstIPLat_Lng_relP	F_F_F	Dest. IP lat., long. and reliability	BFO_SUBNET_LL=1
dstPort	U16	Destination port	
l4Proto	U8	Layer4 protocol	

4.3.1 VLAN

If BFO_VLAN>0 and BFO_MAX_VLAN>0, the following additional column is displayed:

Column	Type	Description	Flags
vlanID	R(U16)	VLAN IDs	BFO_VLAN=1
vlanHdr	R(H32)	VLAN headers (hex)	BFO_VLAN=2
vlanTPID_	R(H16_	VLAN tag protocol identifier (TPID),	BFO_VLAN=3
PCP_	U8_	priority code point (PCP),	
DEI_	U8_	drop eligible indicator (DEI),	
VID	U16)	VLAN identifier (VID)	

4.3.2 MPLS

If BFO_MPLS>0 and BFO_MAX_MPLS>0, the following additional column is displayed:

Column	Type	Description	Flags
mplsLabels	R(U32)	MPLS labels	BFO_MPLS=1
mplsLabelsHex	R(H32)	MPLS labels (hex)	BFO_MPLS=2
mplsHdrsHex	R(H32)	MPLS headers (hex)	BFO_MPLS=3
mplsLabel_ToS_S_TTL	R(U32_U8_U8_U8)	MPLS headers details	BFO_MPLS=4

4.3.3 PPP

If BFO_PPP=1, the following additional column is displayed:

Column	Type	Description	Flags
pppHdr	H32	PPP header	

4.3.4 LAPD

If BFO_LAPD=1 and LAPD_ACTIVATE=1, the following additional column is displayed:

Column	Type	Description	Flags
lapdSAPI	U8	LAPD SAPI	
lapdTEI	U8	LAPD TEI	

4.3.5 L2TP

If BFO_L2TP=1, the following additional columns are displayed:

Column	Type	Description	Flags
l2tpHdr	H16	L2TP header	
l2tpTID	U16	L2TP tunnel ID	
l2tpSID	U16	L2TP session ID	
l2tpCCSID	U32	L2TP control connection/session ID	
l2tpSrcIP	IP4	L2TP source IP address	

If BFO_SUBNET_TEST_L2TP=1, the following columns are displayed:

l2tpSrcIPASN	U32	L2TP source IP ASN	BFO_SUBNET ASN=1
l2tpSrcIPCOC	H32	L2TP source IP country organization code	BFO_SUBNET HEX=1
l2tpSrcIPCC	SC	L2TP source IP country	
l2tpSrcIPCnty	S	L2TP source IP county	CNTYCTY=1
l2tpSrcIPCity	S	L2TP source IP city	CNTYCTY=1
l2tpSrcIPOrg	S	L2TP source IP organization	BFO_SUBNET ORG=1
l2tpSrcIPLat_Lng_relP	F_F_F	L2TP source IP lat., long. and reliability	BFO_SUBNET LL=1
l2tpDstIP	IP4	L2TP destination IP address	

If BFO_SUBNET_TEST_L2TP=1, the following columns are displayed:

l2tpDstIPASN	U32	L2TP dest. IP ASN	BFO_SUBNET ASN=1
l2tpDstIPCOC	H32	L2TP dest. IP country organization code	BFO_SUBNET HEX=1
l2tpDstIPCC	SC	L2TP dest. IP country	
l2tpDstIPCnty	S	L2TP dest. IP county	CNTYCTY=1
l2tpDstIPCity	S	L2TP dest. IP city	CNTYCTY=1
l2tpDstIPOrg	S	L2TP dest. IP organization	BFO_SUBNET ORG=1
l2tpDstIPLat_Lng_relP	F_F_F	L2TP dest. IP lat., long. and reliability	BFO_SUBNET LL=1

4.3.6 GRE

If BFO_GRE=1, the following additional columns are displayed:

Column	Type	Description	Flags
greHdr	H32	GRE header	
greSrcIP	IP4	GRE source IP address	

If BFO_SUBNET_TEST_GRE=1, the following columns are displayed:

greSrcIPASN	U32	GRE source IP ASN	BFO_SUBNET ASN=1
greSrcIPCOC	H32	GRE source IP country organization code	BFO_SUBNET HEX=1
greSrcIPCC	SC	GRE source IP country	
greSrcIPCnty	S	GRE source IP county	CNTYCTY=1

Column	Type	Description	Flags
greSrcIPCity	S	GRE source IP city	CNTYCTY=1
greSrcIPOrg	S	GRE source IP organization	BFO_SUBNET_ORG=1
greSrcIPLat_Lng_relP	F_F_F	GRE source IP lat., long. and reliability	BFO_SUBNET_LL=1
greDstIP	IP4	GRE destination IP address	
If BFO_SUBNET_TEST_GRE=1, the following columns are displayed:			
greDstIPASN	U32	GRE dest. IP ASN	BFO_SUBNET ASN=1
greDstIPCOC	H32	GRE dest. IP country organization code	BFO_SUBNET_HEX=1
greDstIPCC	SC	GRE dest. IP country	
greDstIPCnty	S	GRE dest. IP county	CNTYCTY=1
greDstIPCity	S	GRE dest. IP city	CNTYCTY=1
greDstIPOrg	S	GRE dest. IP organization	BFO_SUBNET_ORG=1
greDstIPLat_Lng_relP	F_F_F	GRE dest. IP lat., long. and reliability	BFO_SUBNET_LL=1

4.3.7 Teredo

If BFO_TEREDO=1, the following additional columns are displayed:

Column	Type	Description	Flags
trdoDstIP	IP4	Next Teredo flow: dest IPv4 address	

If BFO_SUBNET_TEST_TEREDO=1, the following columns are displayed:

trdoDstIPASN	U32	Teredo dest. IP ASN	BFO_SUBNET ASN=1
trdoDstIPCOC	H32	Teredo dest. IP country organization code	BFO_SUBNET HEX=1
trdoDstIPCC	SC	Teredo dest. IP country	
trdoDstIPCnty	S	Teredo dest. IP county	CNTYCTY=1
trdoDstIPCity	S	Teredo dest. IP city	CNTYCTY=1
trdoDstIPOrg	S	Teredo dest. IP organization	BFO_SUBNET ORG=1
trdoDstIPLat_Lng_relP	F_F_F	Teredo dest. IP lat., long. and reliability	BFO_SUBNET LL=1
trdoDstPort	U16	Next Teredo flow: destination port	

If IPV6_ACTIVATE=0, then no further column is displayed.

Teredo IPv6 source address decode

trdo6SrcFlgs	H8	Flags	
trdo6SrcSrvIP4	IP4	Server IPv4	

If BFO_SUBNET_TEST_TEREDO=1, the following columns are displayed:

trdo6SrcSrvIP4ASN	U32	Server IP ASN	BFO_SUBNET ASN=1
trdo6SrcSrvIP4COC	H32	Server IP country organization code	BFO_SUBNET HEX=1
trdo6SrcSrvIP4CC	SC	Server IP country	
trdo6SrcSrvIP4Cnty	S	Server IP county	CNTYCTY=1
trdo6SrcSrvIP4City	S	Server IP city	CNTYCTY=1
trdo6SrcSrvIP4Org	S	Server IP organization	BFO_SUBNET ORG=1
trdo6SrcSrvIP4Lat_Lng_relP	F_F_F	Server IP lat., long. and reliability	BFO_SUBNET TEST LL=1
trdo6SrcCPIP4	IP4	Client public IPv4	

If BFO_SUBNET_TEST_TEREDO=1, the following columns are displayed:

trdo6SrcCPIP4ASN	U32	Client public IP ASN	BFO_SUBNET ASN=1
trdo6SrcCPIP4COC	H32	Client public IP country organization code	BFO_SUBNET HEX=1
trdo6SrcCPIP4CC	SC	Client public IP country	
trdo6SrcCPIP4Cnty	S	Client public IP county	CNTYCTY=1
trdo6SrcCPIP4City	S	Client public IP city	CNTYCTY=1
trdo6SrcCPIP4Org	S	Client public IP organization	BFO_SUBNET ORG=1
trdo6SrcCPIP4Lat_Lng_relP	F_F_F	Client public IP lat., long. and reliability	BFO_SUBNET LL=1

Column	Type	Description	Flags
trdo6SrcCPPort	U16	Client public port	
Teredo IPv6 destination address decode			
trdo6DstFlgs	H8	Flags	
trdo6DstSrvIP4	IP4	Server IPv4	
If BFO_SUBNET_TEST_TEREDO=1, the following columns are displayed:			
trdo6DstSrvIP4ASN	U32	Server IP ASN	BFO_SUBNET ASN=1
trdo6DstSrvIP4COC	H32	Server IP country organization code	BFO_SUBNET_HEX=1
trdo6DstSrvIP4CC	SC	Server IP country	
trdo6DstSrvIP4Cnty	S	Server IP county	CNTYCTY=1
trdo6DstSrvIP4City	S	Server IP city	CNTYCTY=1
trdo6DstSrvIP4Org	S	Server IP organization	BFO_SUBNET_ORG=1
trdo6DstSrvIP4Lat_Lng_relP	F_F_F	Server IP lat., long. and reliability	BFO_SUBNET_LL=1
trdo6DstCPIP4	IP4	Client public IPv4	
If BFO_SUBNET_TEST_TEREDO=1, the following columns are displayed:			
trdo6DstCPIP4ASN	U32	Client public IP ASN	BFO_SUBNET ASN=1
trdo6DstCPIP4COC	H32	Client public IP country organization code	BFO_SUBNET_HEX=1
trdo6DstCPIP4CC	SC	Client public IP country	
trdo6DstCPIP4Cnty	S	Client public IP county	CNTYCTY=1
trdo6DstCPIP4City	S	Client public IP city	CNTYCTY=1
trdo6DstCPIP4Org	S	Client public IP organization	BFO_SUBNET_ORG=1
trdo6DstCPIP4Lat_Lng_relP	F_F_F	Client public IP lat., long. and reliability	BFO_SUBNET_LL=1
trdo6DstCPPort	U16	Client public port	

4.3.8 flowInd

It is useful to identify flows when post processing operations, such as sort or filters are applied to a flow file and only a B or an A flow is selected. Moreover a packet file generated with the -s option supplies the flow index which simplifies the mapping of singular packets to the appropriate flow.

4.3.9 flowStat

The flowStat column is to be interpreted as follows:

flowStat	Description
2 ⁰⁰ (=0x00000000 00000001)	Inverted flow, did not initiate connection
2 ⁰¹ (=0x00000000 00000002)	No Ethernet header

	flowStat	Description
2 ⁰²	(=0x00000000 00000004)	Pure L2 flow
2 ⁰³	(=0x00000000 00000008)	Point to Point Protocol over Ethernet Discovery (PPPoED)
2 ⁰⁴	(=0x00000000 00000010)	Point to Point Protocol over Ethernet Service (PPPoES)
2 ⁰⁵	(=0x00000000 00000020)	Link Layer Discovery Protocol (LLDP)
2 ⁰⁶	(=0x00000000 00000040)	ARP
2 ⁰⁷	(=0x00000000 00000080)	Reverse ARP
2 ⁰⁸	(=0x00000000 00000100)	VLANs
2 ⁰⁹	(=0x00000000 00000200)	MPLS unicast
2 ¹⁰	(=0x00000000 00000400)	MPLS multicast
2 ¹¹	(=0x00000000 00000800)	L2TP v2/3
2 ¹²	(=0x00000000 00001000)	GRE v1/2
2 ¹³	(=0x00000000 00002000)	PPP header after L2TP or GRE
2 ¹⁴	(=0x00000000 00004000)	IPv4 flow
2 ¹⁵	(=0x00000000 00008000)	IPv6 flow
2 ¹⁶	(=0x00000000 00010000)	IPvX bogus packet
2 ¹⁷	(=0x00000000 00020000)	IPv4/6 in IPv4/6
2 ¹⁸	(=0x00000000 00040000)	Ethernet over IP
2 ¹⁹	(=0x00000000 00080000)	Teredo tunnel
2 ²⁰	(=0x00000000 00100000)	Anything in Anything (AYIYA) tunnel
2 ²¹	(=0x00000000 00200000)	GPRS Tunneling Protocol (GTP)
2 ²²	(=0x00000000 00400000)	Virtual eXtensible Local Area Network (VXLAN)
2 ²³	(=0x00000000 00800000)	Control and Provisioning of Wireless Access Points (CAPWAP), Lightweight Access Point Protocol (LWAPP)
2 ²⁴	(=0x00000000 01000000)	Stream Control Transmission Protocol (SCTP)
2 ²⁵	(=0x00000000 02000000)	SSDP/UPnP
2 ²⁶	(=0x00000000 04000000)	Encapsulated Remote Switch Packet ANalysis (ERSPAN)
2 ²⁷	(=0x00000000 08000000)	Cisco Web Cache Communication Protocol (WCCP)
2 ²⁸	(=0x00000000 10000000)	SIP/RTP
2 ²⁹	(=0x00000000 20000000)	Generic Network Virtualization Encapsulation (GENEVE)
2 ³⁰	(=0x00000000 40000000)	IPsec Authentication Header (AH)
2 ³¹	(=0x00000000 80000000)	IPsec Encapsulating Security Payload (ESP)
2 ³²	(=0x00000001 00000000)	Acquired packet length < minimal L2 datagram
2 ³³	(=0x00000002 00000000)	Acquired packet length < packet length in L3 header
2 ³⁴	(=0x00000004 00000000)	Acquired packet length < minimal L3 header
2 ³⁵	(=0x00000008 00000000)	Acquired packet length < minimal L4 header
2 ³⁶	(=0x00000010 00000000)	IPv4 fragmentation present
2 ³⁷	(=0x00000020 00000000)	IPv4 fragmentation error (refer to the <code>tcpFlags</code> plugin for more details)

flowStat	Description
2^{38} ($=0x000000040$ 00000000)	IPv4 1. fragment out of sequence or missing
2^{39} ($=0x000000080$ 00000000)	Packet fragmentation pending or Fragmentation sequence not completed when flow timed-out
2^{40} ($=0x00000100$ 00000000)	Flow timeout instead of protocol termination
2^{41} ($=0x00000200$ 00000000)	Alarm mode: remove this flow instantly
2^{42} ($=0x00000400$ 00000000)	Autopilot: flow removed to free space in main hash map
2^{43} ($=0x00000800$ 00000000)	Stop dissecting, error or unhandled protocol
2^{44} ($=0x00001000$ 00000000)	Consecutive duplicate IP ID
2^{45} ($=0x00002000$ 00000000)	PPPL3 header not readable, compressed
2^{46} ($=0x00004000$ 00000000)	IPv4 header length < 20 bytes
2^{47} ($=0x00008000$ 00000000)	IP payload length > framing length
2^{48} ($=0x00010000$ 00000000)	Header description overrun
2^{49} ($=0x00020000$ 00000000)	pcapd and PD_ALARM=1: if set dumps the packets from this flow to a new pcap
2^{50} ($=0x00040000$ 00000000)	Land attack: same srcIP && dstIP && srcPort && dstPort
2^{51} ($=0x00080000$ 00000000)	Timestamp jump, probably due to multi-path delay or NTP operation
2^{52} ($=0x00100000$ 00000000)	RESERVED, do not use
2^{55} ($=0x00800000$ 00000000)	Subnet tested for that flow
2^{56} ($=0x01000000$ 00000000)	Tor address detected
2^{57} ($=0x02000000$ 00000000)	A packet had a priority tag (VLAN tag with ID 0)
2^{58} ($=0x04000000$ 00000000)	IPv4 packet
2^{59} ($=0x08000000$ 00000000)	IPv6 packet
2^{62} ($=0x40000000$ 00000000)	Flow duration limit, same findex for all subflows
2^{63} ($=0x80000000$ 00000000)	PCAP packet length > MAX_MTU in <i>ioBuffer.h</i> , caplen reduced

4.3.10 hdrDesc

The `hdrDesc` column describes the protocol stack in the flow in a human readable way. Note that it gives the user a lookahead of what is to be expected, even if not in the appropriate IPv4/6 mode. For example, in IPv4 several different headers stacks can be displayed by one flow if Teredo or different fragmentation is involved. T2 then dissects only to the last header above the said protocol and sets the *Stop dissecting* bit in the flow status (2^{41} ($=0x00000400$ 00000000)).

4.3.11 lapdFType

The `lapdFType` column is to be interpreted as follows:

lapdFType	Description
0	Information frame
1	Supervisory frame
3	Unnumbered frame

4.3.12 lapdFunc

The `lapdFunc` column is to be interpreted as follows:

lapdFunc	Description
REJ	Supervisory frame – REject
RNR	Supervisory frame – Receive Not Ready
RR	Supervisory frame – Receive Ready
CFGR	Unnumbered frame – ConFiGuRe
DISC	Unnumbered frame – DISConnect
DM	Unnumbered frame – Disconnected Mode
FRMR	Unnumbered frame – FRaMe Reject
SABME	Unnumbered frame – Set Asynchronous Balanced Mode Extended
SIM	Unnumbered frame – Set Initialization Mode
TEST	Unnumbered frame – Test
UA	Unnumbered frame – Unnumbered Acknowledgement
UI	Unnumbered frame – Unnumbered Information
UP	Unnumbered frame – Unnumbered Poll
XID	Unnumbered frame – eXchange IDentification

4.3.13 trdo6SrcFlgs and trdo6DstFlgs

The `trdo6SrcFlgs` and `trdo6DstFlgs` columns are to be interpreted as follows:

trdo6{Src,Dst}Flgs	Description
2^0 (=0x01)	Group/individual
2^1 (=0x02)	Universal/local
2^2 (=0x04)	—
2^3 (=0x08)	—
2^4 (=0x10)	—
2^5 (=0x20)	—
2^6 (=0x40)	Currently unassigned
2^7 (=0x80)	Behind NAT, new versions do not set this bit anymore

4.3.14 Geo labeling

The country and organization coding scheme are defined in the following files:

- `utils/subnet/whoCntryCds.txt`
- `utils/subnet/whoOrgCds.txt`

The special country code (CC) values [0-9] [0-9] are used to represent private addresses or special address ranges such as Teredo or multicast:

CC	IPv4 addresses	IPv6 addresses	Description
00	0.0.0.0/32	::/128	Unspecified address
01	127.0.0.0/8	::1/128	Loopback address
02	169.254.0.0/16	fe80::/10	Link-local address
03		fc00::/7	Unique local address
04	10.0.0.0/8		Private network
05	172.16.0.0/12		Private network
06	192.0.0.0/24		Private network
07	192.168.0.0/16		Private network
08	198.18.0.0/15		Private network
10	224.0.0.0/4	ff00::/8	Multicast
11	255.255.255.255/32		Broadcast
20	100.64.0.0/10		Shared address space
21	192.0.2.0/24		TEST-NET-1
22	198.51.100.0/16		TEST-NET-2
23	203.0.113.0/24		TEST-NET-3
24	240.0.0.0/4		Reserved
25		100::/64	Discard prefix
26		2001:20::/28	ORCHIDv2
27		2001:db8::/32	Address used in documentation and example source code
60	192.88.99.0/24		Reserved (formerly used for IPv6 to IPv4 relay)
61		::ffff:0:0/96	IPv4 mapped address
62		::ffff:0:0:0/96	IPv4 translated address
63		64:ff9b::/96	IPv4/IPv6 translation
64		2001::/32	Teredo tunneling
65		2002::/16	The 6to4 addressing scheme (now deprecated)

The country organization codes (BFO_SUBNET_HEX=1) can be decoded with t2netID, e.g., t2netID 0x138020a5. Try t2netID --help for more information.

4.4 Packet File Output

In packet mode (-s option), the basicFlow plugin outputs the following columns:

Column	Type	Description	Flags
flowInd	U64	Flow index	
flowStat	H64	Flow status	
time	TS	Date time of packet	
relTime	U64.U32	Duration since start of pcap or interface sniffing	RELTIME=1
pktIAT	F	Packet inter-arrival time	
pktTrip	F	Packet round-trip time	
flowDuration	F	Flow duration	
numHdrs	U16	Number of headers (depth) in hdrDesc	T2_PRI_HDRDESC=1
hdrDesc	S	Headers description	T2_PRI_HDRDESC=1
vlanTPID_	H16_	VLAN tag protocol identifier (TPID),	BFO_VLAN=3
PCP_	U8_	priority code point (PCP),	

Column	Type	Description	Flags
DEI_	U8_	drop eligible indicator (DEI),	
VID	U16	VLAN identifier (VID)	
vlanHdr	H32	VLAN headers	BFO_VLAN=2
vlanID	U16	VLAN numbers	BFO_VLAN=1

If BFO_MPLS>0 and BFO_MAX_MPLS>0, one of the following column is displayed:

mplsLabels	R(U32)	MPLS labels	BFO_MPLS=1
mplsLabelsHex	R(H32)	MPLS labels (hex)	BFO_MPLS=2
mplsHdrsHex	R(H32)	MPLS headers (hex)	BFO_MPLS=3
mplsLabel_ToS_S_TTL	R(U32_U8_U8_U8)	MPLS headers details	BFO_MPLS=4
srcMac	MAC	Source MAC address	
dstMac	MAC	Destination MAC address	
ethType	H16	Ethernet type	

If LAPD_ACTIVATE=1 and BFO_LAPD=1, the following six columns are displayed:

lapdSAPI	U8	LAPD SAPI	
lapdTEI	U8	LAPD TEI	
lapdFType	U8	LAPD frame type	
lapdFunc	S	LAPD command (U-Frame) or Supervisory frame type	
lapdNR	U8	LAPD Receive Sequence Number	
lapdNS	U8	LAPD Send Sequence Number	
srcIP	IP	Source IP address	
srcIPCC	SC	Source IP country	BFO_SUBNET_TEST=1
srcIPOrg	S	Source IP organization	BFO_SUBNET_TEST=1&& BFO_SUBNET_ORG=1
srcPort	U16	Source port	
dstIP	IP	Destination IP address	
dstIPCC	SC	Destination IP country	BFO_SUBNET_TEST=1
dstIPOrg	S	Destination IP organization	BFO_SUBNET_TEST=1&& BFO_SUBNET_ORG=1
dstPort	U16	Destination port	
l4Proto	U8	Layer 4 protocol	

4.5 Post-Processing

The t2whois program provides an offline whois and geolocation query option using T2 subnet files. It can be found in \$T2HOME/utils/t2whois/ and can be compiled by typing make. The use of the program is straightforward:

```
t2whois 1.2.3.4
```

Try t2whois -h for more information.

5 basicStats

5.1 Description

The basicStats plugin supplies basic layer four statistics for each flow.

5.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
BS_AGGR_CNT	0	0: A+B counts off, 0 1: add A+B counts	
BS_REV_CNT	1	0: native send counts, 1: add reverse counts from opposite flow	
BS_MOD	0	> 1: modulo factor of packet length; else: off	
BS_PAD	1	1: Aggregated padding bytes	
BS_STATS	1	Output statistics (min, max, average, ...)	
BS_PL_STATS	1	1: Packet Length statistics	
BS_IAT_STATS	1	1: IAT statistics	

If BS_STATS==1, the following additional flags can be used:

BS_VAR	0	Output the variance	
BS_STDDEV	1	Output the standard deviation	
BS_SK	0	Skew/kurtosis calculation	BS_VAR=1
BS_XCLD	0	0: do not exclude any value from statistics, 1: include (BS_XMIN, UINT16_MAX), 2: include [0, BS_XMAX), 3: include [BS_XMIN, BS_XMAX], 4: exclude (BS_XMIN, BS_XMAX)	
BS_XMIN	1	minimal included/excluded from statistics	BS_XCLD>0
BS_XMAX	UINT16_MAX	maximal included/excluded from statistics	BS_XCLD>0

5.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- BS_XMIN
- BS_XMAX

5.3 Flow File Output

The basicStats plugin outputs the following fields:

Column	Type	Description	Flags
pktsSnt	U64	Number of transmitted packets	
pktsRcvd	U64	Number of received packets	BS_REV_CNT=1
pktsRTAggr	U64	Number of received + transmitted packets	BS_AGGR_CNT=1
padBytesSnt	I64	Number of transmitted padding bytes	BS_PAD=1
12-7BytesSnt	U64	Number of transmitted layer 2-7 bytes	
12-7BytesRcvd	U64	Number of received layer 2-7 bytes	BS_REV_CNT=1
12-7BytesRTAggr	U64	Number of received + transmitted layer 2-7 bytes	BS_AGGR_CNT=1

If BS_STATS=1, the following columns, whose value depends on BS_XCLD, are provided

If BS_PL_STATS=1, the following five columns are displayed

minL2-7PktSz	U16	Minimum layer 2-7 packet size	
maxL2-7PktSz	U16	Maximum layer 2-7 packet size	
avgL2-7PktSz	F	Average layer 2-7 packet size	
stdL2-7PktSz	F	Standard deviation layer 2-7 packet size	BS_STDDEV=1
varL2-7PktSz	F	Variance layer 2-7 packet size	BS_VAR=1
skewL2-7PktSz	F	Skewness layer 2-7 packet size	BS_SK=1 && BS_VAR=1
kurL2-7PktSz	F	Kurtosis layer 2-7 packet size	BS_SK=1 && BS_VAR=1

If BS_IAT_STATS=1, the following five columns are displayed

miniIAT	F	Minimum IAT	
maxIAT	F	Maximum IAT	
avgIAT	F	Average IAT	
variIAT	F	Variance IAT	BS_VAR=1
stdIAT	F	Standard deviation IAT	BS_STDDEV=1
skewIAT	F	Skewness IAT	BS_SK=1 && BS_VAR=1
kuriIAT	F	Kurtosis IAT	BS_SK=1 && BS_VAR=1
pktps	F	Sent packets per second	
bytps	F	Sent bytes per second	
pktAsm	F	Packet stream asymmetry	
bytAsm	F	Byte stream asymmetry	

5.3.1 Packet size

Packet sizes depend on PACKETLENGTH in *tanalyzert2/src/packetCapture.h*):

- PACKETLENGTH=0:
 - Output *L2PktSz and *L2Bytes* columns
 - Packet size include layer 2, 3 and 4 header
- PACKETLENGTH=1:
 - Output *L3PktSz and *L3Bytes* columns
 - Packet size include layer 3 and 4 header
- PACKETLENGTH=2:
 - Output *L4PktSz and *L4Bytes* columns
 - Packet size include layer 4 header
- PACKETLENGTH=3:
 - Output *L7PktSz and *L7Bytes* columns
 - Packet size include only layer 7

5.4 Packet File Output

In packet mode (-s option), the basicStats plugin outputs the following columns:

Column	Type	Description	Flags
pktLen	U32	Packet size on the wire	
udpLen	U16	Length in UDP/UDP-Lite header	
snapL4Len	U16	Snapped layer 4 length	
snapL7Len	U16	Snapped layer 7 length	
l7Len	U16	Layer 7 length	
pktLenMod	U16	Modulo factor of packet length	BS_MOD>1
padLen	I64	Number of padding bytes	

5.5 Plugin Report Output

The following information is reported:

- MAC of biggest packets/bytes talker and packets/bytes counts
- IP of biggest packets/bytes talker and packets/bytes counts (ANONYM_IP=0)

6 bayesClassifier

6.1 Description

The bayesClassifier plugin classifies flows based on different models and different source plugins.

6.2 Dependencies

6.2.1 External Libraries

This plugin depends on the **jansson** library.

Ubuntu:	sudo apt-get install libjansson-dev
Arch:	sudo pacman -S jansson
Gentoo:	sudo emerge jansson
openSUSE:	sudo zypper install libjansson
Red Hat/Fedora⁵:	sudo dnf install jansson
macOS⁶:	brew install jansson

6.2.2 Other Plugins

This plugin requires the [nFrstPkts](#) plugin.

6.2.3 Required Files

The file `bayes_config.json` is required.

6.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
BAYES_CFG	"bayes_config.cfg"	Name of the Bayes config file
BAYES_UNKNOWN	"unknown"	Default class name
BAYES_MIN_POST_PROB	0.0001	Minimum post-probability
BAYES_MIN_NUM_PKT	1	Minimum amount of packets per flow

6.4 Flow File Output

The bayesClassifier plugin outputs the following columns:

⁵If the `dnf` command could not be found, try with `yum` instead

⁶Brew is a packet manager for macOS that can be found here: <https://brew.sh>

Column	Type	Description	Flags
bayesClass	SC	Naive Bayes Class Name	

6.5 Known Bugs and Limitations

Currently the naivebayes library only supports discrete input features.

7 bgpDecode

7.1 Description

The bgpDecode plugin analyzes BGP traffic.

7.2 Dependencies

None.

7.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
BGP_DEBUG	0	Activate debug output
BGP_IP_FORMAT	1	IP addresses representation: 0: hex, 1: IP, 2: int
BGP_AS_FORMAT	0	AS number representation: 0: ASPLAIN, 1: ASDOT, 2: ASDOT+
BGP_NOTIF_FORMAT	0	Notifications representation: 0: uint8, 1: string (code only), 2: string (code and subcode) (not implemented)
BGP_TRAD_BOGONS	1	Flag traditional bogons
BGP_RT	1	Store routing information in a hashtable (required for MOAS detection)
BGP_DEBUG_RT	0	Activate debug output for routing information
BGP_OUTPUT_RT	1	Output routing tables
BGP_RT_MASK	0	Use the mask as part of the key for the routing table

If BGP_OUTPUT_RT=1 then the following flags can be used:

BGP_ORIG_ID	0	Output originator id
BGP_AGGR	0	Output aggregator
BGP_CLUSTER	0	Output cluster list
BGP_COMMUNITIES	0	Output communities
BGP_MASK_FORMAT	1	Netmask representation: 0: hex, 1: IP, 2: int
BGP_AS_PATH_AGGR	0	Aggregate repetitions of the same AS
BGP_ASIZE	255	Size of arrays for update records

Name	Default	Description
BGP_SUFFIX	"_bgp.txt"	Suffix to use for routing table information
BGP_ANOM_SUFFIX	"_bgp_anomalies.txt"	Suffix for anomaly file
BGP_MOAS_SUFFIX	"_bgp_moas.txt"	Suffix for multiple origin AS (MOAS) file

7.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCTRL>0):

- BGP_SUFFIX
- BGP_ANOM_SUFFIX
- BGP_MOAS_SUFFIX

7.4 Flow File Output

The bgpDecode plugin outputs the following columns:

Column	Type	Description	Flags
bgpStat	H16	BGP status	
bgpAFlgs	H16	BGP anomalies	
bgpMsgT	H8	BGP message types	
bgpNOpen_	U32_	Number of BGP OPEN messages,	
Upd_	U32_	UPDATE messages,	
Notif_	U32_	NOTIFICATION messages,	
KeepAl_	U32_	KEEPALIVE messages,	
RteRefr	U32	ROUTE-REFRESH messages	
bgpVersion	U8	BGP version	
bgpSrcAS_dstAS	U32_U32	Source and destination AS	BGP_AS_FORMAT=0
bgpSrcAS_dstAS	SC_SC	Source and destination AS	BGP_AS_FORMAT>0
bgpSrcId_dstId	IP_IP	Source and destination BGP ID	BGP_IP_FORMAT=0
bgpSrcId_dstId	U32_U32	Source and destination BGP ID	BGP_IP_FORMAT=1
bgpSrcId_dstId	H32_H32	Source and destination BGP ID	BGP_IP_FORMAT=2
bgpHTime	U16	BGP hold time (sec)	
bgpCaps	H16	Capabilities	
bgpPAttr	H32	Path attributes	
bgpNAdver	U32	Total number of advertised routes	
bgpNWdrwn	U32	Total number of withdrawn routes	
bgpMaxAdver	U32	Max. num. of advertised routes per record	
bgpAvgAdver	D	Average num. of advertised routes per record	
bgpMaxWdrwn	U32	Max. num. of withdrawn routes per record	
bgpAvgWdrwn	D	Average num. of withdrawn routes per record	
bgpAdvPref	H32	Advertised prefixes	
bgpWdrnPref	H32	Withdrawn prefixes	
bgpNIGP_	U32_	Number of routes from origin IGP	

Column	Type	Description	Flags
EGP_	U32_	EGP,	
INC	U32	INCOMPLETE	
bgpMinASPLen	U8	Minimum AS path length	
bgpMaxASPLen	U8	Maximum AS path length	
bgpAvgASPLen	D	Average AS path length	
bgpMaxNPrepAS	U32	Maximum number of prepended AS	
bgpMinLatUp	D	Minimum inter-arrival time for update	
bgpMaxLatUp	D	Maximum inter-arrival time for update	
bgpAvgLatUp	D	Average inter-arrival time for update	
bgpMinLatKA	D	Minimum inter-arrival time for keep-alive	
bgpMaxLatKA	D	Maximum inter-arrival time for keep-alive	
bgpAvgLatKA	D	Average inter-arrival time for keep-alive	
bgpNotifCode_Subcode	U8_U8	Notification (fatal error) code and subcode	BGP_NOTIF_FORMAT=0
bgpNotifCode_Subcode	SC_U8	Notification (fatal error) code and subcode	BGP_NOTIF_FORMAT=1

7.4.1 bgpStat

The `bgpStat` column is to be interpreted as follows:

bgpStat	Description
0x0001	Flow is BGP
0x0002	Connection Not Synchronized
0x0004	Bad Message Length
0x0008	Bad Message Type
0x0010	Unsupported Version Number
0x0040	Unacceptable Hold Time
0x0080	Invalid network mask (> 32)
0x0100	Inter-arrival time for update or keep-alive < 0
0x0200	AS Mismatch
0x0400	Atomic Aggregate
0x4000	One of the array was full... increase BGP_ASIZE
0x8000	Malformed packet (snapped or segmented)

7.4.2 bgpAFlgs

The `bgpAFlgs` column is to be interpreted as follows:

bgpAFlgs	Description
0x0001	Bogons advertisement
0x0002	Prefix more specific than /24 was advertised
0x0004	Prefix less specific than /8 was advertised

bgpAFlags	Description
0x0008	Possible blackhole: community with tag 666
0x0010	Possible loop: My AS in AS path
0x0020	Multiple Origin AS (same prefix announced by more than one origin AS)
0x0040	AS prepended more than 10 times in AS path
0x0080	AS number reserved for private use in AS path (AS: 64512-65534, AS4: 4200000000-4294967294)
0x0100	Route for more specific prefix advertised

7.4.3 bgpMsgT

The `bgpMsgT` column is to be interpreted as follows:

bgpMsgT	Description
0x01	—
0x02	OPEN Message
0x04	UPDATE Message
0x08	NOTIFICATION Message
0x10	KEEPALIVE Message
0x20	ROUTE-REFRESH Message
0x40	—
0x80	—

7.4.4 bgpHTime

The `bgpHTime` column indicates the number of seconds which can elapse without receiving a message. It should be three times the frequency of keep-alive messages (the default is to send one keep-alive message every 30 seconds, thus having a hold-time of 90s). Common values for the `bgpHTime` column are:

bgpHTime	Description
0	Infinite hold-time (no keep-alive messages are sent)
1,2	Illegal values
3	Minimum legal value
< 20	Warning: A hold-time of less than 20 seconds increases the chances of peer flapping
90	Juniper
180	Cisco

7.4.5 bgpCaps

The `bgpCaps` column is to be interpreted as follows:

bgpCaps	Description
0x0001	Multiprotocol Extensions for BGP-4
0x0002	Route Refresh Capability for BGP-4
0x0004	Outbound Route Filtering Capability
0x0008	Multiple routes to a destination capability
0x0010	Extended Next Hop Encoding
0x0020	Graceful Restart Capability
0x0040	Support for 4-octet AS number capability
0x0080	Support for Dynamic Capability (capability specific)
0x0100	Multisession BGP Capability
0x0200	ADD-PATH Capability
0x0400	Enhanced Route Refresh Capability
0x0800	Long-Lived Graceful Restart (LLGR) Capability
0x1000	FQDN Capability
0x8000	Unhandled Capability, i.e., none of the above

7.4.6 bgpPAttr

The `bgpPAttr` column is to be interpreted as follows (bold attributes are mandatory, attributes in italic are deprecated):

bgpPAttr	Description	bgpPAttr	Description
0x00000001	ORIGIN	0x00010000	NEW_AS_PATH
0x00000002	AS_PATH	0x00020000	NEWAGGREGATOR
0x00000004	NEXT_HOP	0x00040000	<i>SSA, SAFI Specific Attribute</i>
0x00000008	MULTI_EXIT_DISC (MED)	0x00080000	Connector Attribute
0x00000010	LOCAL_PREF	0x00100000	<i>AS_PATHLIMIT</i>
0x00000020	ATOMIC_AGGREGATE	0x00200000	PMSI_TUNNEL
0x00000040	AGGREGATOR	0x00400000	Tunnel Encapsulation Attribute
0x00000080	COMMUNITIES	0x00800000	Traffic Engineering
0x00000100	ORIGINATOR_ID	0x01000000	IPv6 Address Specific Extended Community
0x00000200	CLUSTER_LIST	0x02000000	–
0x00000400	DPA (Designation Point Attribute)	0x04000000	PE Distinguisher Labels
0x00000800	ADVERTISER		
0x00001000	RCID_PATH / CLUSTER_ID		
0x00002000	MP_REACH_NLRI		
0x00004000	MP_UNREACH_NLRI		
0x00008000	EXTENDED_COMMUNITIES		

7.4.7 bgpAdvPref and bgpWdrnPref

The bgpAdvPref and bgpWdrnPref columns are to be interpreted as follows:

bgpAdvPref	Description	bgpAdvPref	Description	bgpAdvPref	Description
0x00000001	/1	0x000001000	/13	0x01000000	/25
0x00000002	/2	0x000020000	/14	0x02000000	/26
0x00000004	/3	0x000040000	/15	0x04000000	/27
0x00000008	/4	0x000080000	/16	0x08000000	/28
0x00000010	/5	0x000100000	/17	0x100000000	/29
0x00000020	/6	0x000200000	/18	0x200000000	/30
0x00000040	/7	0x000400000	/19	0x400000000	/31
0x00000080	/8	0x000800000	/20	0x800000000	/32
0x00000100	/9	0x001000000	/21		
0x00000200	/10	0x002000000	/22		
0x00000400	/11	0x004000000	/23		
0x00000800	/12	0x008000000	/24		

7.4.8 bgpNotifCode_Subcode

The bgpNotifCode_Subcode column is to be interpreted as follows:

Code	Subcode	Description
1	Message Header Error	
	1	Connection Not Synchronized
	2	Bad Message Length
	3	Bad Message Type
2	OPEN Message Error	
	1	Unsupported Version Number
	2	Bad Peer AS
	3	Bad BGP Identifier
	4	Unsupported Optional Parameter
	5	Deprecated
	6	Unacceptable Hold time
	7	Unsupported capability
3	UPDATE Message Error	
	1	Malformed Attribute List
	2	Unrecognized Well-known Attribute
	3	Missing Well-known Attribute
	4	Attribute Flags Error
	5	Attribute Length Error
	6	Invalid ORIGIN Attribute
	7	Deprecated

Code	Subcode	Description
8		Invalid NEXT_HOP Attribute
9		Optional Attribute Error
10		Invalid Network Field
11		Malformed AS_PATH
4		Hold Timer Expired (no subcode)
5		Finite State Machine Error
	1	Receive Unexpected Message in OpenSent State
	2	Receive Unexpected Message in OpenConfirm State
	3	Receive Unexpected Message in Established State
6		Cease
	1	Maximum Number of Prefixes Reached
	2	Administrative Shutdown
	3	Peer De-configured
	4	Administrative Reset
	5	Connection Rejected
	6	Other Configuration Change
	7	Connection Collision Resolution
	8	Out of Resources
7		ROUTE-REFRESH Message Error
	1	Invalid Message Length

7.5 Additional Output

If BGP_OUTPUT_RT=1, then a PREFIX_bgp.txt file is created. Note that the suffix can be configured with BGP_SUFFIX. This file uses the configuration options defined in Section 7.3.

Column	Type	Description	Flags
NLRI	R(S)	Target network	
AS	U32/S	Originating AS	BGP_AS_FORMAT
NextHop	IP4	Next hop	
MED	U32	Multi Exit Discriminator (MED)	
LocPref	U32	Local Preference	
Origin	S	Origin (IGP, EGP, INCOMPLETE, UNKNOWN)	
OriginatorID	IP4	Originator ID	BGP_ORIG_ID=1
OriginAS	R(U32)	Origin AS	
UpstreamAS	R(U32)	Upstream AS	
DestAS	U32	Destination AS	
Aggregator	S	Aggregator (AS:Origin)	BGP_AGGR=1
ASPath	S	List of AS to visit to reach target network	
ASPathLen	U32	Length of the AS path	
MaxNprepAS	U32	Maximum number of prepended AS	

Column	Type	Description	Flags
ClusterList	R(IP4)	Cluster list	BGP_CLUSTER=1
ClusterListLen	U32	Cluster list length	BGP_CLUSTER=1
Communities	R(S)	List of communities (AS:tag)	BGP_COMMUNITIES=1
WithdrawnRoutes	R(S)	List of withdrawn routes	
flowInd	U64	Flow index of the advertisement	
pktNo	U64	Packet index of the advertisement	
RecNum	U64	Record index (within the packet) of the advertisement	
time	U32	Timestamp of the advertisement	

7.6 Plugin Report Output

The number of BGP packets and OPEN, UPDATE, NOTIFICATION, KEEP-ALIVE and ROUTE-REFRESH messages is reported. In addition, the aggregated [bgpAFlags](#) anomalies are reported.

7.7 Post-Processing

7.7.1 bgpR

The `bgpR` script creates a `PREFIX_bgp_s.txt` file, which is similar to the input file, but easier to process. The target networks are split and sorted, redundant records are omitted and the list of countries and continents to visit to reach the target network is added ([ASPathCountries](#) and [ASPathContinents](#)).

In addition, the following files are created:

- `PREFIX_bgp_mpath.txt`: outputs, for all networks which have more than one path, the number of distinct paths.
- `PREFIX_bgp_conf.txt`: lists possible configuration errors, e.g., an AS number prepended N times, followed by AS number N .

The script can also be used to plot AS paths between AS numbers, countries and continents (Figure 5).

Usage:

The `bgpR` script uses the `PREFIX_bgp.txt` file described in Section 7.5 as input: `./bgpR PREFIX_bgp.txt`. For a complete list of options, use the `-h` or `--help` option: `bgpR --help`.

Plot Configuration:

To color specific countries, edit the `bgpR` script (search for `colorN`), by adding a case for the missing country or by changing the color. For example, to color Switzerland in red, add the following line in the switch:

```
case "CH": color = "red"; break;
```

For the coloring of the edges, search for `cstr` and edit the semi-colon separated string. A complete list of colors can be found at <http://www.graphviz.org/doc/info/colors.html>.

7.7.2 ASPathCountries

For a list of countries, refer to `countrycodes.txt` in the doc folder.

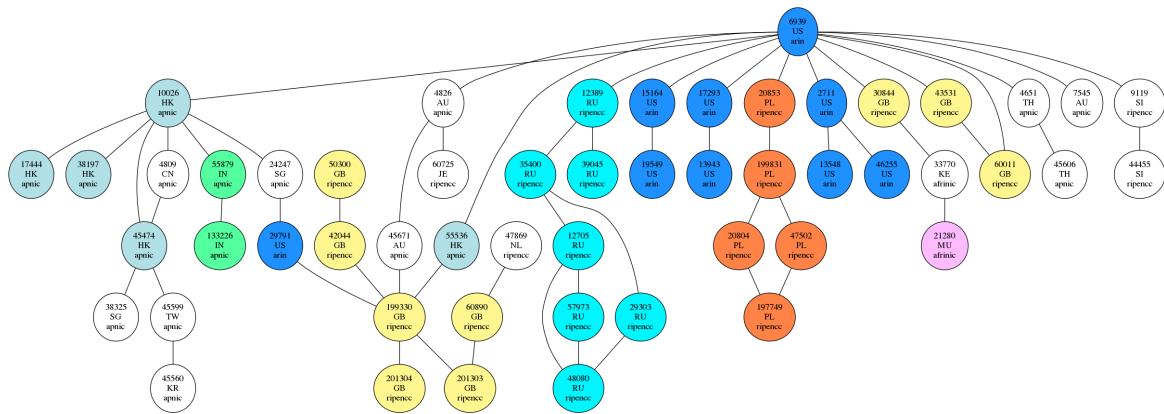


Figure 5: Paths between AS

7.7.3 ASPathContinents

The ASPathContinents column is to be interpreted as follows:

ASPathContinents	Description
afrinic	Africa Region
apnic	Asia/Pacific Region
arin	Canada, USA and some Caribbean Islands
ietf	Reserved/Unknown
lacnic	Latin America and some Caribbean Islands
ripencc	Europe, the Middle East and Central Asia



7.7.4 bgp2ng

The `bgp2ng` script can be used to generate plots similar to `bgpR` but readable by netgraph (Figure 6). It uses the `PREFIX_bgp_s.txt` file described in Section 7.7.1 as input:

```
./bgp2ng PREFIX_bgp_s.txt
```

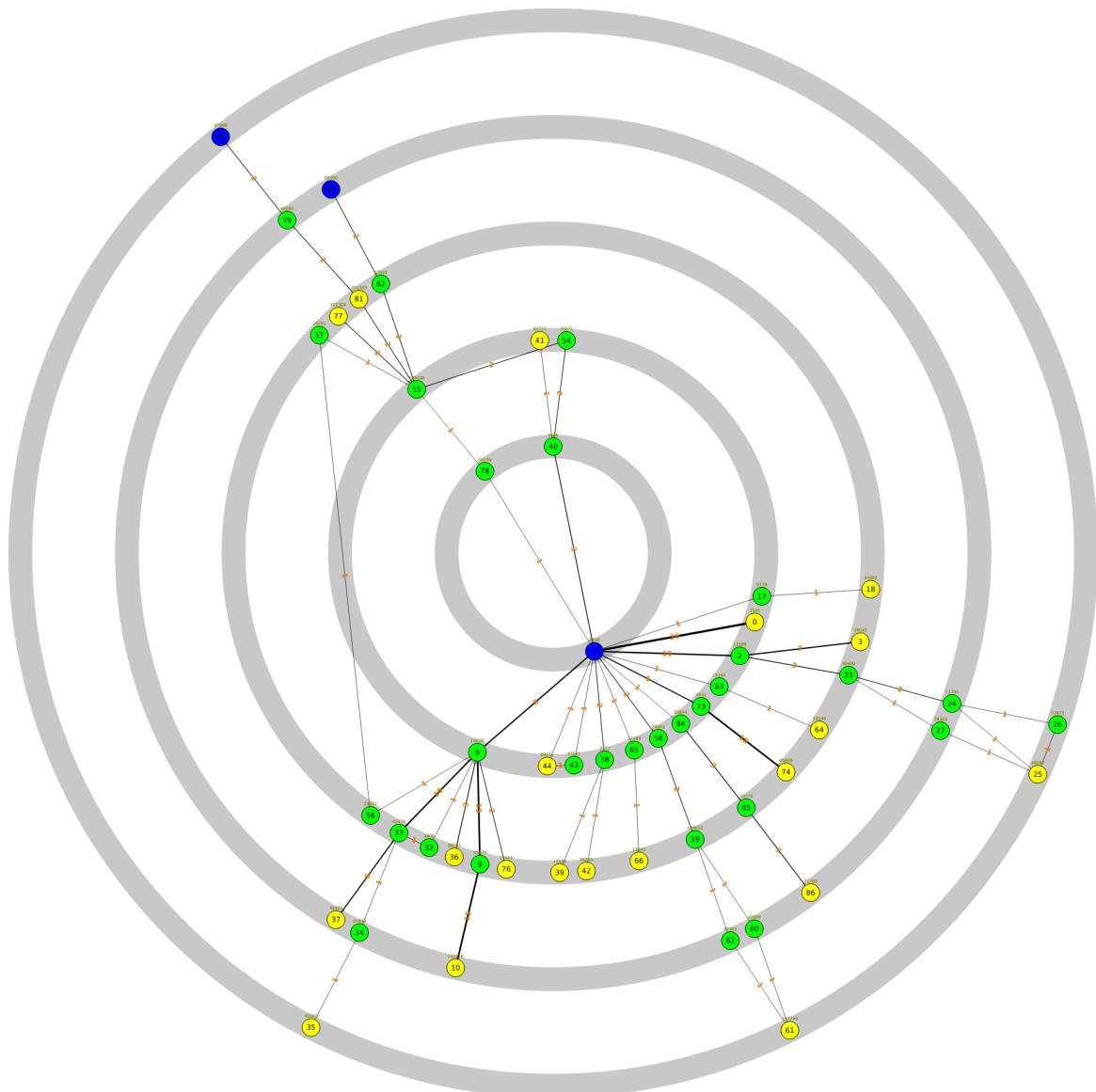


Figure 6: Paths between AS

It creates the following files:

- `PREFIX_bgp_netgraph0.txt`: connections between network and next hop

- PREFIX_bgp_netgraph1.txt: connections between first and last AS
- PREFIX_bgp_netgraph2.txt: connections between all AS
- PREFIX_bgp_netgraph3.txt: connections between first and last country
- PREFIX_bgp_netgraph4.txt: connections between all countries
- PREFIX_bgp_netgraph5.txt: connections between first and last continent
- PREFIX_bgp_netgraph6.txt: connections between all countries
- PREFIX_bgp_netgraph7.txt: connections between network, next hop, first and last AS, country and continent, ASPath

7.7.5 bgpMOAScn

The `bgpMOAScn` script adds information regarding the country of the multiple origins AS (MOAS) reported in the file `FILE_moas.txt`. Save the results in `FILE_moas_cn.txt`. Use `bgpMOAScn -h` for more information. Note that it is best to validate the countries using `whois` on the AS number and network to ensure the information is up to date, e.g., `whois AS1234` or `whois 1.2.3.4/24`.

7.8 Anomalies

Anomalies are summarized in the `bgpAFlgs` columns and in `FILE_bgp_anomalies.txt`.

- Bogons advertisement:
`awk '/^BOGON/' FILE_bgp_anomalies.txt`
- Prefix more specific than /24:
`awk '/^SPEC24/' FILE_bgp_anomalies.txt`
- Prefix less specific than /8:
`awk '/^SPEC8/' FILE_bgp_anomalies.txt`
- Possible blackhole:
`awk '/^BLACKHOLE/' FILE_bgp_anomalies.txt`
- Possible loop:
`awk '/^LOOP/' FILE_bgp_anomalies.txt`
- Multiple Origin AS (MOAS) are reported in `FILE_moas.txt` (see also `bgpMOAScn`)
- AS number prepended more than 10 times in AS path:
`awk '/^NPREPAS/' FILE_bgp_anomalies.txt`
- Private/Reserved AS numbers:
`awk '/^PRIVAS/' FILE_bgp_anomalies.txt`
- More specific prefix to existing network:
`awk '/^MSPEC/' FILE_bgp_anomalies.txt`

In addition, the number of distinct paths (if bigger than one) to reach a specific network is summarized in the file `FILE_bgp_mpath.txt` along with some statistics regarding the different AS path lengths (minimum, maximum, range, mean, median, standard deviation and mode). The file also highlights whether a MOAS was detected for the network. Possible misconfigurations are reported in `FILE_bgp_conf.txt`. Note that the last two files are created by the `bgpR` script.

7.9 Examples

- BGP flows can be extracted from a flow file by using the `bgpStat` column as follows:

```
tawk 'hdr() || strtonum($bgpStat)' FILE_flows.txt
```

- BGP flows with anomalies can be extracted from a flow file by using the `bgpAFlgs` column as follows:

```
tawk 'hdr() || strtonum($bgpAFlgs)' FILE_flows.txt
```

- More details about the anomalies listed in FILE_bgp_anomalies.txt can be found by using the flowInd, pktNo and RecNum columns and FILE_bgp_s.txt or FILE_bgp.txt files as follows, e.g., for flowInd=1, pktNo=2 and RecNum=3 (note that tawk flow and packet functions could also be used):

```
tawk '$flowInd == 1 && $pktNo == 2 && $RecNum == 3' FILE_bgp_s.txt
```

Alternatively, the following command can be used to achieve similar results:

```
grep -P "1\t2\t3" FILE_bgp_s.txt
```

- Display the 10 networks which experienced the most more specific prefix advertisements:

```
tawk '/^MSPEC/ { a[$ASorNet]++ } END { for (i in a) print i, a[i] }'  
FILE_bgp_anomalies.txt | sort -nrk2 | head -10
```

- To plot the AS paths for a specific network, proceed as follows:

- Run the bgpR script: bgpR FILE_bgp.txt
- Run the bgp2ng script: bgp2ng FILE_bgp_s.txt
- Use tawk as follows, e.g., to keep network 123.123.123.0/24 only:

```
tawk '  
    hdr() {  
        printf "%s\t%s\t%s\n", chomp($0), "ASP1", "ASP2"  
        next  
    }  
  
    $NLRI ~ /^123\.123\.123\.0\/24$/ {  
        l = split($ASPath, asp, ";")  
        for (i = 1; i < l; i++) {  
            printf "%s\t%s\t%s\n", chomp($0), asp[i], asp[i+1]  
        }  
    }  
' FILE_bgp_netgraph7.txt
```

- Open the file with Traviz/Netgraph

7.10 References

- [RFC4271](#): A Border Gateway Protocol 4 (BGP-4)
- [IPv4 traditional bogons list](#)
- <https://www.iana.org/assignments/>

8 binSink

8.1 Description

The binSink plugin is one of the basic output plugin for Tranalyzer2. It uses the output prefix (-w option) to generate a binary flow file with suffix `_flows.bin`. All standard output from every plugin is stored in binary format in this file.

8.2 Dependencies

8.2.1 External Libraries

If gzip compression is activated (`BFS_GZ_COMPRESS=1`), then **zlib** must be installed.

BFS_GZ_COMPRESS=1		
Ubuntu:	<code>sudo apt-get install zlib1g-dev</code>	
Arch:	<code>sudo pacman -S zlib</code>	
Gentoo:	<code>sudo emerge zlib</code>	
openSUSE:	<code>sudo zypper install zlib-devel</code>	
Red Hat/Fedora ⁷ :	<code>sudo dnf install zlib-devel</code>	
macOS ⁸ :	<code>brew install zlib</code>	

8.2.2 Core Configuration

This plugin requires the following core configuration:

- `$T2HOME/tranalyzer2/src/tranalyzer.h`:
 - `BLOCK_BUF=0`

8.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
<code>BFS_GZ_COMPRESS</code>	0	Compress (gzip) the output
<code>BFS_SFS_SPLIT</code>	1	Split the output file (Tranalyzer -W option)
<code>BFS_FLOWS_SUFFIX</code>	<code>"_flows.bin"</code>	Suffix to use for the output file

8.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- `BFS_FLOWS_SUFFIX`

⁷If the `dnf` command could not be found, try with `yum` instead

⁸Brew is a packet manager for macOS that can be found here: <https://brew.sh>

8.4 Post-Processing

8.5 t2b2t

The program `t2b2t` can be used to transform binary Tranalyzer files generated by the `binSink` or `socketSink` plugin into text or JSON files. The converted files use the same format as the ones generated by the `txtSink` or `jsonSink` plugin.

The program can be found in `$T2HOME/utils/t2b2t` and can be compiled by typing `make`.

The use of the program is straightforward:

- bin→txt: `t2b2t -r FILE_flows.bin -w FILE_flows.txt`
- bin→JSON: `t2b2t -r FILE_flows.bin -j -w FILE_flows.json`
- bin→compressed txt: `t2b2t -r FILE_flows.bin -c -w FILE_flows.txt.gz`
- bin→compressed JSON: `t2b2t -r FILE_flows.bin -c -j -w FILE_flows.json.gz`

If the `-w` option is omitted, the destination is inferred from the input file, e.g., the examples above would produce the same output files with or without the `-w` option. Note that `-w -` can be used to output to stdout.

Additionally, the `-n` option can be used **not** to print the name of the columns as the first row.

Try `t2b2t -h` for more information.

8.6 Custom File Output

- `PREFIX_flows.bin`: Binary representation of Tranalyzer output

9 bitForensic

9.1 Description

The bitForensic plugin enables the user to search the packets for specific `uint8_t`, `uint16_t`, `uint32_t` and `uint64_t` patterns, combined with user-defined bit masks and nibble swap operations.

9.2 Dependencies

None.

9.2.1 Core Configuration

This plugin requires no special core configuration.

9.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
<code>BF_PLEN</code>	0	0: Pattern search, 1: Length	
<code>BF_TOTLEN</code>	0	0: Residual length, 1: Total length	<code>BF_PLEN=1</code>
<code>BF_EXLEN</code>	2	Length excluded	<code>BF_PLEN=1</code>
<code>BF_PAT</code>	0x0915	Pattern: 1,2,4,8 bytes, defines <code>BF_PWDTH</code>	
<code>BF_MSK</code>	0xfffff	Mask: 1,2,4,8 bytes, defines <code>BF_PWDTH</code>	
<code>BF_NETODR</code>	1	Search pattern network order	
<code>BF_NIBBLESWP</code>	0	Swap nibbles in search pattern (0-1)	
<code>BF_DNUM</code>	10	Max bPDPoS flow storage	
<code>BF_SAVE_BCH</code>	0	Extract content: B/D info	
<code>BF_BSHIFT</code>	14	Start byte shift	
<code>BF_RMDIR</code>	1	Empty <code>BF_V_PATH</code> before starting	<code>BF_SAVE_BCH=1</code>
<code>BF_V_PATH</code>	" <code>/tmp/BF/</code> "	Path for raw content	<code>BF_SAVE_BCH=1</code>
<code>BF_FNAME</code>	" <code>bfnudel</code> "	Default content file name prefix	

9.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- `BF_RMDIR`
- `BF_V_PATH`
- `BF_FNAME`

9.4 Flow File Output

The bitForensic plugin outputs the following columns:

Column	Type	Description	Flags
<code>bfStat</code>	H8	Status	
<code>bfPDPoS</code>	R(U16)	Pattern Detect Position	

9.4.1 bfStat

The `bfStat` column is to be interpreted as follows:

bfStat	Description
0x01	Pattern found
0x02	Length byte, word found
0x04	—
0x08	Save content
0x10	BF_PWDTN0
0x20	BF_PWDTN1
0x40	BF_PWDTN2
0x80	—

9.5 Packet File Output

In packet mode (-s option), the bitForensic plugin outputs the following columns:

Column	Type	Description	Flags
<code>bfStat</code>	H8	Status	
<code>bfPDPoS</code>	R(U16)	Pattern Detect Position	

9.6 Plugin Report Output

The following information is reported:

- Aggregated `bfStat`
- Number of bit patterns
- Max. number of file handles (BF_SAVE_BCH=1)

10 cdpDecode

10.1 Description

The cdpDecode plugin analyzes CDP traffic.

10.2 Dependencies

10.2.1 Core Configuration

This plugin requires the following core configuration:

- `$T2HOME/tranalyzer2/src/networkHeaders.h`:
 - `ETH_ACTIVATE>0`

10.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
CDP_NADDR	5	Maximum number of IPv4 addresses
CDP_NMADDR	5	Maximum number of management addresses
CDP_NIPPG	5	Maximum number of IP prefix gateways
CDP_STRLEN	25	Maximum length of strings to store
CDP_LSTRLEN	100	Maximum length of long strings to store

10.4 Flow File Output

The cdpDecode plugin outputs the following columns:

Column	Type	Description
<code>cdpStat</code>	H8	Status
<code>cdpVer</code>	U8	Version
<code>cdpTTL</code>	U8	Time To Live (sec)
<code>cdpTLVTypes</code>	H32	TLV types
<code>cdpDevice</code>	SC	Device ID
<code>cdpPlatform</code>	S	Platform
<code>cdpSWVersion</code>	S	Software Version
<code>cdpPortID</code>	SC	Port ID
<code>cdpCaps</code>	H32	Capabilities
<code>cdpDuplex</code>	H8	Duplex
<code>cdpNVLAN</code>	U16	Native VLAN
<code>cdpVoipVLAN</code>	U16	VoIP VLAN
<code>cdpVTPMngmtDmn</code>	SC	VTP management domain
<code>cdpMAddrs</code>	R(IP4)	Management Addresses
<code>cdpAddrs</code>	R(IP4)	Addresses

Column	Type	Description
cdpIPPref_cdr	R(IP4_U8)	IP Prefix, CIDR

10.4.1 cdpStat

The `cdpStat` column is to be interpreted as follows:

cdpStat	Description
0x01	Flow is CDP
0x02	—
0x04	—
0x08	—
0x10	—
0x20	String truncated... increase CDP_STRLEN
0x40	Invalid TLV length
0x80	Snapped payload

10.4.2 cdpTLVTypes

The `cdpTLVTypes` column is to be interpreted as follows:

cdpTLVTypes	Description	cdpTLVTypes	Description
2^0 (=0x0000 0001)	—	2^{16} (=0x0001 0000)	Power Consumption
2^1 (=0x0000 0002)	Device ID	2^{17} (=0x0002 0000)	MTU
2^2 (=0x0000 0004)	Addresses	2^{18} (=0x0004 0000)	Trust Bitmap
2^3 (=0x0000 0008)	Port ID	2^{19} (=0x0008 0000)	Untrusted Port CoS
2^4 (=0x0000 0010)	Capabilities	2^{20} (=0x0010 0000)	System Name
2^5 (=0x0000 0020)	Software Version	2^{21} (=0x0020 0000)	System OID
2^6 (=0x0000 0040)	Platform	2^{22} (=0x0040 0000)	Management Address
2^7 (=0x0000 0080)	IP Prefixes	2^{23} (=0x0080 0000)	Location
2^8 (=0x0000 0100)	Protocol Hello	2^{24} (=0x0100 0000)	External Port ID
2^9 (=0x0000 0200)	VTP Management Domain	2^{25} (=0x0200 0000)	Power Requested
2^{10} (=0x0000 0400)	Native VLAN	2^{26} (=0x0400 0000)	Power Available
2^{11} (=0x0000 0800)	Duplex	2^{27} (=0x0800 0000)	Port Unidirectional
2^{12} (=0x0000 1000)	—	2^{28} (=0x1000 0000)	Energy Wise
2^{13} (=0x0000 2000)	—	2^{29} (=0x2000 0000)	Spare Pair POE
2^{14} (=0x0000 4000)	VoIP VLAN Reply	2^{30} (=0x4000 0000)	—
2^{15} (=0x0000 8000)	VoIP VLAN Query	2^{31} (=0x8000 0000)	Any type ≥ 31

10.4.3 cdpCaps

The `cdpCaps` column is to be interpreted as follows:

<code>cdpCaps</code>	Description
0x0000 0001	Router
0x0000 0002	Transparent Bridge
0x0000 0004	Source Route Bridge
0x0000 0008	Switch
0x0000 0010	Host
0x0000 0020	IGMP capable
0x0000 0040	Repeater
0x00000100–0x80000000	Reserved

10.4.4 cdpDuplex

The `cdpDuplex` column is to be interpreted as follows:

<code>cdpDuplex</code>	Description
0x0001	Half
0x0002	Full

10.5 Packet File Output

In packet mode (-s option), the `cdpDecode` plugin outputs the following columns:

Column	Type	Description	Flags
<code>cdpStat</code>	H8	Status	
<code>cdpVer</code>	U8	Version	
<code>cdpTTL</code>	U8	Time To Live (sec)	
<code>cdpTLVTypes</code>	H16	TLV types	
<code>cdpDevice</code>	SC	Device ID	
<code>cdpPlatform</code>	S	Platform	
<code>cdpPortID</code>	SC	Port ID	
<code>cdpCaps</code>	H32	Capabilities	
<code>cdpDuplex</code>	H8	Duplex	
<code>cdpNVLAN</code>	U16	Native VLAN	
<code>cdpVoipVLAN</code>	U16	VoIP VLAN	
<code>cdpVTPMgmtDmn</code>	SC	VTP management domain	
<code>cdpMAddrs</code>	IP4	Management Addresses	
<code>cdpAddrs</code>	IP4	Addresses	

10.6 Monitoring Output

In monitoring mode, the cdpDecode plugin outputs the following columns:

Column	Type	Description	Flags
cdpPkts	U64	Number of CDP packets	

10.7 Plugin Report Output

The following information is reported:

- Aggregated [cdpStat](#)
- Aggregated [cdpTLVTypes](#)
- Aggregated [cdpCaps](#)
- Number of CDP packets

11 centrality

11.1 Description

This plugin produces a connection matrix from pcap files and calculates the centrality of each IP address. The centrality is defined by the corresponding entry in the eigenvector to the largest eigenvalue of the adjacency matrix (Section 11.3).

11.2 Dependencies

11.2.1 Other Plugins

This plugin requires the `basicStats` plugin if `CENTRALITY_MATRIXENTRIES` ≥ 2 .

11.3 Eigenvector centrality

Let $G = (V, E)$ be a graph, where V is a set of vertices v_i and E a set of edges connecting those vertices. We define the *adjacency matrix* $A \in \mathbb{N}_0^{n \times n}$ matrix labelled by the vertices of G , where the entry $A_{i,j} = 1$, if there is a connecting edge between v_i and v_j .

Eigenvalues are defined as roots of the characteristic polynomial $p(\lambda) = \det(A - \lambda * I)$, where I is the identity matrix and A the adjacency matrix.

Each eigenvalue $\lambda \in \mathbb{C}$ has a corresponding eigenvector $\mathbf{x} \in \mathbb{C}^{n \times 1}$ which satisfies the equation $A * \mathbf{x} = \lambda * \mathbf{x}$.

The centrality of vertex i is defined by: $\text{centrality}(v_i) := \mathbf{x}_i$, where \mathbf{x} is the eigenvector to $\lambda_{\max} := \max(\lambda \in \mathbb{R})$. All values of \mathbf{x} are strictly positive.

This Plugin uses a simple version of the power iteration to calculate this eigenvector:

$\mathbf{x}_{n+1} = \frac{A * \mathbf{x}_n}{\|A * \mathbf{x}_n\|}$. For $n \rightarrow \infty$, \mathbf{x}_n converges to \mathbf{x} .

11.4 Configuration Flags

The following flags can be used to control the output of the plugin:

Variable	Default	Description
<code>CENTRALITY_MATRIXENTRIES</code>	1	0: $A_{i,j} \in \{0, 1\}$ 1: $A_{i,j} = \text{number of flows between } v_i \text{ and } v_j$ 2: $A_{i,j} = \text{number of bytes sent between } v_i \text{ and } v_j$ 3: $A_{i,j} = \text{bytes asymmetry between } v_i \text{ and } v_j$ 4: $A_{i,j} = \text{number of packets sent between } v_i \text{ and } v_j$ 5: $A_{i,j} = \text{packets asymmetry between } v_i \text{ and } v_j$
<code>CENTRALITY_TIME_CALC</code>	1	0: centrality calculated at application termination, $\mathbb{N} \ni n \neq 0$: centrality calculated every n seconds (dump time)
<code>CENTRALITY_IP_FORMAT</code>	1	0: IPs as unsigned integer 1: IPs as hex
<code>CENTRALITY_MATRIXFILE</code>	0	2: IPs in compressed format, e.g., 1.2.3.4
<code>CENTRALITY_TRAVIZ</code>	0	1: Write a file with triplet matrix 1: Traviz output mode

In addition, `CENTRALITY_SUFFIX` and `MATRIX_SUFFIX` can be used to control the suffix for the centrality and matrix output files ("`_centrality.txt`" and "`_matrix.txt`" respectively).

11.5 Flow File Output

There is no output to the flow file.

11.6 Centrality File Output

If CENTRALITY_TRAVIZ=1, then the first row is as follows:

```
% number_of_rows time IP centrality
```

Where `number_of_rows` is the actual number of entries in the file and `time`, `IP` and `centrality` represent the name of the various columns.

Column	Type	Description	Flags
1	U32/U64	Time (seconds)	CENTRALITY_TRAVIZ=1
2	IP4/U32	IPv4 address	
3	D	Centrality	

11.7 Matrix File Output

This File shows the adjacency matrix of your network in triplet matrix format. You can restore the original matrix A by assigning $A_{row,column} = value$ and $A_{i,j} = 0$ for all unassigned indices.

The first row and column will be full of ones as this is the maximum centrality host used for normalization.

Column	Type	Description
1	I	Matrix row
2	I	Matrix column
3	I	value of $A_{row,column}$

11.8 TODO

- Add support for IPv6
- Analyzing centralities in known networks
- Classification of networks by their centrality

12 clickhouseSink

12.1 Description

The clickhouseSink plugin outputs flows to a ClickHouse database.

12.2 Dependencies

12.2.1 External Libraries

This plugin depends on the **clickhouse-cpp** and **clickhouse** libraries.

On macOS, the **clickhouse-cpp** library can be installed with Brew⁹ (see below). Unfortunately, there is no package for the **clickhouse-cpp** library on Linux, but you can install it from source with the following commands:

```
$ git clone https://github.com/clickhouse/clickhouse-cpp
$ cd clickhouse-cpp
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install
$ sudo ldconfig
```

Make sure /usr/local/lib/ is in your library path:

```
$ ldconfig -p | tail -n +2 | grep -o '/.*/' | sort -u
```

If /usr/local/lib/ is **NOT** in your library path, you can add it with

```
$ echo "/usr/local/lib/" | sudo tee -a /etc/ld.so.conf.d/mylibs.conf
$ sudo ldconfig
```

ClickHouse packages are available, but repositories must be added in most cases:

12.2.2 Ubuntu

Make sure to run **ALL** the commands below. Otherwise an old version of the clickhouse server (incompatible with the plugin) will be installed.

```
$ sudo apt install libabsl-dev
$ sudo apt-get install apt-transport-https ca-certificates dirmngr
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv E0C56BD4
$ echo "deb https://repo.clickhouse.com/deb/stable/ main/" | sudo tee \
    /etc/apt/sources.list.d/clickhouse.list
$ sudo apt-get update
$ sudo apt-get install -y clickhouse-server clickhouse-client
$ sudo service clickhouse-server start
```

⁹Brew is a packet manager for macOS that can be found here: <https://brew.sh>

12.2.3 CentOS or RedHat

```
$ sudo yum install abseil-cpp
$ sudo yum install yum-utils
$ sudo rpm --import https://repo.clickhouse.com/CLOCKHOUSE-KEY.GPG
$ sudo yum-config-manager --add-repo https://repo.clickhouse.com/rpm/clickhouse.repo
$ sudo yum install clickhouse-server clickhouse-client
$ sudo /etc/init.d/clickhouse-server start
```

12.2.4 Arch Linux

Replace yay with your favorite AUR helper.

```
$ sudo pacman -S abseil-cpp
$ yay -S clickhouse-server-bin clickhouse-client-bin clickhouse-common-static-bin
$ sudo systemctl start clickhouse-server
```

12.2.5 openSUSE

```
$ sudo zypper install abseil-cpp
$ sudo zypper install clickhouse
```

12.2.6 macOS

```
$ brew install clickhouse-cpp
$ wget 'https://builds.clickhouse.com/master/macos/clickhouse'
$ chmod a+x ./clickhouse
$ ./clickhouse
```

12.2.7 Core Configuration

This plugin requires the following core configuration:

- *\$T2HOME/tranalyzer2/src/tranalyzer.h*:
 - BLOCK_BUF=0

12.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
CLICKHOUSE_OVERWRITE_DB	2	0: abort if DB already exists 1: overwrite DB if it already exists 2: reuse DB if it already exists
CLICKHOUSE_OVERWRITE_TABLE	2	0: abort if table already exists 1: overwrite table if it already exists 2: append to table if it already exists
CLICKHOUSE_TRANSACTION_NFLOWS	10000	0: one transaction > 0: one transaction every <i>n</i> flows

Name	Default	Description
CLICKHOUSE_HOST	"127.0.0.1"	Address of the database
CLICKHOUSE_DBPORT	9000	Port the DB is listening to
CLICKHOUSE_USER	"default"	Username to connect to DB
CLICKHOUSE_PASS	""	Password to connect to DB
CLICKHOUSE_DBNAME	"tranalyzer"	Name of the database
CLICKHOUSE_TABLE_NAME	"flow"	Name of the table

12.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- CLICKHOUSE_HOST
- CLICKHOUSE_DBPORT
- CLICKHOUSE_USER
- CLICKHOUSE_PASSWORD
- CLICKHOUSE_DBNAME
- CLICKHOUSE_TABLE_NAME

12.4 Example

```
# Run Tranalyzer
$ t2 -r file.pcap

# Connect to the ClickHouse database
$ clickhouse-client -d tranalyzer

# Number of flows
:) SELECT count(*) FROM flow;

# 10 first srcIP/dstIP pairs
:) SELECT "srcIP", "dstIP" FROM flow LIMIT 10;

# All flows from 1.2.3.4 to 1.2.3.5
:) SELECT * FROM flow WHERE "srcIP.1" = 4 AND "srcIP.2" = toIPv6('1.2.3.4') AND
    "dstIP.1" = 4 AND "dstIP.2" = toIPv6('1.2.3.5');
```

For examples of more complex queries, have a look in `$T2HOME/scripts/t2fm/clickhouse/`.

12.4.1 Clean up an existing database

```
# Connect to the ClickHouse database
$ clickhouse-client
```

```
# Drop the database
:) DROP DATABASE tranalyzer;
```

13 connStat

13.1 Description

The connStat plugin counts the connections between different IPs and ports per flow and during the pcap lifetime in order to produce an operational picture for anomaly detection.

13.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
CS_HSDRM	1	Decrement IP counters when flows die
CS_SDIPMAX	1	0: Number of src dst IP connections, 1: IP src dst connection with the highest count
CS_MR_SPOOF	0	1: Activate MAC spoof detector
CS_PBNMAX	1	1: Report packet/byte biggest talker

13.3 Flow File Output

The connStat plugin outputs the following columns:

Column	Type	Description	Flags
connSip	U32	Number of unique source IPs	
connDip	U32	Number of unique destination IPs	
connSipDip	U32	Number of connections between source and destination IPs	
connSipDprt	U32	Number of connections between source IP and destination port	
connMacSpf	U32	Number of MAC addresses per source IP	
connF	F	The <i>f</i> number: connSipDprt/connSip [EXPERIMENTAL]	CS_MR_SPOOF=1
connG	F	The <i>g</i> number: connSipDprt/connSipDip [EXPERIMENTAL]	
connNumPCnt	U64	Number of unique IP's source packet count	CS_PBNMAX=1
connNumBCnt	U64	Number of unique IP's source byte count	CS_PBNMAX=1

13.4 Monitoring Output

In monitoring mode, the connStat plugin outputs the following columns:

Column	Type	Description	Flags
connSip	U32	Number of unique source IPs	
connDip	U32	Number of unique destination IPs	
connSipDip	U32	Number of connections between source and destination IPs	
connSipDprt	U32	Number of connections between source IP and destination port	
connF	F	The <i>f</i> number: connSipDprt/connSip [EXPERIMENTAL]	
connG	F	The <i>g</i> number: connSipDprt/connSipDip [EXPERIMENTAL]	

13.5 Plugin Report Output

The following information is reported:

- Number of unique source IPs
- Number of unique destination IPs
- Number of unique source/destination IPs connections
- Max unique number of source IP / destination port connections
- IP connF=connSipDprt/connSip
- IP connG=connSipDprt/connSipDip
- Source IP with max connections (ANONYM_IP=0)
- Destination IP with max connections (ANONYM_IP=0)
- Biggest L3 packets talker (CS_PBNMAX=1)
- Biggest L3 bytes talker (CS_PBNMAX=1)

14 covertChannels

14.1 Description

The covertChannels plugin detects various types of covert channels (CCs) in IP traffic. Currently, it detects most publicly available covert channel tools. In the future, the goal is to also detect more discreet covert channels and custom implementations based on current research (covert timing channels, SkyDe, ...). This plugin produces only output to the flow file. Configuration is achieved by user defined compiler switches in `src/covertChannels.h`.

14.2 Required Files

14.2.1 cc_dns_whitelist.txt

The file `cc_dns_whitelist.txt` contains a domain names whitelist for the DNS CCs detection. Domains in this file will never be flagged as a covert channel.

- One domain name per line.
- Lines starting with % are comments.
- Suffix match is used to compare domain names against the whitelist.

14.2.2 cc_ping_whitelist.txt

The file `cc_ping_whitelist.txt` contains a whitelist of PING payloads. When using the ICMP whitelist detection method, all payload patterns not in this file will be considered as a covert channel.

- Lines starting with % are comments.
- One hex encoded pattern per line.
- The pattern starts at the 25th byte of the ICMP payload.
- Prefix match is used to compare the payload against the whitelist patterns.

For instance, to whitelist the PING packet shown in Figure 7, the whitelist should contain the following pattern:
101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637

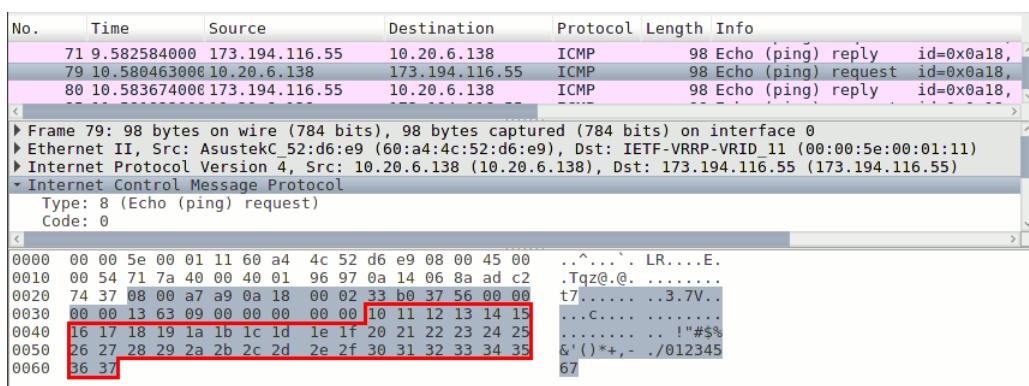


Figure 7: Wireshark view of whitelisted PING pattern.

14.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
CC_DETECT_DNS	1	detect CCs in DNS traffic
CC_DETECT_ICMP_ASYM	1	detect CCs in ICMP traffic (using flow asymmetry)
CC_DETECT_ICMP_WL	0	detect CCs in ICMP traffic (using payload whitelist)
CC_DETECT_ICMP_NP	0	detect CCs in ICMP traffic (bidirectional non-ping flow)
CC_DETECT_HCOVERT	1	detect CCs in HTTP GET requests (hcovert)
CC_DETECT_DEVCC	1	detect CCs in TCP timestamp field (devcc)
CC_DETECT_IPID	1	detect CCs in the IP Identification field (covert_tcp)
CC_DETECT_RTP_TS	0	detect CCs in the RTP timestamp field
CC_DETECT_SKYDE	0	detect CCs in Skype silent packets (SkyDe)
CC_DEBUG_MESSAGES	0	activate debug output

14.4 Flow File Output

The covertChannels plugin outputs the following column:

Column	Type	Description	Flags
covertChannels	H16	Detected covert channels bitmask	

14.4.1 covertChannels

The covertChannels column is to be interpreted as follows:

covertChannels	Description
2^0 (=0x0001)	DNS CC (iodine, dnstunnel, nstx, ...)
2^1 (=0x0002)	ICMP CC: asymmetric flow (hans, itun, loki, icmptx, ...)
2^2 (=0x0004)	ICMP CC: non-whitelisted payload (hans, itun, loki, icmptx, ...)
2^3 (=0x0008)	ICMP CC: bidirectional non-PING flow
2^4 (=0x0010)	HTTP GET URL-encoded CC (hcovert)
2^5 (=0x0020)	TCP timestamp CC (devcc)
2^6 (=0x0040)	IP Identification CC (covert_tcp)
2^7 (=0x0080)	RTP timestamp CC
2^8 (=0x0100)	Skype silent packets CC (SkyDe)

14.5 Plugin Report Output

The following information is reported:

- Aggregated `covertChannels`
- Number of covert channels packets

14.6 TODO

- Smarter IPID covert channels detection (stegtunnel)
- SSH/Telnet based covert timing channels detection

15 descriptiveStats

15.1 Description

The descriptiveStats plugin calculates various statistics about a flow. Because the inter-arrival time of the first packet is per definition always zero, it is removed from the statistics. Therefore the inter-arrival time statistics values for flows with only one packet is set to zero.

15.2 Dependencies

15.2.1 Other Plugins

This plugin requires the [pktSIATHisto](#) plugin.

15.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
DS_PS_CALC	1	Compute statistics for packet sizes	
DS_IAT_CALC	1	Compute statistics for inter-arrival times	
DS_QUARTILES	0	Quartiles calculation: 0: Use linear interpolation 1: Use the mean	DS_PS_CALC=1

15.4 Flow File Output

The descriptiveStats plugin outputs the following columns:

Column	Type	Description	Flags
dsMinPl	F	Minimum packet length	DS_PS_CALC=1
dsMaxPl	F	Maximum packet length	DS_PS_CALC=1
dsMeanPl	F	Mean packet length	DS_PS_CALC=1
dsLowQuartilePl	F	Lower quartile of packet lengths	DS_PS_CALC=1
dsMedianPl	F	Median of packet lengths	DS_PS_CALC=1
dsUppQuartilePl	F	Upper quartile of packet lengths	DS_PS_CALC=1
dsIqdPl	F	Inter quartile distance of packet lengths	DS_PS_CALC=1
dsModePl	F	Mode of packet lengths	DS_PS_CALC=1
dsRangePl	F	Range of packet lengths	DS_PS_CALC=1
dsStdPl	F	Standard deviation of packet lengths	DS_PS_CALC=1
dsRobStdPl	F	Robust standard deviation of packet lengths	DS_PS_CALC=1
dsSkewPl	F	Skewness of packet lengths	DS_PS_CALC=1
dsExcPl	F	Excess of packet lengths	DS_PS_CALC=1
dsMinIat	F	Minimum inter-arrival time	DS_IAT_CALC=1
dsMaxIat	F	Maximum inter-arrival time	DS_IAT_CALC=1
dsMeanIat	F	Mean inter-arrival time	DS_IAT_CALC=1

Column	Type	Description	Flags
dsLowQuartileIat	F	Lower quartile of inter-arrival times	DS_IAT_CALC=1
dsMedianIat	F	Median of inter-arrival times	DS_IAT_CALC=1
dsUppQuartileIat	F	Upper quartile of inter-arrival times	DS_IAT_CALC=1
dsIqdIat	F	Inter quartile distance of inter-arrival times	DS_IAT_CALC=1
dsModeIat	F	Mode of inter-arrival times	DS_IAT_CALC=1
dsRangeIat	F	Range of inter-arrival times	DS_IAT_CALC=1
dsStdIat	F	Standard deviation of inter-arrival times	DS_IAT_CALC=1
dsRobStdIat	F	Robust standard deviation of inter-arrival times	DS_IAT_CALC=1
dsSkewIat	F	Skewness of inter-arrival times	DS_IAT_CALC=1
dsExcIat	F	Excess of inter-arrival times	DS_IAT_CALC=1

15.5 Known Bugs and Limitations

Because the `pktSIATHisto` plugin stores the inter-arrival times in statistical bins, the original time information is lost. Therefore, the calculation of the inter-arrival times statistics is, due to its logarithmic binning, only a rough approximation of the original timing information. Nevertheless, this representation has shown to be useful in practical cases of anomaly and application classification.

16 dfft

16.1 Description

The dfft plugin performs Discrete Fast Fourier Transform on packet signals.

16.2 Dependencies

16.2.1 Other Plugins

This plugin requires the `nFrstPkts` plugin.

16.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
DFFT_F	1	Algorithm: 0: DFT, 1: FFT	
DFFT_N	16	Number of complex imput samples	

16.4 Flow File Output

The dfft plugin outputs the following columns:

Column	Type	Description	Flags
<code>dfftStat</code>	H8	Status	
<code>dfftF</code>	R	real_imag	

16.4.1 dfftStat

The `dfftStat` column is to be interpreted as follows:

dfftStat	Description
0x01	FT successful
0x02	—
0x04	—
0x08	—
0x10	Num input < N
0x20	Input clipped to N
0x40	—
0x80	—

16.5 Plugin Report Output

The following information is reported:

- Aggregated `dfftStat`

17 dhcpDecode

17.1 Description

This dhcpDecode plugin analyzes DHCP traffic.

17.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
DHCPMOTOUT	1	Message types/options representation: 0: bitfield, 1: numbers, 2: names	
DHCPOPTMAX	20	maximum stored options	DHCPMOTOUT=0
DHCPSMSGMAX	20	maximum stored message types	DHCPMOTOUT=0
DHCPNMMAX	10	maximal number of domain/host names per flow	
DHCPMASKFRMT	1	Netmask representation: 0: hex, 1: IP	
DHCP_ADD_CNT	0	Print the number of times a given mac/domain/host appeared	
DHCP_FLAG_MAC	0	Store a global mapping IP→MAC and add the source and destination MAC address to every flow [EXPERIMENTAL]	
DHCP_FM_DEBUG	0	print debug information about DHCP_FLAG_MAC operations	

17.3 Flow File Output

The dhcpDecode plugin outputs the following columns:

Column	Type	Description	Flags
dhcpStat	H16	Status, warnings and errors	
dhcpMTypeBF	H32	Message type bitfield	DHCPMOTOUT=0
dhcpMType	R(U8)	Message type number list	DHCPMOTOUT=1
dhcpMTypeNms	R(SC)	Message type name list	DHCPMOTOUT=2
dhcpHWType	H64	Hardware type	
dhcpCHWAdd	R(MAC)	Client hardware addresses	DHCP_ADD_CNT=0
dhcpCHWAdd_HWCnt	R(MAC_U32)	Client hardware addresses and count	DHCP_ADD_CNT=1

If IPV6_ACTIVATE=0 | 2, the following columns are output:

dhcpNetmask	H32/IP4	Network mask	DHCPMASKFRMT=0/1
dhcpGWIP	IP4	Gateway IP	
dhcpDnsIP	IP4	DNS IP	
dhcpHopCnt	H32	Hop Count	
dhcpSrvName	S	Server host name	
dhcpBootFile	S	Boot file name	
dhcpOptCnt	U16	Option Count	
dhcpOptBF1_BF2_BF3	H64_H64_H64	Options bitfield	DHCPMOTOUT=0

Column	Type	Description	Flags
dhcpOpts	R(U8)	Options	DHCPMOTOUT=1
dhcpOptNms	R(S)	Options names	DHCPMOTOUT=2
dhcpOptBF1_BF2_BF3	H64_H64_H64	Option bitfield	DHCPBITFLD=1
dhcpHosts	R(S)	Maximal DHCPNMMAX hosts	DHCP_ADD_CNT=0
dhcpHosts_HCnt	R(S_U16)	Maximal DHCPNMMAX hosts and count	DHCP_ADD_CNT=1
dhcpDomains	R(S)	Maximal DHCPNMMAX domains	DHCP_ADD_CNT=0
dhcpDomains_DCnt	R(S_U16)	Maximal DHCPNMMAX domains and count	DHCP_ADD_CNT=1
dhcpMaxSecEl	U16	Maximum seconds elapsed	
dhcpLeaseT	U32	Lease time	
dhcpRenewT	U32	Renewal time	
dhcpRebindT	U32	Rebind time	
dhcpCliIP	IP4	DHCP client IP	
dhcpYourIP	IP4	DHCP your (client) IP	
dhcpNextServer	IP4	DHCP next server IP	
dhcpRelay	IP4	DHCP relay agent IP	
dhcpLFlow	U64	DHCP linked flow	
dhcpSrcMac	MAC	DHCP source MAC address	DHCP_FLAG_MAC=1
dhcpDstMac	MAC	DHCP destination MAC address	DHCP_FLAG_MAC=1

17.3.1 dhcpStat

The `dhcpStat` status bit field is to be interpreted as follows:

dhcpStat	Description
0x0001	DHCP detected
0x0002	Boot request
0x0004	Boot reply
0x0008	Broadcast
0x0010	Client ID (option 61) different from client MAC address (DHCP header)
0x0020	Option overload: server host name and/or boot file name carry options
0x0040	Seconds elapsed probably encoded as little endian
0x0080	Non Ethernet hardware
0x0100	Option list truncated... increase DHCOPTMAX or DHCPMSGMAX
0x0200	Client HW address, domain or host name list truncated... increase DHCPNMMAX
0x0400	—
0x0800	Warning: unknown message type
0x1000	Error: DHCP invalid length
0x2000	Error: DHCP magic number corrupt
0x4000	Error: DHCP options corrupt
0x8000	Something weird happened...

17.3.2 dhcpMType and dhcpMTypeBF

For IPv4, the `dhcpMType` and `dhcpMTypeBF` columns are to be interpreted as follows:

dhcpMType	dhcpMTypeBF	Description
0	0x0000 0001	—
1	0x0000 0002	DHCP Discover Message
2	0x0000 0004	DHCP Offer Message
3	0x0000 0008	DHCP Request Message
4	0x0000 0010	DHCP Decline Message
5	0x0000 0020	DHCP Acknowledgment Message
6	0x0000 0040	DHCP Negative Acknowledgment Message
7	0x0000 0080	DHCP Release Message
8	0x0000 0100	DHCP Informational Message
9	0x0000 0200	DHCP Force Renew Message
10	0x0000 0400	DHCP Lease Query Message
11	0x0000 0800	DHCP Lease Unassigned Message
12	0x0000 1000	DHCP Lease Unknown Message
13	0x0000 2000	DHCP Lease Active Message
14	0x0000 4000	DHCP Bulk Lease Query Message
15	0x0000 8000	DHCP Lease Query Done Message
16	0x0001 0000	DHCP Active Lease Query Message
17	0x0002 0000	DHCP Lease Query Status Message
18	0x0004 0000	DHCP TLS Message
19	0x0008 0000	—
—	0x8000 0000	All values bigger than 30 are reported here

For IPv6, the `dhcpMType` and `dhcpMTypeBF` columns are to be interpreted as follows:

dhcpMType	dhcpMTypeBF	Description
0	0x0000 0001	DHCPv6 Reserved
1	0x0000 0002	DHCPv6 SOLICIT
2	0x0000 0004	DHCPv6 ADVERTISE
3	0x0000 0008	DHCPv6 REQUEST
4	0x0000 0010	DHCPv6 CONFIRM
5	0x0000 0020	DHCPv6 RENEW
6	0x0000 0040	DHCPv6 REBIND
7	0x0000 0080	DHCPv6 REPLY
8	0x0000 0100	DHCPv6 RELEASE

dhcpMType	dhcpMTypeBF	Description
9	0x0000 0200	DHCPv6 DECLINE
10	0x0000 0400	DHCPv6 RECONFIGURE
11	0x0000 0800	DHCPv6 INFORMATION-REQUEST
12	0x0000 1000	DHCPv6 RELAY-FORW
13	0x0000 2000	DHCPv6 RELAY-REPL
14	0x0000 4000	DHCPv6 LEASEQUERY
15	0x0000 8000	DHCPv6 LEASEQUERY-REPLY
16	0x0000 1000	DHCPv6 RELAY-FORW
17	0x0000 2000	DHCPv6 RELAY-REPL
18	0x0000 4000	DHCPv6 LEASEQUERY
19	0x0000 8000	DHCPv6 LEASEQUERY-REPLY
20	0x0001 0000	DHCPv6 LEASEQUERY-DONE
21	0x0002 0000	DHCPv6 LEASEQUERY-DATA
22	0x0004 0000	DHCPv6 RECONFIGURE-REQUEST
23	0x0008 0000	DHCPv6 RECONFIGURE-REPLY
24	0x0010 0000	DHCPv6 DHCPV4-QUERY
25	0x0020 0000	DHCPv6 DHCPV4-RESPONSE
26	0x0040 0000	DHCPv6 ACTIVELEASEQUERY
27	0x0080 0000	DHCPv6 STARTTLS

17.3.3 dhcpHWType

The `dhcpHWType` column is to be interpreted as follows:

dhcpHWType	Description
2^0 (=0x0000 0000 0000 0001)	—
2^1 (=0x0000 0000 0000 0002)	Ethernet
2^2 (=0x0000 0000 0000 0004)	Experimental Ethernet
2^3 (=0x0000 0000 0000 0008)	Amateur Radio AX.25
2^4 (=0x0000 0000 0000 0010)	Proteon ProNET Token Ring
2^5 (=0x0000 0000 0000 0020)	Chaos
2^6 (=0x0000 0000 0000 0040)	IEEE 802
2^7 (=0x0000 0000 0000 0080)	ARCNET
2^8 (=0x0000 0000 0000 0100)	Hyperchannel
2^9 (=0x0000 0000 0000 0200)	Lanstar
2^{10} (=0x0000 0000 0000 0400)	Autonet Short Address
2^{11} (=0x0000 0000 0000 0800)	LocalTalk
2^{12} (=0x0000 0000 0000 1000)	LocalNet (IBM PCNet or SYTEK LocalNET)

dhcpHWType	Description
2^{13} (=0x0000 0000 0000 2000)	Ultra link
2^{14} (=0x0000 0000 0000 4000)	SMDS
2^{15} (=0x0000 0000 0000 8000)	Frame Relay
2^{16} (=0x0000 0000 0001 0000)	ATM, Asynchronous Transmission Mode
2^{17} (=0x0000 0000 0002 0000)	HDLC
2^{18} (=0x0000 0000 0004 0000)	Fibre Channel
2^{19} (=0x0000 0000 0008 0000)	ATM, Asynchronous Transmission Mode
2^{20} (=0x0000 0000 0010 0000)	Serial Line
2^{21} (=0x0000 0000 0020 0000)	ATM, Asynchronous Transmission Mode
2^{22} (=0x0000 0000 0040 0000)	MIL-STD-188-220
2^{23} (=0x0000 0000 0080 0000)	Metricom
2^{24} (=0x0000 0000 0100 0000)	IEEE 1394.1995
2^{25} (=0x0000 0000 0200 0000)	MAPOS
2^{26} (=0x0000 0000 0400 0000)	Twinaxia
2^{27} (=0x0000 0000 0800 0000)	EUI-64
2^{28} (=0x0000 0000 1000 0000)	HIPARP
2^{29} (=0x0000 0000 2000 0000)	IP and ARP over ISO 7816-3
2^{30} (=0x0000 0000 4000 0000)	ARPSec
2^{31} (=0x0000 0000 8000 0000)	IPsec tunnel
2^{32} (=0x0000 0001 0000 0000)	Infiniband
2^{33} (=0x0000 0002 0000 0000)	CAI, TIA-102 Project 25 Common Air Interface
2^{34} (=0x0000 0004 0000 0000)	Wiegand Interface
2^{35} (=0x0000 0008 0000 0000)	Pure IP
2^{63} (=0x8000 0000 0000 0000)	All values bigger than 62 are reported here

17.3.4 dhcpHopCnt

The `dhcpHopCnt` column is to be interpreted as follows:

dhcpHopCnt	Description
0x00000000-0x00010000	Number of hops (0–16) (2^{HopCount})
0x80000000	Invalid hop count (> 16)

17.3.5 dhcpOptBF1_BF2_BF3

The `dhcpOptBF1_BF2_BF3` column is to be interpreted as follows:

dhcpOptBF1	Description
2^0 (=0x0000.0000.0000.0001)	Pad
2^1 (=0x0000.0000.0000.0002)	Subnet Mask
2^2 (=0x0000.0000.0000.0004)	Time Offset (deprecated)
2^3 (=0x0000.0000.0000.0008)	Router
2^4 (=0x0000.0000.0000.0010)	Time Server
2^5 (=0x0000.0000.0000.0020)	Name Server
2^6 (=0x0000.0000.0000.0040)	Domain Name Server
2^7 (=0x0000.0000.0000.0080)	Log Server
2^8 (=0x0000.0000.0000.0100)	Quote Server
2^9 (=0x0000.0000.0000.0200)	LPR Server
2^{10} (=0x0000.0000.0000.0400)	Impress Server
2^{11} (=0x0000.0000.0000.0800)	Resource Location Server
2^{12} (=0x0000.0000.0000.1000)	Host Name
2^{13} (=0x0000.0000.0000.2000)	Boot File Size
2^{14} (=0x0000.0000.0000.4000)	Merit Dump File
2^{15} (=0x0000.0000.0000.8000)	Domain Name
2^{16} (=0x0000.0000.0001.0000)	Swap Server
2^{17} (=0x0000.0000.0002.0000)	Root Path
2^{18} (=0x0000.0000.0004.0000)	Extensions Path
2^{19} (=0x0000.0000.0008.0000)	IP Forwarding enable/disable
2^{20} (=0x0000.0000.0010.0000)	Non-local Source Routing enable/disable
2^{21} (=0x0000.0000.0020.0000)	Policy Filter
2^{22} (=0x0000.0000.0040.0000)	Maximum Datagram Reassembly Size
2^{23} (=0x0000.0000.0080.0000)	Default IP Time-to-live
2^{24} (=0x0000.0000.0100.0000)	Path MTU Aging Timeout
2^{25} (=0x0000.0000.0200.0000)	Path MTU Plateau Table
2^{26} (=0x0000.0000.0400.0000)	Interface MTU
2^{27} (=0x0000.0000.0800.0000)	All Subnets are Local
2^{28} (=0x0000.0000.1000.0000)	Broadcast Address
2^{29} (=0x0000.0000.2000.0000)	Perform Mask Discovery
2^{30} (=0x0000.0000.4000.0000)	Mask supplier
2^{31} (=0x0000.0000.8000.0000)	Perform router discovery
2^{32} (=0x0000.0001.0000.0000)	Router solicitation address
2^{33} (=0x0000.0002.0000.0000)	Static routing table
2^{34} (=0x0000.0004.0000.0000)	Trailer encapsulation
2^{35} (=0x0000.0008.0000.0000)	ARP cache timeout
2^{36} (=0x0000.0010.0000.0000)	Ethernet encapsulation
2^{37} (=0x0000.0020.0000.0000)	Default TCP TTL
2^{38} (=0x0000.0040.0000.0000)	TCP keepalive interval
2^{39} (=0x0000.0080.0000.0000)	TCP keepalive garbage

dhcpOptBF1	Description
2^{40} (=0x0000.0100.0000.0000)	Network Information Service Domain
2^{41} (=0x0000.0200.0000.0000)	Network Information Servers
2^{42} (=0x0000.0400.0000.0000)	NTP servers
2^{43} (=0x0000.0800.0000.0000)	Vendor specific information
2^{44} (=0x0000.1000.0000.0000)	NetBIOS over TCP/IP name server
2^{45} (=0x0000.2000.0000.0000)	NetBIOS over TCP/IP Datagram Distribution Server
2^{46} (=0x0000.4000.0000.0000)	NetBIOS over TCP/IP Node Type
2^{47} (=0x0000.8000.0000.0000)	NetBIOS over TCP/IP Scope
2^{48} (=0x0001.0000.0000.0000)	X Window System Font Server
2^{49} (=0x0002.0000.0000.0000)	X Window System Display Manager
2^{50} (=0x0004.0000.0000.0000)	Requested IP Address
2^{51} (=0x0008.0000.0000.0000)	IP address lease time
2^{52} (=0x0010.0000.0000.0000)	Option overload
2^{53} (=0x0020.0000.0000.0000)	DHCP message type
2^{54} (=0x0040.0000.0000.0000)	Server identifier
2^{55} (=0x0080.0000.0000.0000)	Parameter request list
2^{56} (=0x0100.0000.0000.0000)	Message
2^{57} (=0x0200.0000.0000.0000)	Maximum DHCP message size
2^{58} (=0x0400.0000.0000.0000)	Renew time value
2^{59} (=0x0800.0000.0000.0000)	Rebinding time value
2^{60} (=0x1000.0000.0000.0000)	Class-identifier
2^{61} (=0x2000.0000.0000.0000)	Client-identifier
2^{62} (=0x4000.0000.0000.0000)	NetWare/IP Domain Name
2^{63} (=0x8000.0000.0000.0000)	NetWare/IP information

dhcpOptBF2	Description
2^{64} (=0x0000.0000.0000.0001)	Network Information Service+ Domain
2^{65} (=0x0000.0000.0000.0002)	Network Information Service+ Servers
2^{66} (=0x0000.0000.0000.0004)	TFTP server name
2^{67} (=0x0000.0000.0000.0008)	Bootfile name
2^{68} (=0x0000.0000.0000.0010)	Mobile IP Home Agent
2^{69} (=0x0000.0000.0000.0020)	Simple Mail Transport Protocol Server
2^{70} (=0x0000.0000.0000.0040)	Post Office Protocol Server
2^{71} (=0x0000.0000.0000.0080)	Network News Transport Protocol Server
2^{72} (=0x0000.0000.0000.0100)	Default World Wide Web Server
2^{73} (=0x0000.0000.0000.0200)	Default Finger Server
2^{74} (=0x0000.0000.0000.0400)	Default Internet Relay Chat Server
2^{75} (=0x0000.0000.0000.0800)	StreetTalk Server

dhcpOptBF2	Description
2^{76} (=0x0000.0000.0000.1000)	StreetTalk Directory Assistance Server
2^{77} (=0x0000.0000.0000.2000)	User Class Information
2^{78} (=0x0000.0000.0000.4000)	SLP Directory Agent
2^{79} (=0x0000.0000.0000.8000)	SLP Service Scope
2^{80} (=0x0000.0000.0001.0000)	Rapid Commit
2^{81} (=0x0000.0000.0002.0000)	FQDN, Fully Qualified Domain Name
2^{82} (=0x0000.0000.0004.0000)	Relay Agent Information
2^{83} (=0x0000.0000.0008.0000)	Internet Storage Name Service
2^{84} (=0x0000.0000.0010.0000)	—
2^{85} (=0x0000.0000.0020.0000)	—
2^{86} (=0x0000.0000.0040.0000)	—
2^{87} (=0x0000.0000.0080.0000)	—
2^{88} (=0x0000.0000.0100.0000)	—
2^{89} (=0x0000.0000.0200.0000)	—
2^{90} (=0x0000.0000.0400.0000)	—
2^{91} (=0x0000.0000.0800.0000)	—
2^{92} (=0x0000.0000.1000.0000)	—
2^{93} (=0x0000.0000.2000.0000)	—
2^{94} (=0x0000.0000.4000.0000)	—
2^{95} (=0x0000.0000.8000.0000)	—
2^{96} (=0x0000.0001.0000.0000)	—
2^{97} (=0x0000.0002.0000.0000)	—
2^{98} (=0x0000.0004.0000.0000)	—
2^{99} (=0x0000.0008.0000.0000)	—
2^{100} (=0x0000.0010.0000.0000)	—
2^{101} (=0x0000.0020.0000.0000)	—
2^{102} (=0x0000.0040.0000.0000)	—
2^{103} (=0x0000.0080.0000.0000)	—
2^{104} (=0x0000.0100.0000.0000)	—
2^{105} (=0x0000.0200.0000.0000)	—
2^{106} (=0x0000.0400.0000.0000)	—
2^{107} (=0x0000.0800.0000.0000)	—
2^{108} (=0x0000.1000.0000.0000)	—
2^{109} (=0x0000.2000.0000.0000)	—
2^{110} (=0x0000.4000.0000.0000)	—
2^{111} (=0x0000.8000.0000.0000)	—
2^{112} (=0x0001.0000.0000.0000)	—
2^{113} (=0x0002.0000.0000.0000)	—
2^{114} (=0x0004.0000.0000.0000)	—
2^{115} (=0x0008.0000.0000.0000)	—

dhcpOptBF2	Description
2 ¹¹⁶ (=0x0010.0000.0000.0000)	—
2 ¹¹⁷ (=0x0020.0000.0000.0000)	—
2 ¹¹⁸ (=0x0040.0000.0000.0000)	—
2 ¹¹⁹ (=0x0080.0000.0000.0000)	—
2 ¹²⁰ (=0x0100.0000.0000.0000)	—
2 ¹²¹ (=0x0200.0000.0000.0000)	—
2 ¹²² (=0x0400.0000.0000.0000)	—
2 ¹²³ (=0x0800.0000.0000.0000)	—
2 ¹²⁴ (=0x1000.0000.0000.0000)	—
2 ¹²⁵ (=0x2000.0000.0000.0000)	—
2 ¹²⁶ (=0x4000.0000.0000.0000)	—
2 ¹²⁷ (=0x8000.0000.0000.0000)	—

dhcpOptBF3	Description
2 ¹²⁸ (=0x0000.0000.0000.0001)	TFTP Server IP address
2 ¹²⁹ (=0x0000.0000.0000.0002)	Call Server IP address
2 ¹³⁰ (=0x0000.0000.0000.0004)	Discrimination string
2 ¹³¹ (=0x0000.0000.0000.0008)	Remote statistics server IP address
2 ¹³² (=0x0000.0000.0000.0010)	802.1P VLAN ID
2 ¹³³ (=0x0000.0000.0000.0020)	802.1Q L2 Priority
2 ¹³⁴ (=0x0000.0000.0000.0040)	Diffserv Code Point
2 ¹³⁵ (=0x0000.0000.0000.0080)	HTTP Proxy for phone-specific applications
2 ¹³⁶ (=0x0000.0000.0000.0100)	PANA Authentication Agent
2 ¹³⁷ (=0x0000.0000.0000.0200)	LoST Server
2 ¹³⁸ (=0x0000.0000.0000.0400)	CAPWAP Access Controller addresses
2 ¹³⁹ (=0x0000.0000.0000.0800)	OPTION-IPv4_Address-MoS
2 ¹⁴⁰ (=0x0000.0000.0000.1000)	OPTION-IPv4_FQDN-MoS
2 ¹⁴¹ (=0x0000.0000.0000.2000)	SIP UA Configuration Service Domains
2 ¹⁴² (=0x0000.0000.0000.4000)	OPTION-IPv4_Address-ANDSF
2 ¹⁴³ (=0x0000.0000.0000.8000)	OPTION-IPv6_Address-ANDSF
2 ¹⁴⁴ (=0x0000.0000.0001.0000)	—
2 ¹⁴⁵ (=0x0000.0000.0002.0000)	—
2 ¹⁴⁶ (=0x0000.0000.0004.0000)	—
2 ¹⁴⁷ (=0x0000.0000.0008.0000)	—
2 ¹⁴⁸ (=0x0000.0000.0010.0000)	—
2 ¹⁴⁹ (=0x0000.0000.0020.0000)	—
2 ¹⁵⁰ (=0x0000.0000.0040.0000)	TFTP server address or Etherboot-GRUB configuration path name
2 ¹⁵¹ (=0x0000.0000.0080.0000)	status-code

dhcpOptBF3	Description
2^{152} (=0x0000.0000.0100.0000)	base-time
2^{153} (=0x0000.0000.0200.0000)	start-time-of-state
2^{154} (=0x0000.0000.0400.0000)	query-start-time
2^{155} (=0x0000.0000.0800.0000)	query-end-time
2^{156} (=0x0000.0000.1000.0000)	dhcp-state
2^{157} (=0x0000.0000.2000.0000)	data-source
2^{158} (=0x0000.0000.4000.0000)	—
2^{159} (=0x0000.0000.8000.0000)	—
2^{160} (=0x0000.0001.0000.0000)	—
2^{161} (=0x0000.0002.0000.0000)	—
2^{162} (=0x0000.0004.0000.0000)	—
2^{163} (=0x0000.0008.0000.0000)	—
2^{164} (=0x0000.0010.0000.0000)	—
2^{165} (=0x0000.0020.0000.0000)	—
2^{166} (=0x0000.0040.0000.0000)	—
2^{167} (=0x0000.0080.0000.0000)	—
2^{168} (=0x0000.0100.0000.0000)	—
2^{169} (=0x0000.0200.0000.0000)	—
2^{170} (=0x0000.0400.0000.0000)	—
2^{171} (=0x0000.0800.0000.0000)	—
2^{172} (=0x0000.1000.0000.0000)	—
2^{173} (=0x0000.2000.0000.0000)	—
2^{174} (=0x0000.4000.0000.0000)	—
2^{175} (=0x0000.8000.0000.0000)	Etherboot
2^{176} (=0x0001.0000.0000.0000)	IP Telephone
2^{177} (=0x0002.0000.0000.0000)	Etherboot, PacketCable and CableHome
2^{178} (=0x0004.0000.0000.0000)	—
2^{179} (=0x0008.0000.0000.0000)	—
2^{180} (=0x0010.0000.0000.0000)	—
2^{181} (=0x0020.0000.0000.0000)	—
2^{182} (=0x0040.0000.0000.0000)	—
2^{183} (=0x0080.0000.0000.0000)	—
2^{184} (=0x0100.0000.0000.0000)	—
2^{185} (=0x0200.0000.0000.0000)	—
2^{186} (=0x0400.0000.0000.0000)	—
2^{187} (=0x0800.0000.0000.0000)	—
2^{188} (=0x1000.0000.0000.0000)	—
2^{189} (=0x2000.0000.0000.0000)	—
2^{190} (=0x4000.0000.0000.0000)	—
2^{191} (=0x8000.0000.0000.0000)	—

17.4 Packet File Output

In packet mode (-s option), the dhcpDecode plugin outputs the following columns:

Column	Type	Description	Flags
dhcpStat	H16	Status, warnings and errors	
dhcpMTypeBF	H32	Message type bitfield	DHCPMOTOUT=0
dhcpMType	R(U8)	Message type number list	DHCPMOTOUT=1
dhcpMTypeNms	R(SC)	Message type name list	DHCPMOTOUT=2
dhcpHops	U8	Number of hops	
dhcpHWType	H64	Hardware type	
dhcpTransID	U16	Transaction Identifier	
dhcpOptBF1_BF2_BF3	H64_H64_H64	Options bitfield	DHCPMOTOUT=0
dhcpOpts	R(U8)	Options	DHCPMOTOUT=1
dhcpOptNms	R(S)	Options names	DHCPMOTOUT=2
dhcpLFlow	U16	Linked flow	

17.5 Plugin Report Output

The number of DHCP packets of each type (Section 17.3.2) is reported.

17.6 TODO

- DHCPv6

17.7 References

- [RFC2131](#): Dynamic Host Configuration Protocol
- [RFC2132](#): DHCP Options and BOOTP Vendor Extensions

18 dnsDecode

18.1 Description

The dnsDecode plugin analyzes DNS traffic.

18.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
DNS_MODE	4	0: Only aggregated header count info 1: +REQ records 2: +ANS records 3: +AUX records 4: +ADD records
DNS_HEXON	1	0: Hex output flags off 1: Hex output flags on
DNS_HDRMD	0	Header, OpCode, RetCode: 0: Bitfield 1: Numeric 2: String
DNS_AGGR	0	0: Full vectors 1: Aggregate records
DNS_TYPE	0	Q/A type format: 0: Numeric 1: String
DNS_QRECMAX	15	Max # of query records / flow
DNS_ARECMAX	20	Max # of answer records / flow
DNS_WHO	0	1: Output country and organization of DNS reply addresses
DNS_MAL_TEST	0	0: No tests for malware 1: Mal test @ flow terminated 2: Mal test @ L4Callback, pcad ops
DNS_MAL_TYPE	0	Malware type format: 0: code 1: string

The following additional flag is available in `malsite.h`:

DNS_MAL_DOMAIN	1	0: Malsite IP address labeling mode 1: Malsite domain labeling mode, not implemented yet
----------------	---	---

`DNS_MAL_TEST` controls where the mal test is performed. Only in L4Callback enables a cooperation with pcaps, so that `pcapd` dumps all packets of a flow after the alarm was detected.

18.3 Flow File Output

The dnsDecode plugin outputs the following columns:

Column	Type	Description	Flags
dnsStat	H16	Status, warnings and errors	
dnsHdrOPField	H16	Header field of last packet in flow	
dnsHFlg_	H8_	Aggregated header flags,	DNS_HDRMD=0
OpC_	H16_	operational code and	
RetC	H16	return code	
dnsHFlg	H8	Aggregated header flags	DNS_HDRMD>0
dnsOpC	H16	Operational code	DNS_HDRMD=1
dnsOpN	S	Operational string	DNS_HDRMD=2
dnsRetC	H16	Return code	DNS_HDRMD=1
dnsRetN	S	Return string	DNS_HDRMD=2
dnsCntQu_	R(U16_	# of question records,	
Asw_	U16_	answer records,	
Aux_	U16_	auxiliary records and	
Add	U16)	additional records	
dnsAAqF	F	DDOS DNS AAA / query factor	

If DNS_MODE>0, the following columns are displayed:

dnsTypeBF3_BF2_BF1_BF0	H8_H16_H16_H64	Type bitfields	DNS_HEXON=1
dnsQname	R(S)	Query name records	
dnsMalCnt	U32	Domain malware count	DNS_MAL_TEST>0 && DNS_MAL_DOMAIN=1
dnsMalType	R(S)	Domain malware type string	DNS_MAL_TEST>0 && DNS_MAL_DOMAIN=1&& DNS_MAL_TYPE=1&&
dnsMalCode	R(U32)	Domain malware code	DNS_MAL_TEST>0 && DNS_MAL_DOMAIN=1&& DNS_MAL_TYPE=0
dnsAname	R(S)	Answer name records	
dnsAPname	R(S)	Name CNAME entries	
dns4Aaddress	R(IP4)	Address entries IPv4	
dns4CC_Org	R(SC_S)	IPv4 country and organization	DNS_WHO=1
dns6Aaddress	R(IP6)	Address entries IPv6	
dns6CC_Org	R(SC_S)	IPv6 country and organization	DNS_WHO=1
dnsIPMalCode	R(H32)	IP malware code	DNS_MAL_TEST>0&& DNS_MAL_DOMAIN=0
dnsQType	R(U16)	Query record type entries	DNS_TYPE=0
dnsQTypeN	R(S)	Query record type names	DNS_TYPE=1
dnsQClass	R(U16)	Query record class entries	
dnsAType	R(U16)	Answer record type entries	DNS_TYPE=0
dnsATypeN	R(S)	Answer record type names	DNS_TYPE=1

Column	Type	Description	Flags
dnsAClass	R(U16)	Answer record class entries	
dnsATTL	R(U32)	Answer record TTL entries	
dnsMXpref	R(U16)	MX record preference entries	
dnsSRVprio	R(U16)	SRV record priority entries	
dnsSRVwgt	R(U16)	SRV record weight entries	
dnsSRVprt	R(U16)	SRV record port entries	
dnsOptStat	R(H32)	Option status	

18.3.1 dnsStat

The dnsStat column is to be interpreted as follows:

dnsStat	Description
2 ⁰ (=0x0001)	DNS ports detected
2 ¹ (=0x0002)	NetBIOS DNS
2 ² (=0x0004)	DNS TCP aggregated fragmented content
2 ³ (=0x0008)	DNS TCP fragmented content state
2 ⁴ (=0x0010)	—
2 ⁵ (=0x0020)	Warning: ANY: Zone all from a domain or cached server
2 ⁶ (=0x0040)	Warning: Incremental DNS zone transfer detected
2 ⁷ (=0x0080)	Warning: DNS zone transfer detected
2 ⁸ (=0x0100)	Warning: DNS UDP length exceeded
2 ⁹ (=0x0200)	Warning: following records ignored
2 ¹⁰ (=0x0400)	Warning: Max DNS query records exceeded... increase DNS_QRECMAX
2 ¹¹ (=0x0800)	Warning: Max DNS answer records exceeded... increase DNS_ARECMAX
2 ¹² (=0x1000)	Error: DNS record length error
2 ¹³ (=0x2000)	Error: Wrong DNS PTR detected
2 ¹⁴ (=0x4000)	Warning: DNS length undercut
2 ¹⁵ (=0x8000)	Error: UDP/TCP DNS header corrupt or TCP packets missing

18.3.2 dnsHdrOPField

From the 16 bit DNS header the QR bit and bit five to nine are extracted and mapped in their correct sequence into a byte as indicated below. It provides for a normal single packet exchange flow an accurate status of the DNS transfer. For a multiple packet exchange only the last packet is mapped into the variable. In that case the aggregated header state flags should be considered.

QR	Opcode	AA	TC	RD	RA	Z	AD	CD	Rcode
1	0000	1	0	1	1	1	0	0	0000

18.3.3 dnsHFlg_OpC_RetC

For multi-packet DNS flows e.g. via TCP the aggregated header state bit field describes the status of all packets in a flow. Thus, flows with certain client and server states can be easily identified and extracted during post-processing.

dnsHFlg	Short	Description
2^7 (=0x01)	CD	Checking disabled
2^6 (=0x02)	AD	Authenticated data
2^5 (=0x04)	Z	Zone transfer
2^4 (=0x08)	RA	Recursive query support available
2^3 (=0x10)	RD	Recursion desired
2^2 (=0x20)	TC	Message truncated
2^1 (=0x40)	AA	Authoritative answer
2^0 (=0x80)	QR	0: Query / 1: Response

The four bit opcode field of the DNS header is mapped via $[2^{\text{opcode}}]$ and an OR into a 16 bit field. Thus, the client can be monitored or anomalies easily identified. E.g. appearance of reserved bits might be an indication for a covert channel or malware operation.

dnsOpC	Description
2^0 (=0x0001)	Standard query
2^1 (=0x0002)	Inverse query
2^2 (=0x0004)	Server status request
2^3 (=0x0008)	—
2^4 (=0x0010)	Notify
2^5 (=0x0020)	Update / Register (NetBIOS)
2^6 (=0x0040)	Release (NetBIOS)
2^7 (=0x0080)	Wait For Acknowledge (NetBIOS)
2^8 (=0x0100)	Refresh (NetBIOS)
2^9 (=0x0200)	reserved
2^{10} (=0x0400)	reserved
2^{11} (=0x0800)	reserved
2^{12} (=0x1000)	reserved
2^{13} (=0x2000)	reserved
2^{14} (=0x4000)	reserved
2^{15} (=0x8000)	reserved

The four bit rcode field of the DNS header is mapped via $[2^{\text{rcode}}]$ and an OR into a 16 bit field. It provides valuable information about success of DNS queries and therefore facilitates the detection of failures, misconfigurations and malicious operations.

dnsRetC	Short	Description
2^0 (=0x0001)	No error	Request completed successfully
2^1 (=0x0002)	Format error	Name server unable to interpret query
2^2 (=0x0004)	Server failure	Name server unable to process query due to problem with name server
2^3 (=0x0008)	Name error	Authoritative name server only: Domain name in query does not exist
2^4 (=0x0010)	Not implemented	Name server does not support requested kind of query
2^4 (=0x0020)	Refused	Name server refuses to perform the specified operation for policy reasons
2^5 (=0x0040)	YXDomain	Name exists when it should not
2^6 (=0x0080)	YXRRSet	Resource record set exists when it should not
2^8 (=0x0100)	NXRRSet	Resource record set that should exist does not
2^9 (=0x0200)	NotAuth	Server not authoritative for zone
2^{10} (=0x0400)	NotZone	Name not contained in zone
2^{11} (=0x0800)	—	—
2^{12} (=0x1000)	—	—
2^{13} (=0x2000)	—	—
2^{14} (=0x4000)	—	—
2^{15} (=0x8000)	—	—

18.3.4 dnsTypeBF3_BF2_BF1_BF0

The 16 bit Type Code field is extracted from each DNS record and mapped via $[2^{\text{Typecode}}]$ into a 64 bit fields. Gaps are avoided by additional higher bitfields defining higher codes.

dnsTypeBF3	Short	Description
2^0 (=0x01)	TA	DNSSEC Trust Authorities
2^1 (=0x02)	DLV	DNSSEC Lookaside Validation
2^2 (=0x04)	—	—
2^3 (=0x08)	—	—
2^4 (=0x10)	—	—
2^5 (=0x20)	—	—
2^6 (=0x40)	—	—
2^7 (=0x80)	—	—

dnsTypeBF2	Short	Description
2^0 (=0x0001)	TKEY	Transaction Key
2^1 (=0x0002)	TSIG	Transaction Signature
2^2 (=0x0004)	IXFR	Incremental transfer
2^3 (=0x0008)	AXFR	Transfer of an entire zone
2^4 (=0x0010)	MAILB	Mailbox-related RRs (MB, MG or MR)

dnsTypeBF2	Short	Description
2^5 (=0x0020)	MAILA	Mail agent RRs (OBSOLETE - see MX)
2^6 (=0x0040)	ZONEALL	Request for all records the server/cache has available
2^7 (=0x0080)	URI	URI
2^8 (=0x0100)	CAA	Certification Authority Restriction
2^9 (=0x0200)	—	—
2^{10} (=0x0400)	—	—
2^{11} (=0x0800)	—	—
2^{12} (=0x1000)	—	—
2^{13} (=0x2000)	—	—
2^{14} (=0x4000)	—	—
2^{15} (=0x8000)	—	—

dnsTypeBF1	Short	Description
2^0 (=0x0001)	SPF	
2^1 (=0x0002)	UINFO	
2^2 (=0x0004)	UID	
2^3 (=0x0008)	GID	
2^4 (=0x0010)	UNSPEC	
2^4 (=0x0020)	NID	
2^5 (=0x0040)	L32	
2^6 (=0x0080)	L64	
2^8 (=0x0100)	LP	
2^9 (=0x0200)	EUI48	EUI-48 address
2^{10} (=0x0400)	EUI64	EUI-48 address
2^{11} (=0x0800)	—	—
2^{12} (=0x1000)	—	—
2^{13} (=0x2000)	—	—
2^{14} (=0x4000)	—	—
2^{15} (=0x8000)	—	—

dnsTypeBF0	Short	Description
2^0 (=0x0000.0000.0000.0001)	—	—
2^1 (=0x0000.0000.0000.0002)	A	IPv4 address
2^2 (=0x0000.0000.0000.0004)	NS	Authoritative name server
2^3 (=0x0000.0000.0000.0008)	MD	Mail destination. Obsolete use MX instead
2^4 (=0x0000.0000.0000.0010)	MF	Mail forwarder. Obsolete use MX instead

dnsTypeBF0	Short	Description
2 ⁵ (=0x0000.0000.0000.0020)	CNAME	Canonical name for an alias
2 ⁶ (=0x0000.0000.0000.0040)	SOA	Marks the start of a zone of authority
2 ⁷ (=0x0000.0000.0000.0080)	MB	Mailbox domain name
2 ⁸ (=0x0000.0000.0000.0100)	MG	Mail group member
2 ⁹ (=0x0000.0000.0000.0200)	MR	Mail rename domain name
2 ¹⁰ (=0x0000.0000.0000.0400)	NULL	Null resource record
2 ¹¹ (=0x0000.0000.0000.0800)	WKS	Well known service description
2 ¹² (=0x0000.0000.0000.1000)	PTR	Domain name pointer
2 ¹³ (=0x0000.0000.0000.2000)	HINFO	Host information
2 ¹⁴ (=0x0000.0000.0000.4000)	MINFO	Mailbox or mail list information
2 ¹⁵ (=0x0000.0000.0000.8000)	MX	Mail exchange
2 ¹⁶ (=0x0000.0000.0001.0000)	TXT	Text strings
2 ¹⁷ (=0x0000.0000.0002.0000)	—	Responsible Person
2 ¹⁸ (=0x0000.0000.0004.0000)	AFSDB	AFS Data Base location
2 ¹⁹ (=0x0000.0000.0008.0000)	X25	X.25 PSDN address
2 ²⁰ (=0x0000.0000.0010.0000)	ISDN	ISDN address
2 ²¹ (=0x0000.0000.0020.0000)	RT	Route Through
2 ²² (=0x0000.0000.0040.0000)	NSAP	NSAP address. NSAP style A record
2 ²³ (=0x0000.0000.0080.0000)	NSAP-PTR	—
2 ²⁴ (=0x0000.0000.0100.0000)	SIG	Security signature
2 ²⁵ (=0x0000.0000.0200.0000)	KEY	Security key
2 ²⁶ (=0x0000.0000.0400.0000)	PX	X.400 mail mapping information
2 ²⁷ (=0x0000.0000.0800.0000)	GPOS	Geographical Position
2 ²⁸ (=0x0000.0000.1000.0000)	AAAA	IPv6 Address
2 ²⁹ (=0x0000.0000.2000.0000)	LOC	Location Information
2 ³⁰ (=0x0000.0000.4000.0000)	NXT	Next Domain (obsolete)
2 ³¹ (=0x0000.0000.8000.0000)	EID	Endpoint Identifier
2 ³² (=0x0000.0001.0000.0000)	NIMLOC/NB	Nimrod Locator / NetBIOS general Name Service
2 ³³ (=0x0000.0002.0000.0000)	SRV/NBSTAT	Server Selection / NetBIOS NODE STATUS
2 ³⁴ (=0x0000.0004.0000.0000)	ATMA	ATM Address
2 ³⁵ (=0x0000.0008.0000.0000)	NAPTR	Naming Authority Pointer
2 ³⁶ (=0x0000.0010.0000.0000)	KX	Key Exchanger
2 ³⁷ (=0x0000.0020.0000.0000)	CERT	—
2 ³⁸ (=0x0000.0040.0000.0000)	A6	A6 (OBSOLETE - use AAAA)
2 ³⁹ (=0x0000.0080.0000.0000)	DNAME	—
2 ⁴⁰ (=0x0000.0100.0000.0000)	SINK	—
2 ⁴¹ (=0x0000.0200.0000.0000)	OPT	—

	dnsTypeBF0	Short	Description
2^{42} (=0x0000.0400.0000.0000)	APL	—	
2^{43} (=0x0000.0800.0000.0000)	DS	Delegation Signer	
2^{44} (=0x0000.1000.0000.0000)	SSHFP	SSH Key Fingerprint	
2^{45} (=0x0000.2000.0000.0000)	IPSECKEY	—	
2^{46} (=0x0000.4000.0000.0000)	RRSIG	—	
2^{47} (=0x0000.8000.0000.0000)	NSEC	NextSECure	
2^{48} (=0x0001.0000.0000.0000)	DNSKEY	—	
2^{49} (=0x0002.0000.0000.0000)	DHCID	DHCP identifier	
2^{50} (=0x0004.0000.0000.0000)	NSEC3	—	
2^{51} (=0x0008.0000.0000.0000)	NSEC3PARAM	—	
2^{52} (=0x0010.0000.0000.0000)	TLSA	—	
2^{53} (=0x0020.0000.0000.0000)	SMIMEA	S/MIME cert association	
2^{54} (=0x0040.0000.0000.0000)	—	—	
2^{55} (=0x0080.0000.0000.0000)	HIP	Host Identity Protocol	
2^{56} (=0x0100.0000.0000.0000)	NINFO	—	
2^{57} (=0x0200.0000.0000.0000)	RKEY	—	
2^{58} (=0x0400.0000.0000.0000)	TALINK	Trust Anchor LINK	
2^{59} (=0x0800.0000.0000.0000)	CDS	Child DS	
2^{60} (=0x1000.0000.0000.0000)	CDNSKEY	DNSKEY(s) the Child wants reflected in DS	
2^{61} (=0x2000.0000.0000.0000)	OPENPGPKEY	OpenPGP Key	
2^{62} (=0x4000.0000.0000.0000)	CSYNC	Child-To-Parent Synchronization	
2^{63} (=0x8000.0000.0000.0000)	—	—	

18.4 Packet File Output

In packet mode (-s option), the dnsDecode plugin outputs the following columns:

Column	Type	Description	Flags
dnsIPs	R(IP)	IP addresses (A/AAAA records)	DNS_WHO=0
dnsIPs_cntry_org	R(IP_S_S)	IP addresses, countries and organizations (A/AAAA records)	DNS_WHO=1
dnsStat	H16	Status, warnings and errors	
dnsHdr	H16	Header field of packet	DNS_HDRMD=0
dnsHFlg_OpC_RetC	H8_H16_H16	Aggregated header flags, operational and return codes	DNS_HDRMD=1
dnsHFlg_OpN_RetN	H8_S_S	Aggregated header flags, operational and return strings	DNS_HDRMD=2
dnsCntQu_	U16_	# of question records,	
Asw_	U16_	answer records,	
Aux_	U16_	auxiliary records and	
Add	U16	additional records	

18.5 Monitoring Output

In monitoring mode, the dnsDecode plugin outputs the following columns:

Column	Type	Description	Flags
dnsPkts	U64	Number of DNS packets	
dnsQPkts	U64	Number of DNS Q packets	
dnsRPkts	U64	Number of DNS R packets	

18.6 Plugin Report Output

The following information is reported:

- Aggregated [dnsStat](#)
- Aggregated [dnsHFlg](#), [dnsOpC](#), [dnsRetC](#)
- Number of DNS packets
- Number of DNS Q packets
- Number of DNS R packets
- Number of alarms ([DNS_MAL_TEST](#)>0)

18.7 Example Output

The idea is that the string and integer array elements of question, answer, TTL and Type record entries match by column index so that easy script based mapping and post processing is possible. A sample output is shown below. Especially when large records are present the same name is printed several times which might degrade the readability. Therefore, a next version will have a multiple Aname suppressor switch, which should be off for script based post-processing.

Query name	Answer name	Answer address	TTL	Type
www.macromedia.com;	www.macromedia.com;www-mm.wip4.adobe.com	0.0.0.0;8.118.124.64	2787;4	5;1

18.8 TODO

- Compressed mode for DNS records

19 entropy

19.1 Description

The entropy plugin estimates the entropy of the snpped IP payload distribution. The number of bits of the alphabet can be 1,2,4,8. Default 8 bit, hence an alphabet of 256 symbols. The calculation of the entropy demands a certain minimum number of elements per flow. Two other key parameters, a binary and text based ratio, in combination with the entropy serve as input for AI for content and application classification. The character and binary ratio denote the degree of text or binary content respectively. All is experimental and described in detail in the [Entropy et al](#) tutorial on <https://www.tralyzer.com>.

19.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
ENT_NORM	1	0: # bits 1: Normalized entropy
ENT_NBITS	8	N bit word, vocabulary: 2^N
ENT_HPKTIG	0	Ignore first N packets
ENT_HEAD	0	Start word of entropy calc in payload
ENT_TAIL	1500	Position until entropy is calculated
ENT_THRESL	8	Threshold for minimal payload length
ENT_THRESH	8192	Threshold for maximal payload length
ENT_ALPHAD	0	Print alphabet distribution in flow file

19.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- ENT_HPKTIG
- ENT_HEAD
- ENT_TAIL
- ENT_THRESL
- ENT_THRESH

19.3 Flow File Output

The entropy plugin outputs the following columns:

Column	Type	Description	Flags
PyldEntropy	F	Payload entropy ¹⁰	
PyldChRatio	F	Payload character ratio	

¹⁰A value of -1 indicates that no entropy was calculated

Column	Type	Description	Flags
PyldBinRatio	F	Payload binary ratio	
NumBin0	U32	Number of 0 count bins	ENT_ALPHAD=1
Corr	F	entropy correction	ENT_ALPHAD=1
PyldLen	U32	Payload length	ENT_ALPHAD=1
PyldHisto	R(U32)	Payload histogram	ENT_ALPHAD=1

19.4 Plugin Report Output

The following information is reported:

- NValFlows, Min, Average, Max entropy

20 finDEXer

20.1 Description

This plugin produces a binary index mapping each flow index to its packets positions in the input pcaps. The goal of this plugin is to be able to quickly extract flows from a big pcap without having to re-process it completely. The fextractor tool can be used to extract flows from the pcaps using the generated index.

20.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
FNDXR_SPLIT	1	Split the finDEXer file with t2 -W option
FNDXR_SUFFIX	"_flows.xer"	Suffix for flows output file
FNDXR_PKTSXER_SUFFIX	"_packets.xer"	Suffix for packets output file

20.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCTRL>0):

- FNDXR_SUFFIX
- FNDXR_PKTSXER_SUFFIX

20.3 fextractor

The fextractor tool can be used to extract flows using the generated `_flows.xer` index.

```
Usage: fextractor -r INPUT[:start][,end] (-w OUTPUT | -n) [OPTIONS]... \
        [[DIR@]FLOWINDEX[:start][,end]]... | [PKTNO]...
```

Extract the flows FLOWINDEX using the `_flows.xer` INPUT generated by Tranalyzer2 finDEXer plugin. Alternatively use a list of finDEXer files generated by Tranalyzer2 -W option from index start to end. The extracted flows are written to the OUTPUT pcap.

An optional packet range can be provided on each command line FLOWINDEX to only extract packets in the range [start, end] of this flow. If start or end are omitted, they are replaced by, respectively, the first and the last available packets in the flow. The FLOWINDEX can also optionally be prefixed with a direction A or B, by default both directions are extracted.

When using packet mode (-P), a optional list of packet numbers (PKTNO) can be provided instead of the flow indexes (FLOWINDEX).

OPTIONS:

```
-P      packet mode, extract specific packets (from "t2 -s" output) instead of whole flows.
-r INPUT[:start][,end]
      either read packet indexes from a single _flows.xer file named INPUT
      or read packet indexes from multiple _flows.xer files prefixed by INPUT
      and with suffix in range [start, end]. If start or end are omitted,
```

```

they are replaced by, respectively, first and last available XER files.
When using packet mode (-P), INPUT must instead be in _packets.xer format.
-w OUTPUT write packets to pcap file OUTPUT
    OUTPUT "-" means that the PCAP is written to stdout.
-f      overwrite OUTPUT if it already exists
-i FILE read flow indexes / packet numbers from FILE. FILE can either be in _flows.txt
        format (flow index in 2nd tab-separated column), have one flow index per line,
        or be in _packets.txt format (packet number in 1st tab-separated column) when
        using packet mode (-P).
        FILE "-" means that flows are read from stdin.
-b      by default when FILE is in _flows.txt format, only directions present in
        it are extracted, this option forces both directions to be extracted even if
        only the A or B direction is present in the flow file.
-s N   skip the first N PCAPs
-p DIR search pcaps in DIR
        should only be set if pcaps were moved since Tranalyzer2 was run
-n      print oldest PCAP still available, its first packet timestamp and exit
-h      print this help message and exit

```

Example to extract flow 42, 123 and 1337 to the output.pcap file:

```
fextractor -r ~/t2_output/dmp1_flows.xer -w output.pcap 42 123 1337
```

Example to extract packet 100, 150 and 200 to the output.pcap file:

```
fextractor -P -r ~/t2_output/dmp1_packets.xer -w output.pcap 100 150 200
```

20.4 Example scenario

We want to extract all the flows whose source or destination are in China, to look at them in Wireshark.

First, we run tranalyzer with at least the `findexer`, `basicFlow` and `txtSink` plugins. The `findexer` plugin will generate a `_flows.xer` index file which keeps a list of packets positions in the original PCAP for each flow.

```
[user@machine]$ tranalyzer -r capture01.pcap -w t2_output/capture01
```

We now use the `srcIPCC` and `dstIPCC` columns to filter flows with IPs in China.

```
[user@machine]$ grep IPCC t2_output/capture01_headers.txt
9      SC:N      srcIPCC Source IP country code
12     SC:N      dstIPCC Destination IP country code
```

The country code are in the 9 and 12 columns. The flows to extract can directly be piped to the `fextractor` which then pipe the extracted PCAP to Wireshark.

```
[user@machine]$ awk -F"\t" '$9 == "cn" || $12 == "cn"' t2_output/capture01_flows.txt | \
fextractor -i - -r t2_output/capture01_flows.xer -w - | wireshark -k -i -
```

By using `tawk` we don't even need to look at the column numbers in the header file, we can directly extract the flows of interest using the column names. `tawk` also provides a `-k` option which takes care of extracting the flows and opening them in Wireshark.

```
[user@machine]$ tawk -k '$srcIPCC == "cn" || $dstIPCC == "cn"' t2_output/capture01_flows.txt
```

20.4.1 Extract specific packets from packet forensic mode

We want to extract all packets containing the string "Apache" to an `apache.pcap` file.

```
[user@machine]$ t2 -s -r capture01.pcap -w t2_output/capture01
[user@machine]$ tawk '$17Content ~ /Apache/' t2_output/capture01_packets.txt | \
    fextractor -P -i - -r t2_output/capture01_packets.xer -w apache.pcap
```

20.5 Additional Output (findexer v2)

20.5.1 Flows binary index

A binary index with suffix `_flows.xer` is generated. This file is composed of the following sections in any order, except the flows findexer header which is always at the beginning of the file. All numbers are written in little endian.

flows findexer header

```
struct findexer_header {
    uint64_t magic;           // 0x32455845444e4946 = FINDEXE2
    uint32_t pcapCount;       // number of input pcaps provided to tranalyzer
    // 1 if -r or number of lines in -R file
    uint64_t firstPcapHeader; // offset of the fist pcap headers (see next section)
    // in the _flows.xer file
};
```

flows pcap header

```
struct pcap_header {
    uint64_t nextPcapHeader;  // offset of the next pcap header
    // in the _flows.xer file
    uint64_t flowCount;       // number of flows in this pcap
    uint64_t firstFlowHeader; // offset of the first flow header (see next section)
    // of this pcap in the _flows.xer file
    uint16_t pathLength;      // length of the path string
    char* pcapPath;          // path string (NOT null terminated)
};
```

flow header

```
struct flow_header {
    uint64_t nextFlowHeader; // offset of the next flow header
    // in the _flows.xer file
```

```

    uint64_t flowIndex;           // Tranalyzer flow index (2nd column in flow file)
    uint8_t flags;               // flow flags (see next section)
    uint64_t packetCount;        // number of packets in this flow
#FOREACH packet in the flow
    uint64t_t offset;           // offset in the pcap where to find the packet
#ENDFOREACH
};


```

flow flags

flags	Description
2^0 (=0x01)	This is a B flow.
2^1 (=0x02)	This is the first XER file in which this flow appears.
2^2 (=0x04)	This is the last XER file in which this flow appears.
2^3 (=0x08) and all higher values:	reserved for future use.

20.5.2 Packets binary index

A binary index with suffix `_packets.xer` is generated when `tranalyzer` is run in packet forensics mode (`-s`). This file is composed of the following sections in any order, except the packets `findexer` header which is always at the beginning of the file. All numbers are written in little endian.

packets `findexer` header

```

struct pktsxer_header {
    uint64_t magic;           // 0x3252455853544b50 = PKTSXER2
    uint32_t pcapCount;       // number of input pcaps provided to tranalyzer
                            // 1 if -r or number of lines in -R file
    uint64_t firstPcapHeader; // offset of the fist pcap headers (see next section)
                            // in the _packets.xer file
};


```

packets pcap header

```

struct pkt_pcap_header {
    uint64_t nextPcapHeader;   // offset of the next pcap header
                            // in the _packets.xer file
    uint64_t first_pkt;        // fist PKTNO in this PCAP
    uint64_t last_pkt;         // last PKTNO in this PCAP
    uint16_t pathLength;       // length of the path string
    char* pcapPath;           // path string (NOT null terminated)

#FOREACH packet in the PCAP
    uint64t_t offset;          // offset in the pcap where to find the packet
#ENDFOREACH
};


```

20.6 Limitations

- PcapNg format is not supported (packet offsets in the pcap cannot be computed because of the additional block structures). PcapNg can however be converted in standard Pcap using the following command:

```
editcap -F pcap input.pcapng output.pcap
```
- The finDEXER file cannot be generated when a BPF is used. With a BPF, not all packets are processed by Tranalyzer2 which makes it impossible to compute packets offsets in a PCAP.

20.7 Old format (finDEXER v1)

finDEXER header

```
struct finDEXER_header {
    uint64_t magic;           // 0x52455845444e4946 = finDEXER
    uint32_t pcapCount;       // number of input pcaps provided to tranalyzer
                            // 1 if -r or number of lines in -R file
#FOREACH pcap
    uint64_t pcapHeaderOffset; // offset of the per pcap headers (see next section)
                            // in the _flows.xer file
#endif
};
```

pcap header

```
struct pcap_header {
    uint16_t pathLength;      // length of the path string
    char* pcapPath;          // path string (NOT null terminated)
    uint64_t flowCount;       // number of flows in this pcap
#FOREACH flow
    uint64_t flowIndex;       // Tranalyzer flow index (2nd column in flow file)
    uint64_t packetCount;     // number of packets in this flow
    uint64_t packetsOffset;   // offset in the _flows.xer file where this flow packet
                            // offsets in the pcap (see next section) are located
#endif
};
```

packet offsets

```
#FOREACH packet in the flow
    uint64_t offset; // offset in the pcap where to find the packet
#endif
```

To extract flow 123, the following steps are followed:

- open the _flows.xer file and check it has the right magic value
- for each pcap in pcapCount
 - read the pcap header located at pcapHeaderOffset in the _find.xer file.

- for each flow in flowCount
 - * if flowIndex == 123: read packetCount offsets at position packetsOffset in the _flows.xer file and extract packets located at these offsets in the pcap at pcapPath

21 fnameLabel

21.1 Description

The fnameLabel plugin tags every flow with the name of the file or interface from which the flow originates. Moreover, it adds a hash value or a label which represents the number contained in a file or a specific letter. It is predominantly used to automatically separate flows or packets created by the `-R` or `-D` option. It can be used, e.g., for training classifiers.

21.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
FNL_LBL	1	1: Output label derived from input (Use <code>fileNum</code> for Tranalyzer <code>-D</code> option, otherwise, refer to <code>FNL_IDX</code>)	
FNL_IDX	1	Use the <code>FNL_IDX</code> letter of the filename as label (Tranalyzer <code>-R/-i/-r</code> options)	FBL_LBL=1
FNL_HASH	0	1: Output hash of filename	
FNL_FLNM	1	1: Output filename	
FNL_FREL	1	Use absolute (0) or relative (1) filenames for <code>fnLabel</code> , <code>fnHash</code> and <code>fnName</code>	
FNL_NAMELEN	1024	Max length for filename	

21.3 Flow File Output

The fnameLabel plugin outputs the following columns:

Column	Type	Description	Flags
fnLabel	U32	<code>FNL_IDX</code> letter of the filename/interface	FNL_LBL=1
fnHash	U64	Hash of the filename/interface	FNL_HASH=1
fnName	S	Filename	FNL_FLNM=1

Note that the filename refers to the file in which the flow was created.

21.4 Packet File Output

In packet mode (`-s` option), the fnameLabel plugin outputs the same columns as in the flow file.

22 ftpDecode

22.1 Description

The ftpDecode plugin analyzes FTP traffic. The plugin identifies the client FTP flows automatically and links them via the `ftpCDFindex`, identifying the findex of the associated flows.

22.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
FTP_SAVE	0	Save content to <code>FTP_F_PATH</code>	
FTP_RMDIR	1	Empty <code>FTP_F_PATH</code> before starting	FTP_SAVE=1
FTP_CMD_AGGR	1	Aggregate FTP commands/response codes	
FTP_BTFLD	0	Bitfield coding of FTP commands	
FTP_UXNMLN	10	maximal username length	
FTP_PXNMLN	15	maximal password length	
FTP_MXNMLN	50	maximal name length	
FTP_MAXCPFI	10	maximal number of parent finex	
FTP_MAXUNM	5	maximal number of users	
FTP_MAXPNM	5	maximal number of passwords	
FTP_MAXCNM	20	maximal number of parameters	
FTP_F_PATH	<code>"/tmp/FTPFILES/"</code>	Path for extracted content	
FTP_NONAME	<code>"nudel"</code>	Filename to use when none available	

22.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- `FTP_F_PATH`
- `FTP_NONAME`

22.3 Flow File Output

The ftpDecode plugin outputs the following columns:

Column	Type	Description	Flags
<code>ftpStat</code>	H8	Status bit field	
<code>ftpCDFindex</code>	R(U64)	Command/data finex link	
<code>ftpCBF</code>	H64	Command bitfield	FTP_BTFLD=1
<code>ftpCC</code>	R(SC)	Command codes	
<code>ftpRC</code>	R(U16)	Response codes	
<code>ftpNumUser</code>	U8	Number of users	
<code>ftpUser</code>	R(S)	Users	
<code>ftpNumPass</code>	U8	Number of passwords	

Column	Type	Description	Flags
ftpPass	R(S)	Passwords	
ftpNumCP	U8	Number of command parameters	
ftpCP	R(S)	Command parameters	

22.3.1 ftpStat

The `ftpStat` column is to be interpreted as follows:

ftpStat	Description
2 ⁰ (=0x01)	FTP control port found
2 ¹ (=0x02)	FTP passive parent flow
2 ² (=0x04)	FTP passive parent flow length overrun, possibly by dupACK/retransmits
2 ³ (=0x08)	FTP active parent flow
2 ⁴ (=0x10)	FTP hash map full
2 ⁵ (=0x20)	File error (FTP_SAVE=1)
2 ⁶ (=0x40)	Data flow not detected
2 ⁷ (=0x80)	Array, string or filename overflow

22.3.2 ftpCBF

The `ftpCBF` column is to be interpreted as follows:

ftpCBF	Description	ftpCBF	Description
2^0 (=0x0000.0000.0000.0001)	ABOR	2^{32} (=0x0000.0001.0000.0000)	PORT
2^1 (=0x0000.0000.0000.0002)	ACCT	2^{33} (=0x0000.0002.0000.0000)	PROT
2^2 (=0x0000.0000.0000.0004)	ADAT	2^{34} (=0x0000.0004.0000.0000)	PWD
2^3 (=0x0000.0000.0000.0008)	ALLO	2^{35} (=0x0000.0008.0000.0000)	QUIT
2^4 (=0x0000.0000.0000.0010)	APPE	2^{36} (=0x0000.0010.0000.0000)	REIN
2^5 (=0x0000.0000.0000.0020)	AUTH	2^{37} (=0x0000.0020.0000.0000)	REST
2^6 (=0x0000.0000.0000.0040)	CCC	2^{38} (=0x0000.0040.0000.0000)	RETR
2^7 (=0x0000.0000.0000.0080)	CDUP	2^{39} (=0x0000.0080.0000.0000)	RMD
2^8 (=0x0000.0000.0000.0100)	CONF	2^{40} (=0x0000.0100.0000.0000)	RNFR
2^9 (=0x0000.0000.0000.0200)	CWD	2^{41} (=0x0000.0200.0000.0000)	RNTO
2^{10} (=0x0000.0000.0000.0400)	DELE	2^{42} (=0x0000.0400.0000.0000)	SITE
2^{11} (=0x0000.0000.0000.0800)	ENC	2^{43} (=0x0000.0800.0000.0000)	SIZE
2^{12} (=0x0000.0000.0000.1000)	EPRT	2^{44} (=0x0000.1000.0000.0000)	SMNT
2^{13} (=0x0000.0000.0000.2000)	EPSV	2^{45} (=0x0000.2000.0000.0000)	STAT
2^{14} (=0x0000.0000.0000.4000)	FEAT	2^{46} (=0x0000.4000.0000.0000)	STOR
2^{15} (=0x0000.0000.0000.8000)	HELP	2^{47} (=0x0000.8000.0000.0000)	STOU
2^{16} (=0x0000.0000.0001.0000)	LANG	2^{48} (=0x0001.0000.0000.0000)	STRU
2^{17} (=0x0000.0000.0002.0000)	LIST	2^{49} (=0x0002.0000.0000.0000)	SYST
2^{18} (=0x0000.0000.0004.0000)	LPRT	2^{50} (=0x0004.0000.0000.0000)	TYPE
2^{19} (=0x0000.0000.0008.0000)	LPSV	2^{51} (=0x0008.0000.0000.0000)	USER
2^{20} (=0x0000.0000.0010.0000)	MDTM	2^{52} (=0x0010.0000.0000.0000)	XCUP
2^{21} (=0x0000.0000.0020.0000)	MIC	2^{53} (=0x0020.0000.0000.0000)	XMKD
2^{22} (=0x0000.0000.0040.0000)	MKD	2^{54} (=0x0040.0000.0000.0000)	XPWD
2^{23} (=0x0000.0000.0080.0000)	MLSD	2^{55} (=0x0080.0000.0000.0000)	XRCP
2^{24} (=0x0000.0000.0100.0000)	MLST	2^{56} (=0x0100.0000.0000.0000)	XRMD
2^{25} (=0x0000.0000.0200.0000)	MODE	2^{57} (=0x0200.0000.0000.0000)	XRSQ
2^{26} (=0x0000.0000.0400.0000)	NLST	2^{58} (=0x0400.0000.0000.0000)	XSEM
2^{27} (=0x0000.0000.0800.0000)	NOOP	2^{59} (=0x0800.0000.0000.0000)	XSEN
2^{28} (=0x0000.0000.1000.0000)	OPTS	2^{60} (=0x1000.0000.0000.0000)	CLNT
2^{29} (=0x0000.0000.2000.0000)	PASS		
2^{30} (=0x0000.0000.4000.0000)	PASV		
2^{31} (=0x0000.0000.8000.0000)	PBSZ		

22.4 Packet File Output

In packet mode (-s option), the ftpDecode plugin outputs the following columns:

Column	Type	Description	Flags
ftpStat	H8	Status	

22.5 Monitoring Output

In monitoring mode, the ftpDecode plugin outputs the following columns:

Column	Type	Description	Flags
ftpPkts	U64	Number of FTP control packets	
ftpDataPkts	U64	Number of FTP-DATA packets	

22.6 Plugin Report Output

The following information is reported:

- Aggregated [ftpStat](#)
- Number of FTP control packets
- Number of FTP-DATA packets
- Number of files extracted (FTP_SAVE=1)

23 geoip

23.1 Description

This plugin outputs the geographic location of IP addresses.

23.2 Dependencies

This product includes GeoLite2 data created by MaxMind, available from <http://www.maxmind.com>.

The required dependencies depend on the value of `GEOIP_LIB`:

- `GEOIP_LIB=0`:
Legacy databases (`GeoLiteCity.data.gz` and `GeoLiteCityv6.dat.gz`) require *libgeoip*.
- `GEOIP_LIB=1`:
GeoLite2 requires *libmaxminddb*.

		GEOIP_LIB=1	GEOIP_LIB=0
Ubuntu:	<code>sudo apt-get install libmaxminddb-dev</code>	<code>libgeoip-dev</code>	
Arch:	<code>sudo pacman -S libmaxminddb</code>	<code>geoip</code>	
Gentoo:	<code>sudo emerge libmaxminddb</code>	<code>geoip</code>	
openSUSE:	<code>sudo zypper install libmaxminddb-devel</code>	<code>libGeoIP-devel</code>	
Red Hat/Fedora ¹¹ :	<code>sudo dnf install libmaxminddb-devel</code>	<code>GeoIP-devel</code>	
macOS ¹² :	<code>brew install libmaxminddb</code>	<code>geoip</code>	

23.2.1 Databases Update

The latest version of the databases can be found at <https://dev.maxmind.com/geoip/geoip2/geolite2/> (GeoLite2-City). Legacy databases, the latest version of which can be found at <https://dev.maxmind.com/geoip/legacy/geolite> (Geo Lite City and Geo Lite City IPv6), are also supported.

23.3 Configuration Flags

The following flags can be used to control the output of the plugin (Information in italic only applies to legacy databases):

Name	Default	Description
<code>GEOIP_LIB</code>	2	Library to use: 2: GeoLite2 / Internal libmaxmind (faster) 1: GeoLite2 / libmaxmind 0: GeoLite / geoip (legacy)
<code>GEOIP_SRC</code>	1	Display geo info for the source IP
<code>GEOIP_DST</code>	1	Display geo info for the destination IP

¹¹If the `dnf` command could not be found, try with `yum` instead

¹²Brew is a packet manager for macOS that can be found here: <https://brew.sh>

Name	Default	Description
GEOIP_CONTINENT	2	0: no continent, 1: name (GeoLite2), 2: two letters code
GEOIP_COUNTRY	2	0: no country, 1: name, 2: two letters code, 3: <i>three letters code</i>
GEOIP_CITY	1	Display the city of the IP
GEOIP_POSTCODE	1	Display the postal code of the IP
GEOIP_POSITION	1	Display the position (latitude, longitude) of the IP
GEOIP_METRO_CODE	0	Display the metro (dma) code of the IP (US only)

If `GEOIP_LIB!=0`, the following flags are available:

GEOIP_ACCURACY	1	Display the accuracy of the geolocation
GEOIP_TIMEZONE	1	Display the time zone

The six following flags are only available in GeoLite2 Enterprise databases:

GEOIP_ORG	0	Display the organization of the IP
GEOIP_ISP	0	Display the ISP name of the IP
GEOIP ASN	0	Display the autonomous systems number of the IP
GEOIP_ASNAME	0	Display the autonomous systems name of the IP
GEOIP_CONN	0	Display the connection type of the IP
GEOIP_USRT	0	Display the user type of the IP
GEOIP_DB_FILE	"GeoLite2-City.mmdb"	Name of the database to use for IPv4 and IPv6 (combined)
GEOIP_LANG	"en"	Language to use: de: German, en: English, es: Spanish, fr: French, jp: Japanese, pt-BR: Brazilian Portuguese, ru: Russian, zh-CN: Simplified Chinese
GEOIP_BUFSIZE	64	Buffer size

If `GEOIP_LIB==0`, the following flags are available:

GEOIP_REGION	1	0: no region, 1: name, 2: code
GEOIP_AREA_CODE	0	Display the telephone area code of the IP
GEOIP_NETMASK	1	0: no netmask, 1: netmask as int (cidr), 2: netmask as hex, 3: netmask as IP

Name	Default	Description
GEOIP_DB_CACHE	2	<i>0: read DB from file system (slower, least memory) 1: index cache (cache frequently used index only) 2: memory cache (faster, more memory)</i>
GEOIP_DB_FILE4	"GeoLiteCity.dat"	<i>Name of the database to use for IPv4</i>
GEOIP_DB_FILE6	"GeoLiteCityv6.dat"	<i>Name of the database to use for IPv6</i>
GEOIP_UNKNOWN	"--"	Representation of unknown locations (GeoIP's default)

23.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCTRL>0`):

- `GEOIP_DB_FILE` (require `GEOIP_LIB>0`)
- `GEOIP_DB_FILE4` (require `GEOIP_LIB=0`)
- `GEOIP_DB_FILE6` (require `GEOIP_LIB=0`)
- `GEOIP_UNKNOWN`

23.4 Flow File Output

The geoip plugin outputs the following columns:

Column	Type	Description	Flags
The following columns prefixed with <code>src</code> are only output if <code>GEOIP_SRC=1</code> .			
<code>srcIpContinent</code>	S	Continent name	<code>GEOIP_CONTINENT=1</code>
<code>srcIpContinent</code>	SC	Continent code	<code>GEOIP_CONTINENT=2</code>
<code>srcIpCountry</code>	S	Country name	<code>GEOIP_COUNTRY=1</code>
<code>srcIpCountry</code>	SC	Country code	<code>GEOIP_COUNTRY=2 3</code>
<code>srcIpRegion</code>	SC	Region	<code>GEOIP_LIB=0&&GEOIP_REGION=1</code>
<code>srcIpRegion</code>	S	Region	<code>GEOIP_LIB=0&&GEOIP_REGION=2</code>
<code>srcIpCity</code>	S	City	<code>GEOIP_CITY>0</code>
<code>srcIpPostcode</code>	SC	Postal code	<code>GEOIP_POSTCODE>0</code>
<code>srcIpAccuracy</code>	U16	Accuracy of the geolocation (in km)	<code>GEOIP_LIB>0&&GEOIP_ACCURACY=1</code>
<code>srcIpLat</code>	D	Latitude	<code>GEOIP_LIB>0&&GEOIP_POSITION=1</code>
<code>srcIpLong</code>	D	Longitude	<code>GEOIP_LIB>0&&GEOIP_POSITION=1</code>
<code>srcIpLat</code>	F	Latitude	<code>GEOIP_LIB=0&&GEOIP_POSITION=1</code>
<code>srcIpLong</code>	F	Longitude	<code>GEOIP_LIB=0&&GEOIP_POSITION=1</code>
<code>srcIpMetroCode</code>	U16	Metro (DMA) code (US only)	<code>GEOIP_LIB>0&&GEOIP_METRO_CODE=1</code>
<code>srcIpMetroCode</code>	I32	Metro (DMA) code (US only)	<code>GEOIP_LIB=0&&GEOIP_METRO_CODE=1</code>
<code>srcIpAreaCode</code>	I32	Area code	<code>GEOIP_LIB=0&&GEOIP_AREA_CODE=1</code>
<code>srcIpNetmask</code>	U32	Netmask (CIDR)	<code>GEOIP_LIB=0&&GEOIP_NETMASK=1</code>
<code>srcIpNetmask</code>	H32	Netmask	<code>GEOIP_LIB=0&&GEOIP_NETMASK=2</code>
<code>srcIpNetmask</code>	IP4	Netmask	<code>GEOIP_LIB=0&&GEOIP_NETMASK=3</code>

Column	Type	Description	Flags
srcIpTimeZone	S	Time zone	GEOIP_LIB=0&&GEOIP_TIMEZONE=1
srcIpOrg	S	Organization	GEOIP_LIB>0&&GEOIP_ORG=1
srcIpISP	S	ISP	GEOIP_LIB>0&&GEOIP_ISP=1
srcIpASN	U32	AS number	GEOIP_LIB>0&&GEOIP ASN=1
srcIpASName	S	AS name	GEOIP_LIB>0&&GEOIP_ASNAME=1
srcIpConnT	S	Connection type	GEOIP_LIB>0&&GEOIP_CONN=1
srcIpUsrT	S	User type	GEOIP_LIB>0&&GEOIP_USRT=1

The same columns (with prefix dst instead of src) are output for the destination address if GEOIP_DST=1.

geoStat	H8	Status
---------	----	--------

23.4.1 srcIpContinent

Continent codes are as follows:

Code	Description
AF	Africa
AS	Asia
EU	Europe
NA	North America
OC	Oceania
SA	South America
--	Unknown (see <code>GEOIP_UNKNOWN</code>)

23.4.2 geoStat

The `geoStat` column is to be interpreted as follows:

geoStat	Description
2^0 (=0x01)	A string had to be truncated... increase <code>GEOIP_BUFSIZE</code>
2^1 (=0x02)	Source IP lookup failed
2^2 (=0x04)	Destination IP lookup failed

23.5 Post-Processing

23.5.1 genkml

The `geoip` plugin comes with the `genkml` script which generates a KML (Keyhole Markup Language) file from a flow file. This KML file can then be loaded in Google Earth to display the location of the IP addresses involved in the dump file. Its usage is straightforward:

```
./scripts/genkml FILE_flows.txt
```

23.5.2 t2mmdb

The `t2mmdb` program can be used to query the MaxMind DB. It is a faster and easier to use version of the `mmdblokup` utility.

23.5.3 t2mmdba

The `t2mmdba` script can be used to transform the MaxMind DB into Tranalyzer subnet format.

24 gquicDecode

24.1 Description

The gquicDecode plugin analyzes GQUIC traffic.

24.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
GQUIC_SLEN	63	Max length for string columns, e.g., gquicSNI
GQUIC_DEBUG	0	0: do not print any debug messages 1: print warnings about unhandled cases 2: + print regular info about decoding status

24.3 Flow File Output

The gquicDecode plugin outputs the following columns:

Column	Type	Description	Flags
gquicStat	H8	Status	
gquicPubFlags	H8	Public Flags	
gquicFrameTypes	H16	Frame Types	
gquicCID	U64	Connection ID	
gquicSNI	S	Server Name Indication (SNI)	
gquicUAID	S	Client's User Agent ID	

24.3.1 gquicStat

The gquicStat column is to be interpreted as follows:

gquicStat	Description
0x01	Flow is GQUIC
0x02	Handshake (Stream number is 1)
0x04	Connection ID changed
0x08	—
0x10	—
0x20	—
0x40	Packet was snapped (t2buf failed)
0x80	Packet was malformed, e.g., covert channel

24.3.2 gquicPubFlags

The `gquicPubFlags` column is to be interpreted as follows:

gquicPubFlags	Description
0x01	Header contains a GQUIC Version
0x02	Public Reset packet
0x04	32 byte diversification nonce is present (version >= 33)
0x08	8 byte Connection ID is present (version >= 33)
0x0c	8 byte Connection ID is present (version < 33)
0x30	Number of low-order bytes of the packet number
0x40	Reserved for multipath use
0x80	Reserved (MUST be 0)

24.3.3 gquicFrameTypes

The `gquicFrameTypes` column is to be interpreted as follows:

gquicFrameTypes	Description
2^0 (=0x0001)	PADDING
2^1 (=0x0002)	RST_STREAM
2^2 (=0x0004)	CONNECTION_CLOSE
2^3 (=0x0008)	GOAWAY
2^4 (=0x0010)	WINDOW_UPDATE
2^5 (=0x0020)	BLOCKED
2^6 (=0x0040)	STOP_WAITING
2^7 (=0x0080)	PING

Special Frame Types:

2^{14} (=0x4000)	ACK
2^{15} (=0x8000)	STREAM

24.4 Packet File Output

In packet mode (-s option), the `gquicDecode` plugin outputs the following columns:

Column	Type	Description	Flags
<code>gquicPubFlags</code>	H8	Public Flags	
<code>gquicVersion</code>	SC	Version	
<code>gquicPktNo</code>	U64	Packet Number	
<code>gquicCID</code>	U64	Connection ID	

24.5 Plugin Report Output

The following information is reported:

- Aggregated [gquicStat](#)
- Number of GQUIC packets
- Number of GQUIC Client Hello packets
- Number of GQUIC Rejection packets
- Number of GQUIC Public Reset packets

24.6 Known Bugs and Limitations

- The gquicDecode plugin assumes every UDP packet on port 80 or 443 is GQUIC...

25 gsmDecode

25.1 Description

The gsmDecode plugin analyzes GSM traffic.

25.2 Dependencies

25.2.1 External Libraries

This plugin does not require any external library.

25.2.2 Required Files

The file `tacdb.csv` is required.

25.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
GSM_ARFCNFILE	1	Save ARFCN in a separate file	
GSM_CALLFILE	1	Save calls in a separate file	
GSM_CDFILE	1	Save channels in a separate file	
GSM_IMSIFILE	1	Save IMSI/TMSI/IMEI/IMEISV in a separate file	
GSM_IMMASSFILE	1	Save Immediate Assignments in a separate file	
GSM_OPFILE	1	Save operator names in a separate file	
GSM_SMSFILE	1	Save SMS messages in a separate file	
GSM_ROTATE_TIME	0	Create new files every <i>N</i> seconds (use 0 to deactivate the feature)	
GSM_SPEECHFILE	1	Save audio conversations	
GSM_STATFILE	1	Save GSM statistics in a separate file	
GSM_SPEECH_SPLIT	1	Speech frames handling: 0: Save A and B flows in the same file 1: Create one file per direction	
GSM_TMSI_FORMAT	1	Format for TMSI: 0: Integer, 1: Hexadecimal	
GSM_SPEECH_DIR	"/tmp/gsm_speech"	Folder for extracted audio conversations	GSM_SPEECHFILE=1
GSM_TXT_DIR	"/tmp/gsm_txt"	Folder for output files	
GSM_RMDIR	1	Empty <code>GSM_SPEECH_DIR</code> before starting	GSM_SPEECHFILE=1

The following flag reside in `src/e164_list.h`:

GSM_E164_FORMAT 0 0: Country code, 1: Country name

Name	Default	Description	Flags
The following flags reside in src/mcc_list.h :			
GSM_MCC_FORMAT	0	0: Country code, 1: Country name	
GSM_MNC_FORMAT	0	0: Operator name, 1: Brand name	
GSM_NOT_FOUND	""	Value to use when no entry was found	

The suffix of the output files produced is controlled by the following flags:

Name	Default	Description	Flags
GSM_ARFCNFILE_SUFFIX	"_gsm_arfcn"	Suffix for ARFCN file	GSM_ARFCNFILE=1
GSM_CALLFILE_SUFFIX	"_gsm_calls"	Suffix for calls file	GSM_CALLFILE=1
GSM_CDFILE_SUFFIX	"_gsm_channels"	Suffix for channels file	GSM_CDFILE=1
GSM_IMMASSFILE_SUFFIX	"_gsm_imm_ass"	Suffix for Immediate Assignments file	GSM_IMMASSFILE=1
GSM_IMSIFILE_SUFFIX	"_gsm_imsi"	Suffix for IMSI file	GSM_IMSIFILE=1
GSM_OPFILE_SUFFIX	"_gsm_operators"	Suffix for operators file	GSM_OPFILE=1
GSM_SMSFILE_SUFFIX	"_gsm_sms"	Suffix for SMS file	GSM_SMSFILE=1
GSM_FILES_AMR_EXT	".amr"	File extension for audio files	GSM_SPEECHFILE=1
GSM_FILES_TMP_EXT	".tmp"	File extension for temporary files	
GSM_FILES_TXT_EXT	".txt"	File extension for text files	
GSM_STATFILE_SUFFIX	"_gsm_stats.txt"	Suffix for GSM statistics file	GSM_STATFILE=1

The following flags produce a more verbose output and are mostly useful for debugging:

Name	Default	Description	Flags
GSM_DEBUG_A_RP	0	Print debug messages for A-I/F RP layer	
GSM_DEBUG_A_RP_UNK	0	Report unknown values for A-I/F RP layer	GSM_DEBUG_A_RP=1
GSM_DEBUG_DTAP	0	Print debug messages for A-I/F DTAP layer	
GSM_DEBUG_DTAP_UNK	0	Report unknown values for A-I/F DTAP layer	GSM_DEBUG_DTAP=1
GSM_DEBUG_GSMTAP	0	Print debug messages for GSMTAP layer	
GSM_DEBUG_GSMTAP_UNK	0	Report unknown values for GSMTAP layer	GSM_DEBUG_GSMTAP=1
GSM_DEBUG_LAPD	0	Print debug messages for LAPD layer	
GSM_DEBUG_LAPD_UNK	0	Report unknown values for LAPD layer	GSM_DEBUG_LAPD=1
GSM_DEBUG_LAPDM	0	Print debug messages for LAPDm layer	
GSM_DEBUG_LAPDM_UNK	0	Report unknown values for LAPDm layer	GSM_DEBUG_LAPDM=1
GSM_DEBUG_RSL	0	Print debug messages for RSL layer	
GSM_DEBUG_RSL_UNK	0	Report unknown values for RSL layer	GSM_DEBUG_RSL=1
GSM_DEBUG_SMS	0	Print debug messages for SMS layer	
GSM_DEBUG_SMS_UNK	0	Report unknown values for SMS layer	GSM_DEBUG_SMS=1
GSM_DEBUG	0	Print generic debug messages	
GSM_DEBUG_UNK	0	Report unknown values for other messages	GSM_DEBUG=1

25.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- GSM_RMDIR
- GSM_SPEECH_DIR
- GSM_TXT_DIR
- GSM_ARFCNFILE_SUFFIX

- GSM_CALLFILE_SUFFIX
- GSM_CDFILE_SUFFIX
- GSM_IMMASSFILE_SUFFIX
- GSM_IMSIFILE_SUFFIX
- GSM_OPFILE_SUFFIX
- GSM_SMSFILE_SUFFIX
- GSM_FILES_AMR_EXT
- GSM_FILES_TMP_EXT
- GSM_FILES_TXT_EXT
- GSM_STATFILE_SUFFIX
- GSM_ROTATE_TIME

25.4 Flow File Output

The gsmDecode plugin outputs the following columns:

Column	Type	Description	Flags
gsmStat	H32	Status	
gsmLapdSAPI	U8	LAPD Service Access Point Identifier (SAPI)	
gsmLapdTEI	U8	LAPD Terminal Endpoint Identifier (TEI)	
gsmRs1TN	R(U8)	GSM RSL Timeslot Numbers	
gsmAMRDURATION	FLT	GSM Duration of AMR conversation (seconds)	GSM_SPEECHFILE=1
gsmNumAMRGood_bad	U32_U32	GSM Number of AMR good/bad frames	GSM_SPEECHFILE=1

25.4.1 gsmStat

The gsmStat column is to be interpreted as follows:

gsmStat	Description
0x0000 0001	LAPD Radio Signalling Link (RSL, SAPI 0)
0x0000 0002	LAPD O&M link (SAPI 62)
0x0000 0004	LAPD Layer 2 Management (SAPI 63)
0x0000 0008	RSL Radio Link Layer Management (RLM)
0x0000 0010	RSL Dedicated Channel Management (DCM)
0x0000 0020	RSL Common Channel Management (CCM)
0x0000 0040	RSL TRX Management
0x0000 0080	RSL Location Services
0x0000 0100	RSL ip.access Vendor Specific

gsmStat	Description
0x0000 0200	RSL HUAWEI Paging Extension
0x0000 0400	GSM A-I/F DTAP
0x0000 0800	GSM A-I/F DTAP Call Control (CC)
0x0000 1000	GSM A-I/F DTAP Mobility Management (MM)
0x0000 2000	GSM A-I/F DTAP Radio Resources Management (RR)
0x0000 4000	GSM A-I/F DTAP SMS
0x0000 8000	GSM A-I/F RP
0x0001 0000	GSM SMS TPDU
0x0002 0000	GSM Mobile Application (GSM MAP)
0x0004 0000	AMR speech
0x0008 0000	—
0x0010 0000	Uplink
0x0020 0000	Downlink
0x0040 0000	—
0x0080 0000	—
0x0100 0000	File I/O error
0x0200 0000	—
0x0400 0000	LAPD decoding error
0x0800 0000	LAPDm decoding error
0x1000 0000	RSL decoding error
0x2000 0000	DTAP decoding error
0x4000 0000	SMS decoding error
0x8000 0000	Decoding error

25.4.2 gsmTSC

The gsmTSC (Training Sequence Code) column is to be interpreted as follows:

gsmTSC	Description
0	00100101110000100010010111
1	0010110111011100010110111
2	01000011101110100100001110
3	010001111011101000100011110
4	00011010111001000001101011
5	01001110101100000100111010
6	10100111110110001010011111
7	11101111000100101110111100

25.5 Packet File Output

In packet mode (-s option), the gsmDecode plugin outputs the following columns:

Column	Type	Description	Flags
gsmStat	H32	Status	
gsmLapdSAPI	U8	LAPD Service Access Point Identifier (SAPI)	
gsmLapdTEI	U8	LAPD Terminal Endpoint Identifier (TEI)	
gsmRslMsgType	S	GSM RSL Message type	
gsmRslTN	U8	GSM RSL Timeslot Number	
gsmRslSubCh	U8	GSM RSL Subchannel Number	
gsmRslChannel	S	GSM RSL Channel	
gsmDtApTN	U8	GSM A-I/F DTAP Timeslot Number	
gsmDtApChannel	S	GSM A-I/F DTAP Channel	
gsmHandoverRef	U8	Handover reference	
gsmLAIMCC	S	LAI: Mobile Country Code (MCC)	
gsmLAIMCCCountry	S	LAI: MCC Country	
gsmLAIMNC	S	LAI: Mobile Network Code (MNC)	
gsmLAIMNCOperator	S	LAI: MNC Operator	
gsmLAILAC	H16	LAI: Location Area Code (LAC)	
gsmEncryption	SC	Encryption algorithm	
gsmContent	S	Content (voice or signalling)	
gsmAMRCMR	S	AMR codec mode request (CMR)	GSM_SPEECHFILE=1
gsmAMRFrameType	S	AMR frame type	GSM_SPEECHFILE=1
gsmAMRFrameQ	SC	AMR frame quality	GSM_SPEECHFILE=1

25.6 Plugin Report Output

The following information is reported:

- Aggregated [gsmStat](#)
- Number of GSMTAP packets
- Number of GSM RSL packets
- Number of GSM DTAP packets
- Number of GSM DTAP CC packets
- Number of GSM DTAP MM packets
- Number of GSM DTAP RR packets
- Number of GSM DTAP SMS packets
- Number of GSM DTAP SS packets
- Number of SMS messages

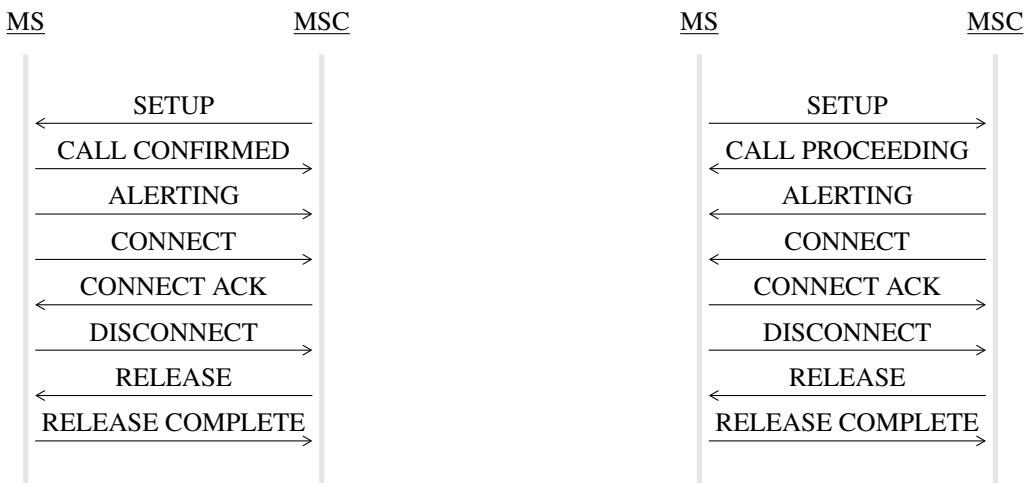
25.7 Additional Output

Non-standard output:

- PREFIX_gsm_arfcn.txt: list of ARFCN, GSM band, up/down frequencies

- PREFIX_gsm_calls.txt: list of calls with numbers, countries, ...
- PREFIX_gsm_channels.txt: list of channels and their content (speech/signalling)
- PREFIX_gsm_imm_ass.txt: list of immediate assignments
- PREFIX_gsm_imsi.txt: list of IMSI/TMSI/IMEI/IMEISV, with manufacturers, models, countries and operators.
- PREFIX_gsm_operators.txt: list of network operators names and time zones
- PREFIX_gsm_sms.txt: list of extracted SMS messages with numbers and countries

25.8 GSM Mobile Terminating and Mobile Originating Call Call Flow Procedures



25.9 Post-Processing

25.9.1 AMR Conversion

The `utils/amr_conv.sh` script can be used to convert extracted AMR conversations to MP3, OGA or WAV files. In addition, the same script can be used to merge two mono AMR files into one stereo MP3, OGA or WAV file. Try `utils/amr_conv.sh --help` for more information

25.9.2 Concatenated SMS messages

The concatenated SMS messages are currently not reassembled. They can be grouped in post-processing with the following `tawk` command:

```
$ tawk 't2rsort(flowInd ";" smsMsgId ";" smsMsgPart)' file_gsm_sms.txt
```

25.10 Acronyms

Acronym	Definition
ACCH	Associated Control Channel
AGCH	Access Grant Channel
AMR	Adaptive Multi-Rate
ARFCN	Absolute Radio-Frequency Channel Number
AuC	Authentication Center
BCCCH	Broadcast Control Channel
BSC	Base Station Controller
BSS	Base Station Subsystem (BTS + BSC)

Acronym	Definition
BTS	Base Transceiver Station
CC	Call Control
CBCH	Cell Broadcast Channel
CCCH	Common Control Channel
CCH	Control Channel
CM	Connection Management
DCCH	Dedicated Control Channel
DL	Downlink
DTAP	Direct Transfer Application Part
EIR	Equipment Identity Register
FACCH	Fast Associated Control Channel
FCCH	Frequency Correction Channel
GSM	Global System for Mobile Communication
HLR	Home Location Register
HSN	Hopping Sequence Number
IMEI	International Mobile Equipment Identity
IMEISV	International Mobile Equipment Identity Software Version
IMSI	International Mobile Subscriber Identity
LAC	Location Area Code
LAI	Location Area Identification
LAPD	Link Access Protocol for D Channel
MAIO	Mobile Allocation Index Offset
MCC	Mobile Country Code
MM	Mobility Management
MNC	Mobile Network Code
MS	Mobile Station
MSC	Mobile Switching Center
NMC	Network Management Center
NSS	Network Subsystem
O&M	Operation & Maintenance
OMC	Operation & Maintenance Center
OMS	Operation & Maintenance Subsystem
PCH	Paging Channel

Acronym	Definition
RACH	Random Access Channel
RR	Radio Resource
RSL	Radio Signalling Link
SACCH	Slow Associated Control Channel
SAPI	Service Access Point Identifier
SC	Service Centre
SCH	Synchronization Channel
SDCCH	Standalone Dedicated Control Channel
SMS	Short Message Service
SMSC	Short Message Service Center
SS	Supplementary Services
TAC	Type Allocation Code
TC	Transcoder
TCH	Traffic Channel
TCH/F	Full Rate Traffic Channel
TCH/H	Half Rate Traffic Channel
TEI	Terminal Endpoint Identifier
TMSI	Temporary Mobile Subscriber Identity
TN	Timeslot Number
TRX	Transceiver
TSC	Training Sequence Code
UL	Uplink
VLR	Visitors Location Register

25.11 References

- [GSM 04.07](#): Mobile radio interface signalling layer 3 general aspects
- [GSM 04.08](#): Mobile radio interface layer 3 specification
- [GSM 08.56](#): BSC-BTS interface layer 2 specification
- [GSM 08.58](#): BSC-BTS interface layer 3 specification

26 httpSniffer

26.1 Description

The httpSniffer plugin processes HTTP header and content information of a flow. The idea is to identify certain HTTP features using flow parameters and to extract certain content such as text or images for further investigation. The httpSniffer plugin requires no dependencies and produces only output to the flow file. User defined compiler switches in *httpSniffer.h* produce optimized code for the specific application.

26.2 Configuration Flags

The flow based output and the extracted information can be controlled by switches and constants listed in the table below. They control the output of host, URL and method counts, names and cookies and the function of content storage.
WARNING: The amount of being stored on disk can be substantial, make sure that the number of concurrent file handles is large enough, use `ulimit -n`.

Name	Default	Description	Flags
HTTP_MIME	1	Mime types	
HTTP_STAT	1	Status codes	
HTTP_MCNT	1	Mime count: GET, POST	
HTTP_HOST	1	Hosts	
HTTP_URL	1	URLs	
HTTP_COOKIE	1	Cookies	
HTTP_IMAGE	1	Image names	
HTTP_VIDEO	1	Video names	
HTTP_AUDIO	1	Audio names	
HTTP_MSG	1	Message names	
HTTP_APPL	1	Application names	
HTTP_TEXT	1	Text names	
HTTP_PUNK	1	POST/else/unknown names	
HTTP_BODY	1	Analyze body and print anomalies	
HTTP_BDURL	1	Refresh and set-cookie URLs	HTTP_BODY=1
HTTP_USRAG	1	User-Agents	
HTTP_XFRWD	1	X-Forwarded-For	
HTTP_REFRR	1	Referer	
HTTP_VIA	1	Via	
HTTP_LOC	1	Location	
HTTP_SERV	1	Server	
HTTP_PWR	1	X-Powered-By	
HTTP_ANTVIR	0	Antivirus Info	
HTTP_AVAST_CID	0	Avast client ID	
HTTP_ESET_UID	0	ESET update ID	
HTTP_STATAGA	1	Aggregate status response	
HTTP_MIMEAGA	1	Aggregate mime response	
HTTP_HOSTAGA	1	Aggregate Hosts	
HTTP_URLAGA	1	Aggregate URLs	

Name	Default	Description	Flags
HTTP_USRAGA	1	Aggregate User-Agents	
HTTP_XFRWDA	1	Aggregate X-Forwarded-For	
HTTP_REFRRA	1	Aggregate Referer	
HTTP_VIAA	1	Aggregate Via	
HTTP_LOCA	1	Aggregate Location	
HTTP_SERVA	1	Aggregate Server	
HTTP_PWRA	1	Aggregate X-Powered-By	
HTTP_SAVE_IMAGE	0	Save all images	
HTTP_SAVE_VIDEO	0	Save all videos	
HTTP_SAVE_AUDIO	0	Save all audios	
HTTP_SAVE_MSG	0	Save all messages	
HTTP_SAVE_TEXT	0	Save all texts	
HTTP_SAVE_APPL	0	Save all applications	
HTTP_SAVE_PUNK	0	Save all else	
HTTP_PUNK_AV_ONLY	0	Save PUT/else only for antivirus	HTTP_SAVE_PUNK=1
HTTP_RMDIR	1	Empty HTTP_*_PATH before starting	HTTP_SAVE=1

Note that `HTTP_SAVE_*` refers to the *Content-Type*, e.g., `HTTP_SAVE_APPL`, will save all payload whose *Content-Type* starts with `application/` (including forms, such as `application/x-www-form-urlencoded`). The maximum memory allocation per item is defined by `HTTP_DATA_C_MAX` listed below. The path of each extracted HTTP content can be set by the `HTTP_XXXX_PATH` constants. HTTP content having no name is assigned a default name defined by `HTTP_NONAME`. Each name is prepended the findex, packet number and an index to facilitate the mapping between flows and its content. The latter constant has to be chosen carefully because for each item (mime, cookie, image, ...), `HTTP_MXFILE_LEN * HTTP_DATA_C_MAX * HASHCHAIANTABLE_SIZE * HASHFACTOR` bytes are allocated.

The filenames are defined as follows:

```
Filename_findex_Flow-Dir(A/B)_{#Packet-in-Flow}_{#Mimetype-in-Flow}
```

So they can easily being matched with the flow or packet file.

Name	Default	Description
HTTP_PATH	"/tmp"	Root path for extracted content
HTTP_IMAGE_PATH	"httpPicture"	Path for pictures
HTTP_VIDEO_PATH	"httpVideo"	Path for videos
HTTP_AUDIO_PATH	"httpAudio"	Path for audios
HTTP_MSG_PATH	"httpMSG"	Path for messages
HTTP_TEXT_PATH	"httpText"	Path for texts
HTTP_APPL_PATH	"httpAppl"	Path for applications
HTTP_PUNK_PATH	"httpPunk"	Path for PUT/else
HTTP_NONAME	"nude1"	File name for unnamed content
HTTP_DATA_C_MAX	20	Maximum dim of all storage array: # / flow
HTTP_CNT_LEN	13	Max # of cnt digits attached to file name
HTTP_FINDEX_LEN	20	String length of findex in decimal format

Name	Default	Description
HTTP_MXFILE_LEN	80	Maximum image name length in bytes
HTTP_MXUA_LEN	400	Maximum User-Agent name length in bytes
HTTP_MXXF_LEN	80	Maximum X-Forward-For name length in bytes
HTTP_AVID_LEN	32	Maximum antivirus client ID length in bytes

26.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- HTTP_RMDIR
- HTTP_PATH
- HTTP_IMAGE_PATH
- HTTP_VIDEO_PATH
- HTTP_AUDIO_PATH
- HTTP_MSG_PATH
- HTTP_TEXT_PATH
- HTTP_APPL_PATH
- HTTP_PUNK_PATH

26.3 Flow File Output

The httpSniffer plugin outputs the following columns:

Column	Type	Description	Flags
httpStat	H16	Status	
httpAFlags	H16	Anomaly flags	
httpMethods	H8	HTTP methods	
httpHeadMimes	H16	HEADMIME-TYPES	
httpCFlags	H8	HTTP content body info	HTTP_BODY=1
httpGet_Post	2U16	Number of GET and POST requests	HTTP_MCNT=1
httpRSCnt	U16	Response status count	HTTP_STAT=1
httpRSCode	RU16	Response status code	HTTP_STAT=1
httpURL_Via_Loc_Srv_ Pwr_UAg_XFr_ Ref_Cky_Mim	10U16	Number of URL, Via, Location, Server, X-Powered-By, User-Agent, X-Forwarded-For, Referer, Cookie and Mime-Type	
httpImg_Vid_Aud_Msg_ Txt_App_Unk	7U16	Number of images, videos, audios, messages, texts, applications and unknown	
httpHosts	RS	Host names	HTTP_HOST=1
httpURL	RS	URLs (including parameters)	HTTP_URL=1
httpMimes	RS	MIME-types	HTTP_MIME=1

Column	Type	Description	Flags
httpCookies	RS	Cookies	HTTP_COOKIE=1
httpImages	RS	Images	HTTP_IMAGE=1
httpVideos	RS	Videos	HTTP_VIDEO=1
httpAudios	RS	Audios	HTTP_AUDIO=1
httpMsgs	RS	Messages	HTTP_MSG=1
httpAppl	RS	Applications	HTTP_APPL=1
httpText	RS	Texts	HTTP_TEXT=1
httpPunk	RS	Payload unknown	HTTP_PUNK=1
httpBdyURL	RS	Body: Refresh, set_cookie URL	HTTP_BODY=1&& HTTP_BDURL=1
httpUsrAg	RS	User-Agent	HTTP_USRAG=1
httpXFor	RS	X-Forwarded-For	HTTP_XFRWD=1
httpRefrr	RS	Referer	HTTP_REFRR=1
httpVia	RS	Via (Proxy)	HTTP_VIA=1
httpLoc	RS	Location (Redirection)	HTTP_LOC=1
httpServ	RS	Server	HTTP_SERV=1
httpPwr	RS	X-Powered-By / Application	HTTP_PWR=1
httpAvastCid	S	Avast client ID	HTTP_AVAST_CID=1
httpEsetUid	S	ESET update ID	HTTP_ESET_UID=1

26.3.1 httpStat

The httpStat column is to be interpreted as follows:

httpStat	Description
2^0 (=0x0001)	Warning: HTTP_DATA_C_MAX entries in flow name array reached
2^1 (=0x0002)	Warning: Filename longer than HTTP_MXFILE_LEN
2^2 (=0x0004)	Internal state: pending URL name
2^3 (=0x0008)	HTTP flow
2^4 (=0x0010)	Internal state: Chunked transfer
2^5 (=0x0020)	Internal state: HTTP flow detected
2^6 (=0x0040)	Internal state: HTTP header parsing in process
2^7 (=0x0080)	Internal state: sequence number init
2^8 (=0x0100)	Internal state: header shift
2^9 (=0x0200)	Internal state: PUT payload sniffing
2^{10} (=0x0400)	Internal state: Image payload sniffing
2^{11} (=0x0800)	Internal state: video payload sniffing
2^{12} (=0x1000)	Internal state: audio payload sniffing
2^{13} (=0x2000)	Internal state: message payload sniffing
2^{14} (=0x4000)	Internal state: text payload sniffing
2^{15} (=0x8000)	Internal state: application payload sniffing

26.3.2 httpAFlags

The `httpAFlags` column denotes HTTP anomalies regarding the protocol and the security. It is to be interpreted as follows:

httpAFlags	Description
2^0 (=0x0001)	Warning: POST query with parameters, possible malware
2^1 (=0x0002)	Warning: Host is IPv4
2^2 (=0x0004)	Warning: Possible DGA
2^3 (=0x0008)	Warning: Mismatched content-type
2^4 (=0x0010)	Warning: Sequence number mangled or error retry detected
2^5 (=0x0020)	Warning: Parse Error
2^6 (=0x0040)	Warning: header without value, e.g., Content-Type: [missing]
2^7 (=0x0080)	—
2^8 (=0x0100)	Info: X-Site Scripting protection
2^9 (=0x0200)	Info: Content Security Policy
2^{10} (=0x0400)	Info: Do not track
2^{11} (=0x0800)	—
2^{12} (=0x1000)	Warning: possible EXE download
2^{13} (=0x2000)	Warning: possible ELF download
2^{14} (=0x4000)	Warning: HTTP 1.0 legacy protocol, often used by malware
2^{15} (=0x8000)	—

26.3.3 httpMethods

The `httpMethods` column is to be interpreted as follows:

httpMethods	Type	Description
2^0 (=0x01)	OPTIONS	Return HTTP methods that server supports for specified URL
2^1 (=0x02)	GET	Request of representation of specified resource
2^2 (=0x04)	HEAD	Request of representation of specified resource without body
2^3 (=0x08)	POST	Request to accept enclosed entity as new subordinate of resource identified by URI
2^4 (=0x10)	PUT	Request to store enclosed entity under supplied URI
2^5 (=0x20)	DELETE	Delete specified resource
2^6 (=0x40)	TRACE	Echo back received request
2^7 (=0x80)	CONNECT	Convert request connection to transparent TCP/IP tunnel

26.3.4 httpHeadMimes

The `httpHeadMimes` column is to be interpreted as follows:

httpHeadMimes	Mime-Type	Description
2^0 (=0x0001)	application	Multi-purpose files: java or postscript, ...
2^1 (=0x0002)	audio	Audio file
2^2 (=0x0004)	image	Image file
2^3 (=0x0008)	message	Instant or email message type
2^4 (=0x0010)	model	3D computer graphics
2^4 (=0x0020)	multipart	Archives and other objects made of more than one part
2^5 (=0x0040)	text	Human-readable text and source code
2^6 (=0x0080)	video	Video stream: Mpeg, Flash, Quicktime, ...
2^8 (=0x0100)	vnd	Vendor-specific files: Word, OpenOffice, ...
2^9 (=0x0200)	x	Non-standard files: tar, SW packages, LATEX, Shockwave Flash, ...
2^{10} (=0x0400)	x-pkcs	public-key cryptography standard files
2^{11} (=0x0800)	—	—
2^{12} (=0x1000)	—	—
2^{13} (=0x2000)	—	—
2^{14} (=0x4000)	—	—
2^{15} (=0x8000)	*	All else

26.3.5 httpCFlags

The httpCFlags column is to be interpreted as follows:

httpCFlags	Description
2^0 (=0x0001)	HTTP set cookie
2^1 (=0x0002)	HTTP refresh detected
2^2 (=0x0004)	Hostname detected
2^3 (=0x0008)	POST Boundary marker
2^4 (=0x0010)	Potential HTTP content
2^5 (=0x0020)	Stream
2^6 (=0x0040)	Quarantine virus upload
2^7 (=0x0080)	Antivirus sample upload
2^8 (=0x0100)	Antivirus Avira detected
2^9 (=0x0200)	Antivirus Avast detected
2^{10} (=0x0400)	Antivirus AVG detected
2^{11} (=0x0800)	Antivirus Bit Defender detected
2^{12} (=0x1000)	Antivirus ESET detected
2^{13} (=0x2000)	Antivirus Microsoft sec detected
2^{14} (=0x4000)	Antivirus Symantec, Norton, ...
2^{15} (=0x8000)	Stream1

26.4 Packet File Output

In packet mode (-s option), the httpSniffer plugin outputs the following columns:

Column	Type	Description	Flags
httpStat	H16	Status	
httpAFlags	H16	Anomaly flags	
httpMethods	H8	HTTP methods	
httpHeadMimes	H16	HEADMIME-TYPES	
httpCFlags	H8	HTTP content body info	HTTP_BODY=1

26.5 Monitoring Output

In monitoring mode, the httpSniffer plugin outputs the following columns:

Column	Type	Description	Flags
httpPkts	U64	Number of HTTP packets	

26.6 Plugin Report Output

The following information is reported:

- Max number of file handles (only if `HTTP_SAVE=1`)
- Number of HTTP packets
- Number of HTTP #GET, #POST, #GET/#POST ratio
- Aggregated [httpStat](#)
- Aggregated [httpHeadMimes](#)
- Aggregated [httpAFlags](#)
- Aggregated [httpCFlags](#) (`HTTP_BODY=1`)

The GET/POST ratio is very helpful in detecting malware operations, if you know the normal ratio of your machines in the network. The file descriptor gives you an indication of the maximum file handles the present pcap will produce. You can increase it by invoking `uname -n mylimit`, but it should not be necessary as we manage the number of handle being open to be always below the max limit.

27 icmpDecode

27.1 Description

The icmpDecode plugin analyzes ICMP and ICMPv6 traffic. It generates global and flow based statistics.

27.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
ICMP_TC_MD	0	Type/code representation: 0: bitfield, 1: explicit array of type code, 2: type code statistics [NOT IMPLEMENTED YET]	
ICMP_NUM	10	Number of type and code information	ICMP_TC_MD=1
ICMP_FDCORR	1	Flow direction correction	
ICMP_PARENT	0	Resolve the parent flow	
ICMP_STATFILE	0	Print ICMP statistics in a separate file	
ICMP_NOCODE	"_"	Symbol to use to represent the absence of a code	

27.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCTRL>0):

- ICMP_NOCODE
- ICMP_SUFFIX

27.3 Flow File Output

The icmpDecode plugin outputs the following columns:

Column	Type	Description	Flags
icmpStat	H8	Status	
icmpTCcnt	U8	Type/Code count	
icmpBFType_Code	H32_H16	Aggregated type (<32) and code bitfield	ICMP_TC_MD=0 && IPV6_ACTIVATE=0
icmpBFTypH_TypL_Code	H32_H32_H16	Aggr. type H(>128), L(<32) and code bitfield	ICMP_TC_MD=0 && IPV6_ACTIVATE=1
icmpType_Code	R(U8_U8)	Type and code fields	ICMP_TC_MD=1
icmpTmGtw	H32	Time/gateway	
icmpEchoSuccRatio	F	Echo reply/request success ratio	
icmpPFindeX	U64	Parent flowIndex	ICMP_PARENT=1

27.3.1 icmpStat

The `icmpStat` column is to be interpreted as follows:

icmpStat	Description
2^0 (=0x01)	Flow is ICMP
2^1 (=0x02)	—
2^2 (=0x04)	—
2^3 (=0x08)	—
2^4 (=0x10)	WANG2 Microsoft bandwidth test
2^5 (=0x20)	ICMP ECHO Seq Num abnormal increment
2^6 (=0x40)	Embedded LOKI covert channel
2^7 (=0x80)	Embedded SSH covert channel

27.3.2 icmpBFType_Code

For **ICMP (IPv4)**, the `icmpBFType_Code` column is to be interpreted as follows:

icmpBFType	Description
2^0 (=0x00000001)	Echo Reply
2^1 (=0x00000002)	—
2^2 (=0x00000004)	—
2^3 (=0x00000008)	Destination Unreachable
2^4 (=0x00000010)	Source Quench
2^5 (=0x00000020)	Redirect (change route)
2^6 (=0x00000040)	Alternate Host Address (Deprecated)
2^7 (=0x00000080)	—
2^8 (=0x00000100)	Echo Request
2^9 (=0x00000200)	Router Advisement
2^{10} (=0x00000400)	Router Selection
2^{11} (=0x00000800)	Time Exceeded
2^{12} (=0x00001000)	Parameter Problem
2^{13} (=0x00002000)	Timestamp Request
2^{14} (=0x00004000)	Timestamp Reply
2^{15} (=0x00008000)	Information Request
2^{16} (=0x00010000)	Information Reply
2^{17} (=0x00020000)	Address Mask Request
2^{18} (=0x00040000)	Address Mask Reply
2^{19} (=0x00080000)	Reserved (for Security)
2^{20} (=0x00100000)	Experimental

icmpBFType	Description
2^{21} (=0x00200000)	Experimental
2^{22} (=0x00400000)	Experimental
2^{23} (=0x00800000)	Experimental
2^{24} (=0x01000000)	Experimental
2^{25} (=0x02000000)	Experimental
2^{26} (=0x04000000)	Experimental
2^{27} (=0x08000000)	Experimental
2^{28} (=0x10000000)	Experimental
2^{29} (=0x20000000)	Experimental
2^{30} (=0x40000000)	Traceroute
2^{31} (=0x80000000)	Datagram Conversion Error (Deprecated)

The `icmpCode` for **Destination Unreachable** (0x00000008) is to be interpreted as follows:

icmpBFCode	Description
2^0 (=0x0001)	Network Unreachable
2^1 (=0x0002)	Host Unreachable
2^2 (=0x0004)	Protocol Unreachable
2^3 (=0x0008)	Port Unreachable
2^4 (=0x0010)	Fragmentation Needed/DF set
2^5 (=0x0020)	Source Route failed
2^6 (=0x0040)	Destination Network unknown
2^7 (=0x0080)	Destination Host unknown
2^8 (=0x0100)	Src host isolated
2^9 (=0x0200)	Dest net administratively prohibited
2^{10} (=0x0400)	Dest host administratively prohibited
2^{11} (=0x0800)	Dest net unreachable for type of service
2^{12} (=0x1000)	Dest host unreachable for type of service
2^{13} (=0x2000)	Communication administratively prohibited
2^{14} (=0x4000)	Precedence violation
2^{15} (=0x8000)	Precedence cut off

For **ICMPv6 (IPv6)**, the `icmpBFType_Code` column is to be interpreted as follows:

icmpType	Description	icmpType	Description
0	Reserved	142	Inverse Neighbor Discovery Advertisement
1	Destination Unreachable	143	Version 2 Multicast Listener Report
2	Packet Too Big	144	Home Agent Address Discovery Request
3	Time Exceeded	145	Home Agent Address Discovery Reply
4	Parameter Problem	146	Mobile Prefix Solicitation
100	Private experimentation	147	Mobile Prefix Advertisement
101	Private experimentation	148	Certification Path Solicitation
102–126	Unassigned	149	Certification Path Advertisement
127	Reserved for expansion of ICMPv6 error messages	150	ICMP messages utilized by experimental mobility protocols such as Seamoby
128	Echo Request	151	Multicast Router Advertisement
129	Echo Reply	152	Multicast Router Solicitation
130	Multicast Listener Query	153	Multicast Router Termination
131	Multicast Listener Report	154	FMIPv6 Messages
132	Multicast Listener Done	155	RPL Control Message
133	Router Solicitation	156	ILNPv6 Locator Update Message
134	Router Advertisement	157	Duplicate Address Request
135	Neighbor Solicitation	158	Duplicate Address Confirmation
136	Neighbor Advertisement	159	MPL Control Message
137	Redirect Message	160–199	Unassigned
138	Router Renumbering	200	Private experimentation
139	ICMP Node Information Query	201	Private experimentation
140	ICMP Node Information Response	255	Reserved for expansion of ICMPv6 informational messages
141	Inverse Neighbor Discovery Solicitation		

The `icmpCode` for **Destination Unreachable (1)** are:

icmpCode	Description
2^0 (=0x0001)	No route to destination
2^1 (=0x0002)	Communication with destination administratively prohibited
2^2 (=0x0004)	Beyond scope of source address
2^3 (=0x0008)	Address unreachable
2^4 (=0x0010)	Port unreachable
2^5 (=0x0020)	Source address failed ingress/egress policy
2^6 (=0x0040)	Reject route to destination
2^7 (=0x0080)	Error in Source Routing Header

The `icmpCode` for **Time Exceeded (3)** are:

icmpCode	Description
2^0 (=0x0001)	Hop limit exceeded in transit
2^1 (=0x0002)	Fragment reassembly time exceeded

The icmpCode for **Parameter Problem (4)** are:

icmpCode	Description
2^0 (=0x0001)	Erroneous header field encountered
2^1 (=0x0002)	Unrecognized Next Header type encountered
2^2 (=0x0004)	Unrecognized IPv6 option encountered
2^3 (=0x0008)	IPv6 First Fragment has incomplete IPv6 Header Chain

The icmpCode for **Router Renumbering (138)** are:

icmpCode	Description
2^0 (=0x0001)	Router Renumbering Command
2^1 (=0x0002)	Router Renumbering Result
255	Sequence Number Reset

The icmpCode for **ICMP Node Information Query (139)** are:

icmpCode	Description
2^0 (=0x0001)	The Data field contains an IPv6 address which is the Subject of this Query
2^1 (=0x0002)	The Data field contains a name which is the Subject of this Query, or is empty, as in the case of a NOOP
2^3 (=0x0004)	The Data field contains an IPv4 address which is the Subject of this Query

The icmpCode for **ICMP Node Information Response (140)** are:

icmpCode	Description
2^0 (=0x0001)	A successful reply. The Reply Data field may or may not be empty
2^1 (=0x0002)	The Responder refuses to supply the answer. The Reply Data field will be empty
2^2 (=0x0004)	The Qtype of the Query is unknown to the Responder. The Reply Data field will be empty

27.4 Packet File Output

In packet mode (-s option), the icmpDecode plugin outputs the following columns:

Column	Type	Description	Flags
icmpStat	H8	Status	
icmpType	U8	Message type	
icmpCode	U8	Message code	
icmpID	H16	Identifier	
icmpSeq	H16	Sequence number	
icmpPFIindex	U64	Parent flowIndex	ICMP_PARENT=1

27.5 Monitoring Output

In monitoring mode, the icmpDecode plugin outputs the following columns:

Column	Type	Description	Flags
icmpPkts	U64	Number of ICMP/ICMPv6 packets	
icmpEchoReq	U64	Number of ICMP/ICMPv6 echo request packets	
icmpEchoRep	U64	Number of ICMP/ICMPv6 echo reply packets	

27.6 Plugin Report Output

The following information is reported:

- Aggregated [icmpStat](#)
- Number of ICMP/ICMPv6 echo request packets
- Number of ICMP/ICMPv6 echo reply packets
- ICMP/ICMPv6 echo reply / request ratio

27.7 Additional Output

The icmpDecode plugin outputs absolute and relative statistics in the `PREFIX_icmpStats.txt` file. Note that the default suffix of “`_icmpStats.txt`” can be changed by editing the `ICMP_SUFFIX` flag.

The output is as follows (`IPV6_ACTIVATE=0 || IPV6_ACTIVATE=2`):

Type	Code	Description
ICMP_ECHOREQUEST	--	Echo request
ICMP_ECHOREPLY	--	Echo reply to an echo request
ICMP_SOURCE_QUENCH	--	Source quenches
ICMP_TRACEROUTE	--	Traceroute packets
ICMP_DEST_UNREACH	ICMP_NET_UNREACH	Network unreachable
ICMP_DEST_UNREACH	ICMP_HOST_UNREACH	Host unreachable
ICMP_DEST_UNREACH	ICMP_PROT_UNREACH	Protocol unreachable
ICMP_DEST_UNREACH	ICMP_PORT_UNREACH	Port unreachable
ICMP_DEST_UNREACH	ICMP_FRAG_NEEDED	Fragmentation needed
ICMP_DEST_UNREACH	ICMP_SR_FAILED	Source route failed
ICMP_DEST_UNREACH	ICMP_NET_UNKNOWN	Network unknown
ICMP_DEST_UNREACH	ICMP_HOST_UNKNOWN	Host unknown
ICMP_DEST_UNREACH	ICMP_HOST_ISOLATED	Host is isolated
ICMP_DEST_UNREACH	ICMP_NET_ANO	Network annotation
ICMP_DEST_UNREACH	ICMP_HOST_ANO	Host annotation
ICMP_DEST_UNREACH	ICMP_NET_UNR_TOS	Unreachable type of network service
ICMP_DEST_UNREACH	ICMP_HOST_UNR_TOS	Unreachable type of host service
ICMP_DEST_UNREACH	ICMP_PKT_FILTERED	Dropped by a filtering device
ICMP_DEST_UNREACH	ICMP_PREC_VIOLATION	Precedence violation

Type	Code	Description
ICMP_DEST_UNREACH	ICMP_PREC_CUTOFF	Precedence cut off
ICMP_REDIRECT	ICMP_REDIR_NET	Network redirection
ICMP_REDIRECT	ICMP_REDIR_HOST	Host redirection
ICMP_REDIRECT	ICMP_REDIR_NETTOS	Network type of service
ICMP_REDIRECT	ICMP_REDIR_HOSTTOS	Host type of service
ICMP_TIME_EXCEEDED	ICMP_EXC_TTL	TTL exceeded in Transit
ICMP_TIME_EXCEEDED	ICMP_EXC_FRAGTIME	Fragment Reassembly Time Exceeded

If IPV6_ACTIVATE>0, then the output becomes:

Type	Code	Description
ICMP6_ECHOREQUEST	--	Echo request
ICMP6_ECHOREPLY	--	Echo reply to an echo request
ICMP6_PKT_TOO_BIG	--	Packet too big
ICMP6_DEST_UNREACH	ICMP6_NO_ROUTE	No route to destination
ICMP6_DEST_UNREACH	ICMP6_COMM_PROHIBIT	Communication with destination prohibited
ICMP6_DEST_UNREACH	ICMP6_BEYOND_SCOPE	Beyond scope of source address
ICMP6_DEST_UNREACH	ICMP6_ADDR_UNREACH	Address unreachable
ICMP6_DEST_UNREACH	ICMP6_PORT_UNREACH	Port unreachable
ICMP6_DEST_UNREACH	ICMP6_SR_FAILED	Source route failed
ICMP6_DEST_UNREACH	ICMP6_REJECT	Reject source to destination
ICMP6_DEST_UNREACH	ICMP6_ERROR_HDR	Error in Source Routing Header
ICMP6_TIME_EXCEEDED	ICMP6_EXC_HOPS	Hop limit exceeded in transit
ICMP6_TIME_EXCEEDED	ICMP6_EXC_FRAGTIME	Fragment reassembly time exceeded
ICMP6_PARAM_PROBLEM	ICMP6_ERR_HDR	Erroneous header field
ICMP6_PARAM_PROBLEM	ICMP6_UNRECO_NEXT_HDR	Unrecognized Next Header type
ICMP6_PARAM_PROBLEM	ICMP6_UNRECO_IP6_OPT	Unrecognized IPv6 option
ICMP6_MCAST_QUERY	--	Multicast Listener Query
ICMP6_MCAST REP	--	Multicast Listener Report
ICMP6_MCAST_DONE	--	Multicast Listener Done
ICMP6_RTER_SOLICIT	--	Router Solicitation
ICMP6_RTER_ADVERT	--	Router Advertisement
ICMP6_NBOR_SOLICIT	--	Neighbor Solicitation
ICMP6_NBOR_ADVERT	--	Neighbor Advertisement
ICMP6_REDIRECT_MSG	--	Redirect Message
ICMP6_RTER_RENUM	ICMP6_RR_CMD (0)	Router Renumbering Command
ICMP6_RTER_RENUM	ICMP6_RR_RES (1)	Router Renumbering Result
ICMP6_RTER_RENUM	ICMP6_RR_RST (255)	Router Renum.: Sequence Number Reset
ICMP6_NODE_INFO_QUERY	ICMP6_NIQ_IP6 (0)	Node Info. Query: contains an IPv6 address
ICMP6_NODE_INFO_QUERY	ICMP6_NIQ_NAME (1)	Contains a name or is empty (NOOP)
ICMP6_NODE_INFO_QUERY	ICMP6_NIQ_IP4 (2)	Contains an IPv4 address
ICMP6_NODE_INFO_RESP	ICMP6_NIR_SUCC (0)	Node Info. Response: Successful reply
ICMP6_NODE_INFO_RESP	ICMP6_NIR_DENIED (1)	Responder refuses to answer
ICMP6_NODE_INFO_RESP	ICMP6_NIR_UNKN (2)	Qtype of the query unknown
ICMP6_INV_NBOR_DSM	--	Inverse Neighbor Discovery Solicitation Msg

Type	Code	Description
ICMP6_INV_NBOR_DAM	--	Inverse Neighbor Disc. Advertisement Msg
ICMP6_MLD2	--	Version 2 Multicast Listener Report
ICMP6_ADDR_DISC_REQ	--	Home Agent Address Discovery Request Msg
ICMP6_ADDR_DISC REP	--	Home Agent Address Discovery Reply Msg
ICMP6_MOB_PREF_SOL	--	Mobile Prefix Solicitation
ICMP6_MOB_PREF_ADV	--	Mobile Prefix Advertisement
ICMP6_CERT_PATH_SOL	--	Certification Path Solicitation Message
ICMP6_CERT_PATH_ADV	--	Certification Path Advertisement Message
ICMP6_EXP_MOBI	--	Experimental mobility protocols
ICMP6_MRDAV	--	Multicast Router Advertisement
ICMP6_MRDSOL	--	Multicast Router Solicitation
ICMP6_MRDTERM	--	Multicast Router Termination
ICMP6_FMIPv6	--	FMIPv6 Messages
ICMP6_RPL_CTRL	--	RPL Control Message
ICMP6_ILNP_LOC_UP	--	ILNPv6 Locator Update Message
ICMP6_DUP_ADDR_REQ	--	Duplicate Address Request
ICMP6_DUP_ADDR_CONF	--	Duplicate Address Confirmation

27.8 Post-Processing

27.8.1 icmpX

The `icmpX` script extracts all ICMP flows and their parents (flows which caused the ICMP message) from a flow file. Run `./icmpX --help` for more information.

27.8.2 protStat

The `protStat` script can be used to sort the `PREFIX_icmpStats.txt` file for the most or least occurring types and codes. It can output the top or bottom *N* protocols or only those with at least a given percentage:

- list all the options: `protStat --help`
- for better readability, use `protStat` with `tcol`: `protStat ... | tcol`
- sorted list of types (by packets):

```
protStat PREFIX_icmpStats.txt
```

- top 10 ICMP types and codes (by packets):

```
protStat PREFIX_icmpStats.txt -n 10 -icmp
```

- bottom 5 ICMPv6 types and codes (by packets):

```
protStat PREFIX_icmpStats.txt -n -5 -icmpv6
```

- ICMP and ICMPv6 types and codes with packets percentage greater than 20%:

```
protStat PREFIX_icmpStats.txt -p 20
```

- ICMP and ICMPv6 types and codes with packets percentage smaller than 5%:

```
protStat PREFIX_icmpStats.txt -p -5
```

28 igmpDecode

28.1 Description

This plugin analyzes IGMP traffic and provides absolute and relative statistics to the `PREFIX_igmpStats.txt` file.

28.2 Required Files

None

28.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
<code>IGMP_STATFILE</code>	1	Print IGMP statistics in a separate file
<code>IGMP_NOCODE</code>	"_"	Symbol to use to represent the absence of a code
<code>IGMP_SUFFIX</code>	<code>"_igmpStats.txt"</code>	Suffix for output file

28.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- `IGMP_NOCODE`
- `IGMP_SUFFIX`

28.4 Flow File Output

The `igmpDecode` plugin outputs the following columns:

Column	Type	Description	Flags
<code>igmpStat</code>	H8	Status	
<code>igmpVersion</code>	RI8	Version	
<code>igmpAType</code>	H32	Aggregated type	
<code>igmpMCastAddr</code>	IP4	Multicast address	
<code>igmpNRec</code>	U16	Number of records	

28.4.1 igmpStat

The `igmpStat` column is to be interpreted as follows:

igmpStat	Description
2^0 (=0x01)	IGMP message had invalid length
2^1 (=0x02)	IGMP message had invalid checksum
2^2 (=0x04)	IGMP message had invalid TTL ($\neq 1$)

igmpStat	Description
2 ³ (=0x08)	IGMP message was invalid for other reasons

28.5 Plugin Report Output

The following information is reported:

- Aggregated [igmpStat](#)
- Number of IGMP packets
- Number of IGMP queries
- Number of IGMP reports
- IGMP query / report ratio

28.6 Additional Output

The plugin exports global statistics about IGMP traffic in the `PREFIX_igmpStats.txt` file. Note that the default suffix of “`_igmpStats.txt`” can be changed by editing the `IGMP_SUFFIX` flag.

28.7 Post-Processing

The `protStat` script can be used to sort the `PREFIX_igmpStats.txt` file for the most or least occurring types and codes. It can output the top or bottom *N* protocols or only those with at least a given percentage:

- list all the options: `protStat --help`
- for better readability, use `protStat` with `tcol`: `protStat ... | tcol`
- sorted list of types (by packets): `protStat PREFIX_igmpStats.txt`
- top 10 IGMP types and codes (by packets): `protStat PREFIX_igmpStats.txt -n 10`
- bottom 5 IGMP types and codes (by packets): `protStat PREFIX_igmpStats.txt -n -5`
- IGMP types and codes with packets percentage greater than 20%: `protStat PREFIX_igmpStats.txt -p 20`
- IGMP types and codes with packets percentage smaller than 5%: `protStat PREFIX_igmpStats.txt -p -5`

29 ircDecode

29.1 Description

The ircDecode plugin analyzes IRC traffic. User defined compiler switches are in *ircDecode.h*.

29.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
IRC_SAVE	0	Save content to <code>IRC_F_PATH</code>	
IRC_RMDIR	1	Empty <code>IRC_F_PATH</code> before starting	IRC_SAVE=1
IRC_CMD_AGGR	1	Aggregate IRC commands/response codes	
IRC_BITFIELD	0	Bitfield coding of IRC commands	
IRC_UXNMLN	10	maximal username length	
IRC_PXNMLN	10	maximal password length	
IRC_NXNMLN	10	maximal nickname length	
IRC_MXNMLN	50	maximal name length	
IRC_MAXUNM	5	Maximal number of users	
IRC_MAXPNM	5	Maximal number of passwords	
IRC_MAXNNM	5	Maximal number of nicknames	
IRC_MAXCNM	20	Maximal number of parameters	
IRC_F_PATH	<code>"/tmp/IRCFILES/"</code>	Path for extracted content	

29.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCTRL>0`):

- `IRC_RMDIR`
- `IRC_F_PATH`

29.3 Flow File Output

The ircDecode plugin outputs the following columns:

Column	Type	Description	Flags
<code>ircStat</code>	H8	Status	
<code>ircCBF</code>	H64	Commands	IRC_BITFIELD=1
<code>ircCC</code>	RSC	Command codes	
<code>ircRC</code>	RU16	Response codes	
<code>ircNumUser</code>	U8	Number of users	
<code>ircUser</code>	RS	Users	
<code>ircNumPass</code>	U8	Number of passwords	
<code>ircPass</code>	RS	Passwords	
<code>ircNumNick</code>	U8	Number of nicknames	

Column	Type	Description	Flags
ircNick	RS	Nicknames	
ircNumC	U8	Number of parameters	
ircC	RS	Content	

29.3.1 ircStat

The `ircStat` column is to be interpreted as follows:

ircStat	Description
2^0 (=0x01)	IRC port found
2^1 (=0x02)	IRC registration successful
2^2 (=0x04)	IRC password incorrect
2^3 (=0x08)	—
2^4 (=0x10)	Unrecognized IRC command
2^5 (=0x20)	File error (<code>IRC_SAVE=1</code>)
2^6 (=0x40)	Array string or filename overflow
2^7 (=0x80)	Invalid format or parsing error

29.3.2 ircCBF

The ircCBF column is to be interpreted as follows:

ircCBF	Description	ircCBF	Description
2 ⁰ (=0x0000.0000.0000.0001)	ADMIN	2 ³² (=0x0000.0001.0000.0000)	PRIVMSG
2 ¹ (=0x0000.0000.0000.0002)	AWAY	2 ³³ (=0x0000.0002.0000.0000)	QUIT
2 ² (=0x0000.0000.0000.0004)	CAP	2 ³⁴ (=0x0000.0004.0000.0000)	REHASH
2 ³ (=0x0000.0000.0000.0008)	CNOTICE	2 ³⁵ (=0x0000.0008.0000.0000)	RESTART
2 ⁴ (=0x0000.0000.0000.0010)	CONNECT	2 ³⁶ (=0x0000.0010.0000.0000)	RULES
2 ⁵ (=0x0000.0000.0000.0020)	CPRIVMSG	2 ³⁷ (=0x0000.0020.0000.0000)	SERVER
2 ⁶ (=0x0000.0000.0000.0040)	DIE	2 ³⁸ (=0x0000.0040.0000.0000)	SERVICE
2 ⁷ (=0x0000.0000.0000.0080)	ENCAP	2 ³⁹ (=0x0000.0080.0000.0000)	SERVLIST
2 ⁸ (=0x0000.0000.0000.0100)	ERROR	2 ⁴⁰ (=0x0000.0100.0000.0000)	SETNAME
2 ⁹ (=0x0000.0000.0000.0200)	HELP	2 ⁴¹ (=0x0000.0200.0000.0000)	SILENCE
2 ¹⁰ (=0x0000.0000.0000.0400)	INFO	2 ⁴² (=0x0000.0400.0000.0000)	SQUERY
2 ¹¹ (=0x0000.0000.0000.0800)	INVITE	2 ⁴³ (=0x0000.0800.0000.0000)	SQUIT
2 ¹² (=0x0000.0000.0000.1000)	ISON	2 ⁴⁴ (=0x0000.1000.0000.0000)	STATS
2 ¹³ (=0x0000.0000.0000.2000)	JOIN	2 ⁴⁵ (=0x0000.2000.0000.0000)	SUMMON
2 ¹⁴ (=0x0000.0000.0000.4000)	KICK	2 ⁴⁶ (=0x0000.4000.0000.0000)	TIME
2 ¹⁵ (=0x0000.0000.0000.8000)	KILL	2 ⁴⁷ (=0x0000.8000.0000.0000)	TOPIC
2 ¹⁶ (=0x0000.0000.0001.0000)	KNOCK	2 ⁴⁸ (=0x0001.0000.0000.0000)	TRACE
2 ¹⁷ (=0x0000.0000.0002.0000)	LINKS	2 ⁴⁹ (=0x0002.0000.0000.0000)	UHNAMES
2 ¹⁸ (=0x0000.0000.0004.0000)	LIST	2 ⁵⁰ (=0x0004.0000.0000.0000)	USER
2 ¹⁹ (=0x0000.0000.0008.0000)	LUSERS	2 ⁵¹ (=0x0008.0000.0000.0000)	USERHOST
2 ²⁰ (=0x0000.0000.0010.0000)	MODE	2 ⁵² (=0x0010.0000.0000.0000)	USERIP
2 ²¹ (=0x0000.0000.0020.0000)	MOTD	2 ⁵³ (=0x0020.0000.0000.0000)	USERS
2 ²² (=0x0000.0000.0040.0000)	NAMES	2 ⁵⁴ (=0x0040.0000.0000.0000)	VERSION
2 ²³ (=0x0000.0000.0080.0000)	NAMESX	2 ⁵⁵ (=0x0080.0000.0000.0000)	WALLOPS
2 ²⁴ (=0x0000.0000.0100.0000)	NICK	2 ⁵⁶ (=0x0100.0000.0000.0000)	WATCH
2 ²⁵ (=0x0000.0000.0200.0000)	NJOIN	2 ⁵⁷ (=0x0200.0000.0000.0000)	WHO
2 ²⁶ (=0x0000.0000.0400.0000)	NOTICE	2 ⁵⁸ (=0x0400.0000.0000.0000)	WHOIS
2 ²⁷ (=0x0000.0000.0800.0000)	OPER	2 ⁵⁹ (=0x0800.0000.0000.0000)	WHOWAS
2 ²⁸ (=0x0000.0000.1000.0000)	PART	2 ⁶⁰ (=0x1000.0000.0000.0000)	-
2 ²⁹ (=0x0000.0000.2000.0000)	PASS	2 ⁶¹ (=0x2000.0000.0000.0000)	-
2 ³⁰ (=0x0000.0000.4000.0000)	PING	2 ⁶² (=0x4000.0000.0000.0000)	-
2 ³¹ (=0x0000.0000.8000.0000)	PONG	2 ⁶³ (=0x8000.0000.0000.0000)	Not supported

29.4 Plugin Report Output

The following information is reported:

- Aggregated [ircStat](#)

- Number of IRC packets

30 jsonSink

30.1 Description

The jsonSink plugin generates JSON output in a file `PREFIX_flows.json`, where `PREFIX` is provided via Tranalyzer `-w` or `-W` option.

30.2 Dependencies

30.2.1 External Libraries

If gzip compression is activated (`JSON_GZ_COMPRESS=1`), then **zlib** must be installed.

JSON_GZ_COMPRESS=1		
Ubuntu:	<code>sudo apt-get install zlib1g-dev</code>	
Arch:	<code>sudo pacman -S zlib</code>	
Gentoo:	<code>sudo emerge zlib</code>	
openSUSE:	<code>sudo zypper install zlib-devel</code>	
Red Hat/Fedora ¹³ :	<code>sudo dnf install zlib-devel</code>	
macOS ¹⁴ :	<code>brew install zlib</code>	

30.2.2 Core Configuration

This plugin requires the following core configuration:

- `$T2HOME/tranalyzer2/src/tranalyzer.h`:
 - `BLOCK_BUF=0`

30.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
<code>JSON_SOCKET_ON</code>	0	Output to a socket (1) or to a file (0)	
<code>JSON_SOCKET_ADDR</code>	"127.0.0.1"	Address of the socket	<code>JSON_SOCKET_ON=1</code>
<code>JSON_SOCKET_PORT</code>	5000	Port of the socket	<code>JSON_SOCKET_ON=1</code>
<code>JSON_GZ_COMPRESS</code>	0	Compress (gzip) the output	
<code>JSON_SPLIT</code>	1	Split the output file (Tranalyzer <code>-W</code> option)	<code>JSON_SOCKET_ON=0</code>
<code>JSON_ROOT_NODE</code>	0	Add a root node (array)	
<code>JSON_SUPPRESS_EMPTY_ARRAY</code>	1	Do not output empty fields	
<code>JSON_NO_SPACES</code>	1	Suppress unnecessary spaces	

¹³If the `dnf` command could not be found, try with `yum` instead

¹⁴Brew is a packet manager for macOS that can be found here: <https://brew.sh>

Name	Default	Description	Flags
JSON_BUFFER_SIZE	1048576	Size of output buffer	
JSON_SELECT	0	Only output specific fields	
JSON_SELECT_FILE	"json-columns.txt"	Filename of the field selector (one column name per line)	JSON_SELECT=1
JSON_SUFFIX	_flows.json"	Suffix for output file	JSON_SOCKET_ON=0

30.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCTRL>0):

- JSON_BUFFER_SIZE
- JSON_SOCKET_ADDR (require JSON_SOCKET_ON=1)
- JSON_SOCKET_PORT (require JSON_SOCKET_ON=1)
- JSON_SUFFIX (require JSON_SOCKET_ON=0)
- JSON_SELECT_FILE

30.4 Plugin Report Output

The following information is reported:

- Number of flows discarded due to main buffer problems

30.5 Custom File Output

- PREFIX_flows.json: JSON representation of Tranalyzer output

30.6 Output Selected Fields Only

When JSON_SELECT=1, the columns to output can be customized with the help of JSON_SELECT_FILE. The filename defaults to json-columns.txt in the user plugin folder, e.g., `~/.tranalyzer/plugins`. The format of the file is simply one field name per line with lines starting with a '#' being ignored. For example, to only output source and destination addresses and ports, create the following file:

```
# Lines starting with a '#' are ignored and can be used to add comments
srcIP
srcPort
dstIP
dstPort
```

30.7 Example

To send compressed data over a socket (JSON_SOCKET_ON=1 and JSON_GZ_COMPRESS=1):

1. nc -l 127.0.0.1 5000 | gunzip
2. tranalyzer -r file.pcap

31 kafkaSink

31.1 Description

The kafkaSink plugin outputs flows to an Apache Kafka event streaming platform.

31.2 Dependencies

31.2.1 External Libraries

This plugin depends on the **librdkafka** library.

OPT2=1		
Ubuntu:	sudo apt-get install	librdkafka-dev
Arch:	sudo pacman -S	librdkafka
Gentoo:	sudo emerge	librdkafka
openSUSE:	sudo zypper install	librdkafka-devel
Red Hat/Fedora¹⁵:	sudo dnf install	librdkafka-devel
macOS¹⁶:	brew install	librdkafka

31.2.2 Core Configuration

This plugin requires the following core configuration:

- \$T2HOME/tralyzer2/src/tralyzer.h:
 - BLOCK_BUF=0

31.3 Services Initialization

The kafkaSink plugin requires a ZooKeeper and a Kafka broker service running on KAFKA_BROKERS (default address and port are 127.0.0.1:9092).

```
# Start the ZooKeeper server and send it to the background
$ zookeeper-server-start.sh /etc/kafka/zookeeper.properties &

# Start the Kafka server and send it to the background
$ kafka-server-start.sh /etc/kafka/server.properties &
```

¹⁵If the `dnf` command could not be found, try with `yum` instead

¹⁶Brew is a packet manager for macOS that can be found here: <https://brew.sh>

31.4 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
KAFKA_BROKERS	"127.0.0.1:9092"	Broker address(es) (comma separated list of host[:port])
KAFKA_TOPIC	"tranalyzer.flows"	Topic to produce to
KAFKA_PARTITION	-1	Target partition: ≥ 0: fixed partition -1: automatic partitioning (unassigned)
KAFKA_RETRIES	3	Max. number of retries when message production failed [0 - 255]
KAFKA_DEBUG	0	Print debug messages

31.4.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- KAFKA_BROKERS
- KAFKA_TOPIC
- KAFKA_PARTITION

31.5 Plugin Report Output

The following information is reported:

- Number of flows discarded

31.6 Example

In this example, the flows will be sent to Kafka to the `tranalyzer.flows` topic. In addition, Tranalyzer information ([INF]) and warnings ([WRN]) will be sent to the `tranalyzer.out`, while errors ([ERR]) will use the `tranalyzer.err` topic.

First, we want to prevent Tranalyzer from coloring the output:

```
$ t2conf tranalyzer2 -D T2_LOG_COLOR=0
```

In this example, only the `basicFlow` and `kafkaSink` plugins will be used:

```
$ t2build tranalyzer2 basicFlow kafkaSink
```

Now, the fun part! Run Tranalyzer as per usual, but redirect `stdout` and `stderr` to a `kcat`¹⁷ process, which will send the data to Kafka:

¹⁷`kcat` was formerly known as `kafkacat`

```
$ t2 -r file.pcap \
 1> >(grep -F -e "[INF]" -e "[WRN]" | kcat -P -b 127.0.0.1:9092 -t tranalyzer.out) \
 2> >(kcat -P -b 127.0.0.1:9092 -t tranalyzer.err)
```

The messages can now be consumed with the help of kafka-console-consumer.

```
# Consume messages for tranalyzer.flows topic
```

```
$ kafka-console-consumer \
  --bootstrap-server localhost:9092 \
  --from-beginning \
  --topic tranalyzer.flows
```

```
# Consume messages for tranalyzer.out topic
```

```
$ kafka-console-consumer \
  --bootstrap-server localhost:9092 \
  --from-beginning \
  --topic tranalyzer.out
```

```
# Consume messages for tranalyzer.err topic
```

```
$ kafka-console-consumer \
  --bootstrap-server localhost:9092 \
  --from-beginning \
  --topic tranalyzer.err
```

32 liveXtr

32.1 Description

This plugin extracts all flows which have the `LIVEXTR` bit set in their `flow_t.status`. This bit can be set by any other plugin wanting to extract traffic. The traffic is stored in a round-robin buffer, this allows to extract packets processed before the `LIVEXTR` bit was set. This plugin can be used when T-Analyzer processes PCAP files and also when it listens on an interface.

32.2 Dependencies

The `liveXtr` plugin depends on the libpcap.

32.3 Configuration Flags

The following flags can be used to control the behavior of the plugin:

Name	Default	Description	Flags
<code>LIVEXTR_BUFSIZE</code>	2^{31} (2 GB)	Size of the round-robin buffer [byte]	
<code>LIVEXTR_MEMORY</code>	1	0: store round-robin buffer in a file, 1: store it in memory	
<code>LIVEXTR_SPLIT</code>	1	Split the output PCAP with t2 -W option	
<code>LIVEXTR_FILE</code>	<code>"/tmp/livextr.data"</code>	Round-robin buffer file location	<code>LIVEXTR_MEMORY=0</code>

32.4 Plugin Report Output

The following information is reported:

- Number of extracted packets

32.5 Additional Output

Non-standard output:

- `PREFIX_livextr.pcap`: PCAP containing the extracted packets.

33 lldpDecode

33.1 Description

The lldpDecode plugin analyzes LLDP traffic.

33.2 Dependencies

33.2.1 Core Configuration

This plugin requires the following core configuration:

- \$T2HOME/tranalyzer2/src/networkHeaders.h:
 - ETH_ACTIVATE>0

33.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
LLDP_TTL_AGGR	1	Aggregate TTL values
LLDP_NUM_TTL	8	Number of different TTL values to store
LLDP_OPT_TLV	1	Output optional TLVs info
LLDP_STRLEN	20	Maximum length of short strings to store
LLDP_LSTRLEN	100	Maximum length of long strings to store

33.4 Flow File Output

The lldpDecode plugin outputs the following columns:

Column	Type	Description	Flags
lldpStat	H16	Status	
lldpTTL	R(U16)	Time To Live (sec)	
lldpTLVTypes	H32	TLV types	
lldpChassis	SC	Chassis ID	
lldpPort	S	Port ID	
lldpPortDesc	S	Port description	LLDP_OPT_TLV=1
lldpSysName	S	System name	LLDP_OPT_TLV=1
lldpSysDesc	S	System description	LLDP_OPT_TLV=1
lldpCaps_enCaps	H16_H16	Supported and enabled capabilities	LLDP_OPT_TLV=1
lldpMngmtAddr	SC	Management address	LLDP_OPT_TLV=1

33.4.1 lldpStat

The lldpStat column is to be interpreted as follows:

lldpStat	Description
0x0001	Flow is LLDP
0x0002	Mandatory TLV missing
0x0004	Optional TLV present
0x0008	Reserved TLV type/subtype used
0x0010	Organization specific TLV used
0x0020	Unhandled TLV used
0x0040	Invalid TLV length
0x0080	—
0x0100	—
0x0200	—
0x0400	—
0x0800	—
0x1000	—
0x2000	String truncated...increase LLDP_STRLEN
0x4000	Too many TTL...increase LLDP_NUM_TTL
0x8000	Snapped payload

33.4.2 lldpTLVTypes

The `lldpTLVTypes` column is to be interpreted as follows:

lldpTLVTypes	Description
2^0 (=0x0000 0001)	End of LLDPDU
2^1 (=0x0000 0002)	Chassis ID
2^2 (=0x0000 0004)	Port ID
2^3 (=0x0000 0008)	Time To Live (sec)
2^4 (=0x0000 0010)	Port description
2^5 (=0x0000 0020)	System name
2^6 (=0x0000 0040)	System description
2^7 (=0x0000 0080)	System capabilities
2^8 (=0x0000 0100)	Management address
TLV types 9 to 126 are reserved	
2^{31} (=0x8000 0000)	TLV type ≥ 31

33.4.3 lldpCaps_enCaps

The `lldpCaps_enCaps` column is to be interpreted as follows:

lldpCaps/lldpEnCaps	Description
0x0001	Other
0x0002	Repeater
0x0004	Bridge
0x0008	WLAN access point
0x0010	Router
0x0020	Telephone
0x0040	DOCSIS cable device
0x0080	Station only
0x0100–0x8000	Reserved

33.5 Packet File Output

In packet mode (-s option), the lldpDecode plugin outputs the following columns:

Column	Type	Description	Flags
lldpStat	H16	Status	
lldpTTL	U16	Time To Live (sec)	
lldpTLVTypes	H32	TLV types	
lldpChassis	SC	Chassis ID	
lldpPort	SC	Port ID	
lldpPortDesc	SC	Port description	LLDP_OPT_TLV=1
lldpSysName	SC	System name	LLDP_OPT_TLV=1
lldpCaps_enCaps	H16_H16	Supported and enabled capabilities	LLDP_OPT_TLV=1
lldpMngmtAddr	SC	Management address	LLDP_OPT_TLV=1

33.6 Monitoring Output

In monitoring mode, the lldpDecode plugin outputs the following columns:

Column	Type	Description	Flags
lldpPkts	U64	Number of LLDP packets	

33.7 Plugin Report Output

The following information is reported:

- Aggregated `lldpStat`
- Aggregated `lldpTLVTypes`
- Aggregated `lldpCaps_enCaps`
- Number of LLDP packets

34 macRecorder

34.1 Description

The macRecorder plugin provides the source- and destination MAC address as well as the number of packets detected in the flow separated by an underscore. If there is more than one combination of MAC addresses, e.g., due to load balancing or router misconfiguration, the plugin prints all recognized MAC addresses separated by semicolons. The number of distinct source- and destination MAC addresses can be output by activating the `MR_NPAIRS` flag. The `MR_MANUF` flags controls the output of the manufacturers for the source and destination addresses. The representation of MAC addresses can be altered using the `MR_MAC_FMT` flag.

34.2 Dependencies

34.2.1 Required Files

The file `manuf.txt` is required if `MR_MANUF > 0` and file `maclbl.txt` is required if `MR_MACLBL > 0`.

34.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
<code>MR_MAC_FMT</code>	1	Format for MAC addresses: 0: hex, 1: mac, 2: int	
<code>MR_NPAIRS</code>	1	Report number of distinct MAC/IP pairs	
<code>MR_MACLBL</code>	2	Format for MAC addresses labels 0: no labels, 1: numerical (int), 2: short Organization 3: full Organization	
<code>MR_MAX_MAC</code>	16	Max number of MAC addresses per flow	
<code>MR_NO_MANUF</code>	" "	Representation of unknown manufacturers	

In addition, the following flags can be found in `macLbl.h`:

Name	Default	Description	Flags
<code>MAC_SORGLEN</code>	12	Maximum length for 'who' information (short version)	
<code>MAC_ORGLEN</code>	44	Maximum length for 'who' information (long version)	

Note that the name of the MAC label file to load can be controlled with `MACLBLFILE` in `macLbl.h`.

34.4 Flow File Output

The macRecorder plugin outputs the following columns:

Column	Type	Description	Flags
macStat	H8	Status	
macPairs	U32	Number of distinct src/dst MAC addresses pairs	MR_NPAIRS=1
srcMac_dstMac_numP	R(H64_H64_U64)	Src/Dst MAC addresses, number of packets	MR_MAC_FMT=0
srcMac_dstMac_numP	R(MAC_MAC_U64)	Src/Dst MAC addresses, number of packets	MR_MAC_FMT=1
srcMac_dstMac_numP	R(U64_U64_U64)	Src/Dst MAC addresses, number of packets	MR_MAC_FMT=2
srcMacLbl_dstMacLbl	R(U32_U32)	Src/Dst MAC label (numerical)	MR_MACLBL=1
srcMacLbl_dstMacLbl	R(SC_SC)	Src/Dst MAC label (string_class)	MR_MACLBL=2
srcMacLbl_dstMacLbl	R(S_S)	Src/Dst MAC label (string)	MR_MACLBL=3

34.4.1 macStat

The `macStat` column is to be interpreted as follows:

macStat	Description
0x01	MAC list overflow... increase <code>MR_MAX_MAC</code>

34.5 Packet File Output

In packet mode (-s option), the `macRecorder` plugin outputs the following columns:

Column	Type	Description	Flags
srcMacLbl	S	Source MAC label	MR_MACLBL>0
dstMacLbl	S	Destination MAC label	MR_MACLBL>0

34.6 Plugin Report Output

The following information is reported:

- Aggregated `macStat`
- MAC pairs per flow: min, max, average

34.7 Example Output

Consider a host with MAC address aa:aa:aa:aa:aa:aa in a local network requesting a website from a public server. Due to load balancing, the opposite flow can be split and transmitted via two routers with MAC addresses bb:bb:bb:bb:bb:bb and cc:cc:cc:cc:cc:cc. The `macRecorder` plugin then produces the following output in default configuration:

```
bb:bb:bb:bb:bb_aa:aa:aa:aa:aa_667;cc:cc:cc:cc:cc_aa:aa:aa:aa:aa_666
```

35 mndpDecode

35.1 Description

The mndpDecode plugin analyzes MNDP traffic.

35.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
MNDP_DEBUG	0	Print debug messages	
MNDP_LSTLEN	5	Max number of elements for lists	
MNDP_STRLEN	32	Max length for strings	

35.3 Flow File Output

The mndpDecode plugin outputs the following columns:

Column	Type	Description	Flags
mndpStat	H8	Status	
mndpMAC	R(MAC)	MAC-Address	MNDP_LSTLEN>0
mndpIdentity	R(STR)	Identity	MNDP_LSTLEN>0
mndpVersion	R(STR)	Version	MNDP_LSTLEN>0
mndpPlatform	R(STR)	Platform	MNDP_LSTLEN>0
mndpSoftwareID	R(STR)	Software-ID	MNDP_LSTLEN>0
mndpBoard	R(STR)	Board	MNDP_LSTLEN>0
mndpUnpack	R(U8)	Packet compression type	MNDP_LSTLEN>0
mndpIface	R(STR)	Interface name	MNDP_LSTLEN>0
mndpIPv4	R(IP4)	IPv4-Address	MNDP_LSTLEN>0
mndpIPv6	R(IP6)	IPv6-Address	MNDP_LSTLEN>0

35.3.1 mndpStat

The mndpStat column is to be interpreted as follows:

mndpStat	Description
0x01	Flow is MNDP
0x02	IPv4 address
0x04	IPv6 address
0x08	Unknown TLV type
0x10	Invalid TLV length, e.g., length of MAC address > 6
0x20	List was truncated... increase MNDP_LSTLEN
0x40	String was truncated... increase MNDP_STRLEN

mndpStat	Description
0x80	Packet was snapped

35.4 Packet File Output

In packet mode (-s option), the mndpDecode plugin outputs the following columns:

Column	Type	Description	Flags
mndpStat	H8	Status	
mndpSeqNo	U16	Sequence Number	
mndpMAC	STR	MAC-Address	
mndpIdentity	STR	Identity	
mndpVersion	STR	Version	
mndpPlatform	STR	Platform	
mndpUptime	U32	Uptime (seconds)	
mndpSoftwareID	STR	Software-ID	
mndpBoard	STR	Board	
mndpUnpack	U8	Packet compression type	
mndpIface	STR	Interface name	
mndpIPv4	IP4	IPv4-Address	
mndpIPv6	IP6	IPv6-Address	

35.5 Plugin Report Output

The following information is reported:

- Aggregated [mndpStat](#)
- Number of MNDP packets

36 modbus

36.1 Description

The modbus plugin analyzes Modbus traffic.

36.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
MB_DEBUG	0	Activate debug output
MB_FE_FRMT	0	Function/Exception codes representation: 0: hex, 1: int
MB_NUM_FUNC	0	Number of function codes to store (0 to hide modbusFC)
MB_UNIQ_FUNC	0	Aggregate multiply defined function codes
MB_NUM_FEX	0	Number of function codes causing exceptions to store (0 to hide modbusFEx)
MB_UNIQ_FEX	0	Aggregate multiply defined function codes causing exceptions
MB_NUM_EX	0	Number of exception codes to store (0 to hide modbusExC)
MB_UNIQ_EX	0	Aggregate multiply defined exception codes

36.3 Flow File Output

The modbus plugin outputs the following columns:

Column	Type	Description	Flags
modbusStat	H16	Status	
modbusUID	U8	Unit identifier	
modbusNPkts	U32	Number of Modbus packets	
modbusNumEx	U16	Number of exceptions	
modbusFCBF	H64	Aggregated function codes	
modbusFC	RH8	List of function codes	MB_NUM_FUNC>0
modbusFExBF	H64	Aggregated function codes which caused exceptions	
modbusFEx	RH8	List of function codes which caused exceptions	MB_NUM_FEX>0
modbusExCBF	H16	Aggregated exception codes	
modbusExC	RH8	List of exception codes	MB_NUM_EX>0

36.3.1 modbusStat

The `modbusStat` column is to be interpreted as follows:

modbusStat	Description
0x0001	Flow is Modbus
0x0002	Non-modbus protocol identifier
0x0004	Unknown function code
0x0008	Unknown exception code
0x0010	Multiple unit identifiers
0x0020	—
0x0040	—
0x0080	—
0x0100	List of function codes truncated... increase <code>MB_NUM_FUNC</code>
0x0200	List of function codes which caused exceptions truncated... increase <code>MB_NUM_FEX</code>
0x0400	List of exception codes truncated... increase <code>MB_NUM_EX</code>
0x0800	—
0x1000	—
0x2000	—
0x4000	Snapped packet
0x8000	Malformed packet

36.3.2 modbusFC and modbusFCBF

The `modbusFC` and `modbusFCBF` columns are to be interpreted as follows:

modbusFC	modbusFCBF	Description
1 = 0x01	0x0000 0000 0000 0002	Read Coils
2 = 0x02	0x0000 0000 0000 0004	Read Discrete Inputs
3 = 0x03	0x0000 0000 0000 0008	Read Multiple Holding Registers
4 = 0x04	0x0000 0000 0000 0010	Read Input Registers
5 = 0x05	0x0000 0000 0000 0020	Write Single Coil
6 = 0x06	0x0000 0000 0000 0040	Write Single Holding Register
7 = 0x07	0x0000 0000 0000 0080	Read Exception Status
8 = 0x08	0x0000 0000 0000 0100	Diagnostic
11 = 0x0b	0x0000 0000 0000 0800	Get Com Event Counter
12 = 0x0c	0x0000 0000 0000 1000	Get Com Event Log
15 = 0x0f	0x0000 0000 0000 8000	Write Multiple Coils
16 = 0x10	0x0000 0000 0001 0000	Write Multiple Holding Registers
17 = 0x11	0x0000 0000 0002 0000	Report Slave ID

modbusFC	modbusFCBF	Description
20 = 0x14	0x0000 0000 0010 0000	Read File Record
21 = 0x15	0x0000 0000 0020 0000	Write File Record
22 = 0x16	0x0000 0000 0040 0000	Mask Write Register
23 = 0x17	0x0000 0000 0080 0000	Read/Write Multiple Registers
24 = 0x18	0x0000 0000 0100 0000	Read FIFO Queue
43 = 0x2b	0x0000 0800 0000 0000	Read Decide Identification

36.3.3 modbusFEx and modbusFExBF

The `modbusFEx` and `modbusFExBF` columns are to be interpreted as `modbusFC` and `modbusFCBF`, respectively.

36.3.4 modbusExC and modbusExCBF

The `modbusExC` and `modbusExCBF` column are to be interpreted as follows:

modbusExC	modbusExCBF	Description
1 = 0x01	0x0002	Illegal function code
2 = 0x02	0x0004	Illegal data address
3 = 0x03	0x0008	Illegal data value
4 = 0x04	0x0010	Slave device failure
5 = 0x05	0x0020	Acknowledge
6 = 0x06	0x0040	Slave device busy
7 = 0x07	0x0080	Negative acknowledge
8 = 0x08	0x0100	Memory parity error
10 = 0xa	0x0400	Gateway path unavailable
11 = 0xb	0x0800	Gateway target device failed to respond

36.4 Packet File Output

In packet mode (-s option), the modbus plugin outputs the following columns:

Column	Type	Description	Flags
<code>mbTranId</code>	U16	Transaction Identifier	
<code>mbProtId</code>	U16	Protocol Identifier	
<code>mbLen</code>	U16	Length	
<code>mbUnitId</code>	U8	Unit identifier	
<code>mbFuncCode</code>	H8	Function code	MB_FE_FRMT=0
<code>mbFuncCode</code>	U8	Function code	MB_FE_FRMT=1

36.4.1 mbFuncCode

If mbFuncCode column is to be interpreted as follows:

mbFuncCode	Description
< 128 (=0x80)	refer to modbusFC and modbusFCBF
≥ 128 (=0x80)	subtract 128 (=0x80) and refer to modbusFEx and modbusFExBF

36.5 Monitoring Output

In monitoring mode, the modbus plugin outputs the following columns:

Column	Type	Description	Flags
modbusNPkts	U64	Number of Modbus packets	
modbusStat	H16	Status	

36.6 Plugin Report Output

The following information is reported:

- Aggregated [modbusStat](#)
- Number of Modbus packets

37 mongoSink

37.1 Description

The mongoSink plugin outputs flows to a MongoDB database.

37.2 Dependencies

37.2.1 External Libraries

This plugin depends on the **libmongoc** library.

Ubuntu:	sudo apt-get install libmongoc-dev
Arch:	sudo pacman -S mongo-c-driver
Gentoo:	sudo emerge mongo-c-driver
Red Hat/Fedora ¹⁸ :	sudo dnf install mongo-c-driver-devel
macOS ¹⁹ :	brew install mongo-c-driver

37.2.2 Core Configuration

This plugin requires the following core configuration:

- \$T2HOME/tralyzer2/src/tralyzer.h:
 - BLOCK_BUF=0

37.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
MONGO_HOST	"127.0.0.1"	Address of the database
MONGO_PORT	27017	Port the database is listening to
MONGO_DBNAME	"tralyzer"	Name of the database
MONGO_TABLE_NAME	"flow"	Name of the database flow table
MONGO_NUM_DOCS	1	Number of documents (flows) to write in bulk
MONGO_QRY_LEN	2048	Max length for query
MONGO_SELECT	0	Only insert specific fields into the DB
MONGO_SELECT_FILE	"mongo-columns.txt"	Filename of the field selector (one column name per line)
BSON_SUPPRESS_EMPTY_ARRAY	1	Output empty fields
BSON_DEBUG	0	Print debug messages

¹⁸If the `dnf` command could not be found, try with `yum` instead

¹⁹Brew is a packet manager for macOS that can be found here: <https://brew.sh>

37.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- `MONGO_HOST`
- `MONGO_PORT`
- `MONGO_DBNAME`
- `MONGO_TABLE_NAME`
- `MONGO_SELECT_FILE` (require `MONGO_SELECT=1`)

37.4 Insertion of Selected Fields Only

When `MONGO_SELECT=1`, the columns to insert into the DB can be customized with the help of `MONGO_SELECT_FILE`. The filename defaults to `mongo-columns.txt` in the user plugin folder, e.g., `~/.tranalyzer/plugins`. The format of the file is simply one field name per line with lines starting with a '#' being ignored. For example, to only insert source and destination addresses and ports, create the following file:

```
# Lines starting with a '#' are ignored and can be used to add comments
srcIP
srcPort
dstIP
dstPort
```

37.5 Working with Timestamps (ISODate)

MongoDB stores timestamps in UTC as `ISODate`. To convert them to localtime, you may use the following query:

```
> db.flow.aggregate([
    $project: {
        localTime: {
            $dateToString: {
                date: "$timeFirst",
                format: "%Y-%m-%d %H:%M:%S",
                timezone: "Europe/Berlin"
            }
        }
    }
])
```

37.6 Example

```
# Run Tranalyzer
$ t2 -r file.pcap

# Connect to the Mongo database
$ mongosh tranalyzer
```

```
# Number of flows
> db.flow.countDocuments()

# 10 first srcIP/dstIP pairs
> db.flow.find({}, { _id: 0, srcIP: 1, dstIP: 1 }).limit(10)

# All flows from 1.2.3.4 to 1.2.3.5
> db.flow.find({ srcIP: "1.2.3.4", dstIP: "1.2.3.5" })
```

For examples of more complex queries, have a look in \$T2HOME/scripts/t2fm/mongo/.

37.6.1 Clean up an existing database

```
# Connect to the Mongo database
$ mongosh tranalyzer

# Drop the database
> db.flow.drop()
```

38 mqttDecode

38.1 Description

The mqttDecode plugin analyzes the MQ Telemetry Transport Protocol (MQTT).

38.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
MQTT_TOPIC_MSG	1	save topics and messages in a separate file	
MQTT_PROTO_LEN	32	Max length for protocol name	
MQTT_CLIENT_ID_LEN	32	Max length for client ID	
MQTT_TOPIC_LEN	32	Max length for topic	
MQTT_TOPIC_MSG_SUFFIX	"_mqtt_msg.txt"	suffix to use for topics and messages file	MQTT_TOPIC_MSG=1

38.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCTRL>0):

- MQTT_TOPIC_MSG_SUFFIX (require MQTT_TOPIC_MSG=1)

38.3 Flow File Output

The mqttDecode plugin outputs the following columns:

Column	Type	Description	Flags
mqttStat	H8	Status	
mqttCPT	H16	Control packet types	
mqttProto	SC	Protocol name	
mqttProtoLevel	U8	Protocol level	
mqttClientID	SC	Client ID	
mqttConAck	H8	Connection status	
mqttTopic	S	Topics	

38.3.1 mqttStat

The mqttStat column is to be interpreted as follows:

mqttStat	Description
0x01	Flow is MQTT
0x02	—
0x04	—
0x08	—

mqttStat	Description
0x10	Reserved Control Packet Type (mqttCPT) (0 or 15) was used
0x20	—
0x40	—
0x80	Packet snapped

38.3.2 mqttCPT

The `mqttCPT` column is to be interpreted as follows:

mqttCPT	Description
2^0 (=0x0001)	Reserved
2^1 (=0x0002)	Client request to connect to server
2^2 (=0x0004)	Connect acknowledgment
2^3 (=0x0008)	Publish message
2^4 (=0x0010)	Publish acknowledgment
2^5 (=0x0020)	Publish complete (assured delivery part 1)
2^6 (=0x0040)	Publish complete (assured delivery part 2)
2^7 (=0x0080)	Publish complete (assured delivery part 3)
2^8 (=0x0100)	Subscribe request
2^9 (=0x0200)	Subscribe acknowledgment
2^{10} (=0x0400)	Unsubscribe request
2^{11} (=0x0800)	Unsubscribe acknowledgment
2^{12} (=0x1000)	PING request
2^{13} (=0x2000)	PING response
2^{14} (=0x4000)	Client is disconnecting
2^{15} (=0x8000)	Reserved

38.3.3 mqttConAck

The `mqttConAck` column is to be interpreted as follows:

mqttConAck	Description
2^0 (=0x01)	Connection Accepted
2^1 (=0x02)	Connection Refused, unacceptable protocol version
2^2 (=0x04)	Connection Refused, identifier rejects
2^3 (=0x08)	Connection Refused, Server unavailable
2^4 (=0x10)	Connection Refused, bad user name or password
2^5 (=0x20)	Connection Refused, not authorized

mqttConAck	Description
2^6 (=0x40)	—
2^7 (=0x80)	Return codes from 0x06 to 0xff are reserved for future use

38.4 Packet File Output

In packet mode (-s option), the mqttDecode plugin outputs the following columns:

Column	Type	Description	Flags
mqttStat	H8	Status	

38.5 Monitoring Output

In monitoring mode, the mqttDecode plugin outputs the following columns:

Column	Type	Description	Flags
mqttPkts	U64	Number of MQTT packets	

38.6 Plugin Report Output

The following information is reported:

- Aggregated [mqttStat](#)
- Number of MQTT packets
- Aggregated [mqttCPT](#)
- Aggregated [mqttConAck](#)

38.7 Additional Output

Non-standard output:

- PREFIX_mqtt_msg.txt: list of topics and messages

38.7.1 _mqtt_msg.txt Output

Column	Type	Description	Flags
pktNo	U64	Packet number	
flowInd	U64	Flow index	
mqttTopic	S	Topic	
mqttMsg	S	Message	

39 mysqlSink

39.1 Description

The mysqlSink plugin outputs flows to a MariaDB / MySQL database.

39.2 Dependencies

39.2.1 External Libraries

This plugin depends on the **MariaDB** or **MySQL** library.

		MariaDB	MySQL
Ubuntu:	sudo apt-get install	libmariadb-dev	libmysqlclient-dev
Arch:	sudo pacman -S	mariadb-libs	
Gentoo:	sudo emerge	mariadb-connector-c	mysql-connector-c
openSUSE:	sudo zypper install	libmariadb-devel	
Red Hat/Fedora ²⁰ :	sudo dnf install	mariadb-connector-c-devel	community-mysql-devel
macOS ²¹ :	brew install	mariadb-connector-c	

39.2.2 Core Configuration

This plugin requires the following core configuration:

- `$T2HOME/tralyzer2/src/tralyzer.h`:
 - `BLOCK_BUF=0`

39.3 Database Setup

39.3.1 Create a User with Create and Write Permissions

```
$ sudo mysql -u root mysql
...
MariaDB [mysql]> create user 'mysql'@'localhost' identified by 'mysql';
MariaDB [mysql]> grant all privileges on *.* to 'mysql'@'localhost' with grant option;
```

39.4 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
<code>MYSQL_OVERWRITE_DB</code>	2	0: abort if DB already exists 1: overwrite DB if it already exists 2: reuse DB if it already exists

²⁰If the `dnf` command could not be found, try with `yum` instead

²¹Brew is a packet manager for macOS that can be found here: <https://brew.sh>

Name	Default	Description
MYSQL_OVERWRITE_TABLE	2	0: abort if table already exists 1: overwrite table if it already exists 2: append to table if it already exists
MYSQL_TRANSACTION_NFLOWS	40000	0: one transaction > 0: one transaction every <i>n</i> flows
MYSQL_QRY_LEN	32768	Max length for query
MYSQL_HOST	"127.0.0.1"	Address of the database
MYSQL_DBPORT	3306	Port the DB is listening to
MYSQL_USER	"mysql"	Username to connect to DB
MYSQL_PASS	"mysql"	Password to connect to DB
MYSQL_DBNAME	"tranalyzer"	Name of the database
MYSQL_TABLE_NAME	"flow"	Name of the table
MYSQL_SELECT	0	Only insert specific fields into the DB
MYSQL_SELECT_FILE	"mysql-columns.txt"	Filename of the field selector (one column name per line)

39.4.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- MYSQL_HOST
- MYSQL_DBPORT
- MYSQL_USER
- MYSQL_PASS
- MYSQL_DBNAME
- MYSQL_TABLE_NAME
- MYSQL_SELECT_FILE (require MYSQL_SELECT=1)

39.5 Insertion of Selected Fields Only

When MYSQL_SELECT=1, the columns to insert into the DB can be customized with the help of MYSQL_SELECT_FILE. The filename defaults to mysql-columns.txt in the user plugin folder, e.g., `~/.tranalyzer/plugins`. The format of the file is simply one field name per line with lines starting with a '#' being ignored. For example, to only insert source and destination addresses and ports, create the following file:

```
# Lines starting with a '#' are ignored and can be used to add comments
srcIP
srcPort
dstIP
dstPort
```

39.6 Example

```
# Run Tranalyzer
$ t2 -r file.pcap
# Connect to the MySQL database
$ mysql tranalyzer
# Number of flows
MariaDB [tranalyzer]> select count(*) from flow;
# 10 first srcIP/dstIP pairs
MariaDB [tranalyzer]> select "srcIP", "dstIP" from flow limit 10;
# All flows from 1.2.3.4 to 1.2.3.5
MariaDB [tranalyzer]> select * from flow where "srcIP" = '1.2.3.4' and "dstIP" = '1.2.3.5';
```

40 nDPI

40.1 Description

This plugin is a simple wrapper around the nDPI library: <https://github.com/ntop/nDPI>. It classifies flows according to their protocol/application by analyzing the payload content instead of using the destination port. This plugin produces output to the flow file and to a protocol statistics file. Configuration is achieved by user defined compiler switches in `src/nDPI.h`.

40.2 Dependencies

40.2.1 External Libraries

This plugin depends on the **libgcrypt** library.

Ubuntu:	sudo apt-get install libgcrypt20-dev
Arch:	sudo pacman -S libgcrypt
Gentoo:	sudo emerge libgcrypt
openSUSE:	sudo zypper install libgcrypt-devel
Red Hat/Fedora ²² :	sudo dnf install libgcrypt-devel
macOS ²³ :	brew install libgcrypt

40.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Variable	Default	Description
NDPI_OUTPUT_NUM	0	Output a numerical classification
NDPI_OUTPUT_STR	1	Output a textual classification
NDPI_OUTPUT_STATS	1	Output nDPI protocol distribution in a separate file
NDPI_GUESS_UNKNOWN	1	Try guessing unknown protocols

40.4 Flow File Output

The nDPI plugin outputs the following columns:

Column	Type	Description	Flags
nDPIIMstrProto	U16	nDPI numerical master protocol	NDPI_OUTPUT_NUM=1
nDPIISubProto	U16	nDPI numerical sub protocol	NDPI_OUTPUT_NUM=1
nDPIclass	S	nDPI based protocol classification	NDPI_OUTPUT_STR=1

²²If the `dnf` command could not be found, try with `yum` instead

²³Brew is a packet manager for macOS that can be found here: <https://brew.sh>

40.5 Packet File Output

In packet mode (-s option), the nDPI plugin outputs the following columns:

Column	Type	Description	Flags
nDPIIMstrProto	U16	nDPI numerical master protocol	NDPI_OUTPUT_NUM=1
nDPISubProto	U16	nDPI numerical sub protocol	NDPI_OUTPUT_NUM=1
nDPIclass	S	nDPI based protocol classification	NDPI_OUTPUT_STR=1

40.6 nDPIIMstrProto

The nDPIIMstrProto column is to be interpreted as follows:

0 Unknown	21 Outlook	42 Mining
1 FTP_CONTROL	22 VK	43 NestLogSink
2 POP3	23 POPS	44 Modbus
3 SMTP	24 Tailscale	45 WhatsAppCall
4 IMAP	25 Yandex	46 DataSaver
5 DNS	26 ntop	47 Xbox
6 IPP	27 COAP	48 QQ
7 HTTP	28 VMware	49 TikTok
8 MDNS	29 SMTPS	50 RTSP
9 NTP	30 DTLS	51 IMAPS
10 NetBIOS	31 UBNTAC2	53 CPHA
11 NFS	32 BFCP	54 iQIYI
12 SSDP	33 YandexMail	55 Zattoo
13 BGP	34 YandexMusic	56 YandexMarket
14 SNMP	35 Gnutella	57 YandexDisk
15 XDMCP	36 eDonkey	58 Discord
16 SMBv1	37 BitTorrent	59 AdobeConnect
17 Syslog	38 Skype_TeamsCall	60 MongoDB
18 DHCP	39 Signal	61 Pluralsight
19 PostgreSQL	40 Memcached	62 YandexCloud
20 MySQL	41 SMBv23	63 OCSP
		64 VXLAN

65 IRC	94 MGCP	123 GoogleMaps
66 MerakiCloud	95 IAX	124 YouTube
67 Jabber	96 TFTP	125 Skype_Teams
68 Nats	97 AFP	126 Google
69 AmongUs	98 YandexMetrika	127 MS-RPCH
70 Yahoo	99 YandexDirect	128 NetFlow
71 DisneyPlus	100 SIP	129 sFlow
72 HART-IP	101 TruPhone	130 HTTP_Connect
73 VRRP	102 ICMPV6	131 HTTP_Proxy
74 Steam	103 DHCPV6	132 Citrix
75 HalfLife2	104 Armagetron	133 NetFlix
76 WorldOfWarcraft	105 Crossfire	134 LastFM
77 Telnet	106 Dofus	135 Waze
78 STUN	107 ADS_Analytic_Track	136 YouTubeUpload
79 IPSec	108 AdultContent	137 Hulu
80 GRE	109 Guildwars	138 CHECKMK
81 ICMP	110 AmazonAlexa	139 AJP
82 IGMP	111 Kerberos	140 Apple
83 EGP	112 LDAP	141 Webex
84 SCTP	113 MapleStory	142 WhatsApp
85 OSPF	114 MsSQL-TDS	143 AppleCloud
86 IP_in_IP	115 PPTP	144 Viber
87 RTP	116 Warcraft3	145 AppleiTunes
88 RDP	117 WorldOfKungFu	146 Radius
89 VNC	118 Slack	147 WindowsUpdate
90 Tumblr	119 Facebook	148 TeamViewer
91 TLS	120 Twitter	149 EthernetGlobalData
92 SSH	121 Dropbox	150 LotusNotes
93 Usenet	122 GMail	151 SAP
		152 GTP
		153 WSD
		154 LLMNR

155	TocaBoca	184	VHUA	213	Starcraft
156	Spotify	185	Telegram	214	Teredo
157	FacebookMessenger	186	CoD_Mobile	215	HotspotShield
158	H323	187	Pandora	216	IMO
159	OpenVPN	188	QUIC	217	GoogleDrive
160	NOE	189	Zoom	218	OCS
161	CiscoVPN	190	EAQ	219	Microsoft365
162	TeamSpeak	191	Ookla	220	Cloudflare
163	Tor	192	AMQP	221	MS_OneDrive
164	CiscoSkinny	193	KakaoTalk	222	MQTT
165	RTCP	194	KakaoTalk_Voice	224	AppleStore
166	RSYNC	195	Twitch	225	OpenDNS
167	Oracle	196	DoH_DoT	226	Git
168	Corba	197	WeChat	227	DRDA
169	UbuntuONE	198	MPEG_TS	228	PlayStore
170	Whois-DAS	199	Snapchat	229	SOMEIP
171	SD-RTN	200	Sina	230	FIX
172	SOCKS	201	GoogleMeet	231	Playstation
173	Nintendo	202	IFLIX	232	Pastebin
174	RTMP	203	Github	233	LinkedIn
175	FTP_DATA	204	BJNP	234	SoundCloud
176	Wikipedia	205	Reddit	235	SteamDatagramRelay
177	ZeroMQ	206	WireGuard	236	LISP
178	Amazon	207	SMPP	237	Diameter
179	eBay	208	DNScrypt	238	ApplePush
180	CNN	209	TINC	239	GoogleServices
181	Megaco	210	Deezer	240	AmazonVideo
182	RESP	211	Instagram	241	GoogleDocs
183	Pinterest	212	Microsoft	242	WhatsAppFiles
				243	TargusDataspeed
				244	DNP3

245	IEC60870	274	Alibaba	303	Psiphon
246	Bloomberg	275	Crashlytics	304	UltraSurf
247	CAPWAP	276	Azure	305	Threema
248	Zabbix	277	iCloudPrivateRelay	306	AliCloud
249	S7Comm	278	EthernetIP	307	AVAST
250	Teams	279	Badoo	308	TiVoConnect
251	WebSocket	280	AccuWeather	309	Kismet
252	AnyDesk	281	GoogleClassroom	310	FastCGI
253	SOAP	282	HSRP	311	FTPS
254	AppleSiri	283	Cybersec	312	NAT-PMP
255	SnapchatCall	284	GoogleCloud	313	Syncthing
256	HP_VIRTGRP	285	Tencent	314	CryNetwork
257	GenshinImpact	286	RakNet	315	Line
258	Activision	287	Xiaomi	316	LineCall
259	FortiClient	288	Edgecast	317	AppleTVPlus
260	Z3950	289	Cachefly	318	DirecTV
261	Likee	290	Softether	319	HBO
262	GitLab	291	MpegDash	320	Vudu
263	AVASTSecureDNS	292	Dazn	321	Showtime
264	Cassandra	293	GoTo	322	Dailymotion
265	AmazonAWS	294	RSH	323	Livestream
266	Salesforce	295	1kxun	324	Tencentvideo
267	Vimeo	296	PGM	325	IHeartRadio
268	FacebookVoip	297	IP_PIM	326	Tidal
269	SignalVoip	298	collectd	327	TuneIn
270	Fuze	299	TunnelBear	328	SiriusXMRadio
271	GTP_U	300	CloudflareWarp	329	Munin
272	GTP_C	301	i3D	330	Elasticsearch
273	GTP_PRIME	302	RiotGames	331	TuyaLP
				332	TPLINK_SHP
				333	Source_Engine
				334	BACnet

335 OICQ	364 UMAS	393 CIP
336 Heroes_of_the_Storm	365 BeckhoffADS	394 Gearman
337 FbookReelStory	366 ISO9506-1-MMS	395 TencentGames
338 SRTP	367 IEEE-C37118	396 GaijinEntertainment
339 OperaVPN	368 Ether-S-Bus	397 ANSI_C1222
340 EpicGames	369 Monero	398 Huawei
341 GeForceNow	370 DCERPC	399 HuaweiCloud
342 Nvidia	371 PROFINET_IO	400 DLEP
343 BITCOIN	372 HiSLIP	401 BFD
344 ProtonVPN	373 UFTP	402 NetEaseGames
345 Thrift	374 OpenFlow	403 PathofExile
346 Roblox	375 JSON-RPC	404 GoogleCall
347 Service_Location_Protocol	376 WebDAV	405 PFCP
348 Mullvad	377 Kafka	406 FLUTE
349 HTTP2	378 NoMachine	407 LoLWildRift
350 HAProxy	379 IEC62056	408 TES_Online
351 RMCP	380 HL7	409 LDP
352 Controller_Area_Network	381 Ceph	410 KNXnet_IP
353 Protobuf	382 GoogleChat	411 Bluesky
354 ETHEREUM	383 Roughtime	412 Mastodon
355 TelegramVoip	384 PrivateInternetAccess	413 Threads
356 SinaWeibo	385 KCP	414 ViberVoip
357 TeslaServices	386 Dota2	415 ZUG
358 PTPv2	387 Mumble	416 JRMI
359 RTPS	388 Yojimbo	417 RipeAtlas
360 OPC-UA	389 ElectronicArts	418 HLS
361 S7CommPlus	390 STOMP	419 ClickHouse
362 FINS	391 Radmin	420 Nano
363 EtherSIO	392 Raft	421 OpenWire

40.7 Plugin Report Output

The following information is reported:

- Number of flows classified

40.8 Additional Output

If `NDPI_OUTPUT_STATS=1` then nDPI protocol distribution statistics are output in `PREFIX_nDPI.txt`.

40.9 Post-Processing

The `protStat` script can be used to sort the `PREFIX_nDPI.txt` file for the most or least occurring protocols (in terms of number of packets or bytes). It can output the top or bottom N protocols or only those with at least a given percentage:

- list all the options: `protStat --help`
- for better readability, use `protStat with tcol`: `protStat ... | tcol`
- sorted list of protocols (by packets): `protStat PREFIX_nDPI.txt`
- sorted list of protocols (by bytes): `protStat PREFIX_nDPI.txt -b`
- top 10 protocols (by packets): `protStat PREFIX_nDPI.txt -n 10`
- bottom 5 protocols (by bytes): `protStat PREFIX_nDPI.txt -n -5 -b`
- protocols with packets percentage greater than 20%: `protStat PREFIX_nDPI.txt -p 20`
- protocols with bytes percentage smaller than 5%: `protStat PREFIX_nDPI.txt -b -p -5`

40.10 How to Update nDPI to New Version

- download latest stable version (or git clone and checkout stable branch)
- delete `src/nDPI` and replace it with this new version
- run the `./new_ndpi_prepatch.sh` script
- build the nDPI plugin: `t2build -r nDPI`
- Replace the `proto.tex` file using the `prototex` utility and regenerate doc:

```
make -C prototex && ./prototex/prototex > doc/proto.tex
```

- Add the new files to SVN and delete removed files before commit.

41 nFrstPkts

41.1 Description

The nFrstPkts plugin supplies the Packet Length (PL) and Inter-Arrival Times (IAT) of the N first packets per flow as a column. The default value for N is 20. It complements the packet mode (`-s` option) with flow based view for the N first packets signal. The plugin supplies several configuration options of how the resulting packet length signal should be represented. Using the `fpsGplt` script files are generated readily post processable by any command line tool (AWK, Perl), Excel or Data mining suit, such as SPSS. As outlined in the configuration below, Signals can be produced with IAT, or relative/absolute time. Also the aggregation of bursts into a single pulse can be configured via `NFRST_MINIAT`. `NFRST_MINPLAVE` controls the meaning of the PL value in pulse aggregation mode. If 0 it corresponds to the BPP measure currently used in research for categorizing media content.

41.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
<code>NFRST_IAT</code>	1	0: Time relative to flow start 1: Inter-arrival time 2: Absolute time	
<code>NFRST_BCORR</code>	0	0: A,B start at 0.0 1: B shift by flow start	<code>NFRST_MINIATS=0</code>
<code>NFRST_PKTCNT</code>	20	Number of packets to record	
<code>NFRST_HDRINFO</code>	0	add L3,L4 header length	
<code>NFRST_MINIATS</code>	0	0: Standard IAT sequence > 0: minimal packet IAT us/ns defining a pulse signal	
<code>NFRST_MINIATU</code>	0	0: Standard IAT sequence > 0: minimal packet IAT us/ns defining a pulse signal	
<code>NFRST_MINPLENFRC</code>	2	Minimal pulse length fraction	
<code>NFRST_PLAVE</code>	1	0: Sum PL (BPP) 1: Average PL	<code>NFRST_MINIATS>0 NFRST_MINIATU>0</code>
<code>NFRST_XMIN</code>	0	Min PL boundary	
<code>NFRST_XMAX</code>	<code>UINT16_MAX</code>	Max PL boundary	

For the rest of this document, `NFRST_MINIAT` is used to represent `(NFRST_MINIATS>0 || NFRST_MINIATU>0)`.

41.3 Flow File Output

The nFrstPkts plugin outputs the following columns:

Column	Type	Description	Flags
<code>nFpCnt</code>	<code>U32</code>	Number of signal samples	
<code>L2L3L4Pl_Iat</code>	<code>R(U16_UT)</code>	L2/L3/L4 or payload length and inter-arrival times for the N first packets	<code>NFRST_HDRINFO=0 && NFRST_MINIAT=0</code>
<code>L2L3L4Pl_Iat_nP</code>	<code>R(U16_UT_UT)</code>	L2/L3/L4 or payload length, inter-arrival times and pulse length for the N first packets	<code>NFRST_HDRINFO=0 && NFRST_MINIAT>0</code>

Column	Type	Description	Flags
HD31_HD41_L2L3L4P1_Iat	R(U8_U8_U16_UT)	L3Hdr, L4Hdr, L2/L3/L4 or payload length and inter-arrival times for the N first packets	NFRST_HDRINFO=1&&NFRST_MINIAT=0
HD31_HD41_L2L3L4P1_Iat_nP	R(U8_U8_U16_UT_UT)	L3Hdr, L4Hdr, L2/L3/L4 or payload length and inter-arrival times for the N first packets	NFRST_HDRINFO=1&&NFRST_MINIAT>0

41.4 Post-Processing

The `fpsGplt` script can be used to transform the packet signal from `nFrstPkts` to gnuplot or `t2plot` format. It produces several signal variants which can also be used for signal processing and AI applications. For more details, refer to the traffic mining tutorial at <https://tranalyzer.com/tutorial/trafficmining>.

```
>fpsGplt -h
Usage:
  fpsGplt [OPTION...] <FILE>
```

Optional arguments:

-f	Flow index to extract, default: all flows
-d	Flow Direction: 0, 1; default both
-t	noTime: counts on x axis; default time on x axis
-i	invert B Flow PL
-s	time sorted
-h, --help	Show this help, then exit

42 netflowSink

42.1 Description

This plugin is an interface of Tranalyzer to a NetFlow collector.

42.2 Dependencies

42.2.1 Other Plugins

This plugin requires the [basicFlow](#), [basicStats](#) and [tcpFlags](#) plugins.
In addition, the [macRecorder](#) plugin is recommended, but optional.

42.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
NF_SERVADD	"127.0.0.1"	Destination address
NF_DPORT	9995	Destination port
NF_SOCKTYPE	0	Socket type: 0: UDP; 1: TCP
NF_VER	9	NetFlow version 9 or 10 (IPFIX)
NF_NUM4FLWS	200	Max # of IPv4 flows in one message
NF_NUM6FLWS	100	Max # of IPv6 flows in one message

42.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCTRL>0`):

- NF_NUM4FLWS
- NF_NUM6FLWS
- NF_SERVADD
- NF_DPORT

42.4 Example

To collect T2 flow data with **nfcapd**, use the following command:

```
nfcapd -T all -B 1000000 -n wurst,127.0.0.1,.
```

43 ntlmsspDecode

43.1 Description

The ntlmsspDecode plugin analyzes the NT LAN Manager (NTLM) Security Support Provider (NTLMSSP) protocol.

43.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
NTLMSSP_CLI_CHALL	0	Output client challenge	
NTLMSSP_DNS	1	Output DNS computer/domain/tree name	
NTLMSSP_NETBIOS	1	Output NetBIOS computer/domain name	
NTLMSSP_VERSION	2	Output format for the version: 0: do not output the version 1: output the version as string 2: major_minor_build_revision	
NTLMSSP_NAME_LEN	64	Max length for string output	
NTLMSSP_SAVE_AUTH_V1	1	Extract NetNTLMv1 hashes	
NTLMSSP_SAVE_AUTH_V2	1	Extract NetNTLMv2 hashes	
NTLMSSP_SAVE_INFO	0	Add flow information in the hashes files	NTLMSSP_SAVE_AUTH_V1=1 NTLMSSP_SAVE_AUTH_V2=1

The suffix used for the NetNTLMv[12] hashes files is controlled by:

- NTLMSSP_AUTH_V1_FILE (default to "_NetNTLMv1.txt") for NetNTLMv1
- NTLMSSP_AUTH_V2_FILE (default to "_NetNTLMv2.txt") for NetNTLMv2

43.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- NTLMSSP_AUTH_V1_FILE
- NTLMSSP_AUTH_V2_FILE

43.3 Flow File Output

The ntlmsspDecode plugin outputs the following columns:

Column	Type	Description	Flags
ntlmsspStat	H8	Status	
ntlmsspTarget	STRC	Target name	
ntlmsspDomain	STRC	Domain name	
ntlmsspUser	STRC	Username	
ntlmsspHost	STRC	Host/workstation	

Column	Type	Description	Flags
ntlmsspNegotiateFlags	H32	Negotiate Flags	
ntlmsspServChallenge	STRC	Server challenge	
ntlmsspNTProofStr	STRC	NT proof string	
ntlmsspCliChallenge	STRC	Client challenge	NTLMSSP_CLI_CHALL=1
ntlmsspSessKey	STRC	Session key	
ntlmsspVersion	STR	Version	NTLMSSP_VERSION=1
ntlmsspVersionMajor_	U8_	Major Version,	NTLMSSP_VERSION=2
Minor_	U8_	Minor Version,	
Build_	U16_	Build Number and	
Rev	U8	NTLM Current Revision)	
ntlmsspNbComputer	STRC	NetBIOS computer name	NTLMSSP_NETBIOS=1
ntlmsspNbDomain	STRC	NetBIOS domain name	NTLMSSP_NETBIOS=1
ntlmsspDnsComputer	STRC	DNS computer name	NTLMSSP_DNS=1
ntlmsspDnsDomain	STRC	DNS domain name	NTLMSSP_DNS=1
ntlmsspDnsTree	STRC	DNS tree name	NTLMSSP_DNS=1
ntlmsspAttrTarget	STRC	Attribute Target Name	
ntlmsspTimestamp	U64.U32/S	Timestamp	

43.3.1 ntlmsspStat

The ntlmsspStat column is to be interpreted as follows:

ntlmsspStat	Description
0x01	Flow is NTLMSSP
0x02	Flow contains Negotiate messages
0x04	Flow contains Challenge messages
0x08	Flow contains Authenticate messages
0x10	NetNTLMv1 hash was extracted for this flow
0x20	NetNTLMv2 hash was extracted for this flow
0x40	String output was truncated... increase NTLMSSP_NAME_LEN
0x80	Decoding error, invalid message type, ...

43.3.2 ntlmsspNegotiateFlags

The ntlmsspNegotiateFlags column is to be interpreted as follows:

ntlmsspNegotiateFlags	Description
2 ⁰ (=0x00000001)	NTLMSSP_NEGOTIATE_UNICODE
2 ¹ (=0x00000002)	NTLMSSP_NEGOTIATE_OEM
2 ² (=0x00000004)	NTLMSSP_REQUEST_TARGET
2 ³ (=0x00000008)	NTLMSSP_NEGOTIATE_00000008 (Reserved, MUST be 0)
2 ⁴ (=0x00000010)	NTLMSSP_NEGOTIATE_SIGN

ntlmsspNegotiateFlags	Description
2^5 (=0x00000020)	NTLMSSP_NEGOTIATE_SEAL
2^6 (=0x00000040)	NTLMSSP_NEGOTIATE_DATAGRAM
2^7 (=0x00000080)	NTLMSSP_NEGOTIATE_LM_KEY
2^8 (=0x00000100)	NTLMSSP_NEGOTIATE_00000100 (Reserved, MUST be 0)
2^9 (=0x00000200)	NTLMSSP_NEGOTIATE_NTLM
2^{10} (=0x00000400)	NTLMSSP_NEGOTIATE_NT_ONLY (Reserved, MUST be 0?)
2^{11} (=0x00000800)	NTLMSSP_NEGOTIATE_ANONYMOUS
2^{12} (=0x00001000)	NTLMSSP_NEGOTIATE_OEM_DOMAIN_SUPPLIED
2^{13} (=0x00002000)	NTLMSSP_NEGOTIATE_OEM_WORKSTATION_SUPPLIED
2^{14} (=0x00004000)	NTLMSSP_NEGOTIATE_00004000 (Reserved, MUST be 0)
2^{15} (=0x00008000)	NTLMSSP_NEGOTIATE_ALWAYS_SIGN
2^{16} (=0x00010000)	NTLMSSP_TARGET_TYPE_DOMAIN
2^{17} (=0x00020000)	NTLMSSP_TARGET_TYPE_SERVER
2^{18} (=0x00040000)	NTLMSSP_TARGET_TYPE_SHARE (Reserved, MUST be 0?)
2^{19} (=0x00080000)	NTLMSSP_NEGOTIATE_EXTENDED_SESSIONSECURITY
2^{20} (=0x00100000)	NTLMSSP_NEGOTIATE_IDENTIFY
2^{21} (=0x00200000)	NTLMSSP_NEGOTIATE_00200000 (Reserved, MUST be 0)
2^{22} (=0x00400000)	NTLMSSP_REQUEST_NON_NT_SESSION_KEY
2^{23} (=0x00800000)	NTLMSSP_NEGOTIATE_TARGET_INFO
2^{24} (=0x01000000)	NTLMSSP_NEGOTIATE_01000000 (Reserved, MUST be 0)
2^{25} (=0x02000000)	NTLMSSP_NEGOTIATE_VERSION
2^{26} (=0x04000000)	NTLMSSP_NEGOTIATE_04000000 (Reserved, MUST be 0)
2^{27} (=0x08000000)	NTLMSSP_NEGOTIATE_08000000 (Reserved, MUST be 0)
2^{28} (=0x10000000)	NTLMSSP_NEGOTIATE_10000000 (Reserved, MUST be 0)
2^{29} (=0x20000000)	NTLMSSP_NEGOTIATE_128
2^{30} (=0x40000000)	NTLMSSP_NEGOTIATE_KEY_EXCH
2^{31} (=0x80000000)	NTLMSSP_NEGOTIATE_56

43.4 Plugin Report Output

The following information is reported:

- Aggregated `ntlmsspStat`
- Number of NTLMSSP packets
- Number of NetNTLMv1 hashes extracted (NTLMSSP_SAVE_AUTH_V1=1)
- Number of NetNTLMv2 hashes extracted (NTLMSSP_SAVE_AUTH_V2=1)

43.5 Additional Output

The following non-standard files are produced:

- PREFIX_NetNTLMv1.txt: NetNTLMv1 hashes (NTLMSSP_SAVE_AUTH_V1=1)
- PREFIX_NetNTLMv2.txt: NetNTLMv2 hashes (NTLMSSP_SAVE_AUTH_V2=1)

43.6 Post-Processing

43.6.1 NTLMSSP Authentications

When `NTLMSSP_SAVE_AUTH_V1=1` or `NTLMSSP_SAVE_AUTH_V1=1`, the plugin produces file(s) with suffix defined by `NTLMSSP_AUTH_V1_FILE` and `NTLMSSP_AUTH_V2_FILE` containing all the NetNTLMv1 and NetNTLMv2 hashes extracted from the traffic. The hashes can then be reversed using JohnTheRipper²⁴ or Hashcat²⁵ as follows:

- NetNTLMv1:
 - `john --wordlist=password.lst -format=netntlm FILE_NetNTLMv1.txt`
 - `hashcat -m 5500 FILE_NetNTLMv1.txt wordlist.txt --show`
- NetNTLMv2:
 - `john --wordlist=password.lst -format=netntlmv2 FILE_NetNTLMv2.txt`
 - `hashcat -m 5600 FILE_NetNTLMv2.txt wordlist.txt --show`

43.7 References

- [MS-NLMP]: NT LAN Manager (NTLM) Authentication Protocol

²⁴<https://github.com/magnumripper/JohnTheRipper>
²⁵<https://hashcat.net>

44 ntpDecode

44.1 Description

The ntpDecode plugin produces a flow based view of NTP operations between computers for anomaly detection and troubleshooting.

44.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
NTP_TS	1	0: no time stamps, 1: print NTP time stamps	
NTP_LIVM_HEX	0	Leap indicator, version and mode: 0: split into three values, 1: aggregated hex number	

44.3 Flow File Output

The ntpDecode plugin outputs the following columns:

Name	Type	Description	Flags
ntpStat	H8	NTP status, warnings and errors	
ntpLiVM	H8	NTP leap indicator, version number and mode	NTP_LIVM_HEX=1
ntpLi_V_M	U8_U8_U8	NTP leap indicator, version number and mode	NTP_LIVM_HEX=0
ntpStrat	H8	NTP stratum	
ntpRefClkId	IP4	NTP root reference clock ID (stratum ≥ 2)	
ntpRefStrId	SC	NTP root reference string (stratum ≤ 1)	
ntpPollInt	U32	NTP poll interval	
ntpPrec	F	NTP precision	
ntpRtDelMin	F	NTP root delay minimum	
ntpRtDelMax	F	NTP root delay maximum	
ntpRtDispMin	F	NTP root dispersion minimum	
ntpRtDispMax	F	NTP root dispersion maximum	
ntpRefTS	TS	NTP reference timestamp	NTP_TS=1
ntpOrigTS	TS	NTP originate timestamp	NTP_TS=1
ntpRecTS	TS	NTP receive timestamp	NTP_TS=1
ntpTranTS	TS	NTP transmit timestamp	NTP_TS=1

44.3.1 ntpStat

The ntpStat column is to be interpreted as follows:

ntpStat	Description
2^0 (=0x01)	NTP port detected
2^1 (=0x02)	—
2^2 (=0x04)	—
2^3 (=0x08)	—
2^4 (=0x10)	—
2^5 (=0x20)	—
2^6 (=0x40)	—
2^7 (=0x80)	—

44.3.2 ntpLiVM

The ntpLiVM column is to be interpreted as follows (refer to Section 44.6 for some examples):

ntpLiVM	Description
xx..	Leap indicator
..xx x...	Version number
.... .xxx	Mode

The Leap Indicator bits are to be interpreted as follows:

Leap Indicator	Description
0x0	No warning
0x1	Last minute has 61 seconds
0x2	Last minute has 59 seconds
0x3	Alarm condition, clock not synchronized

The Mode bits are to be interpreted as follows:

Mode	Description
0x0	Reserved
0x1	Symmetric active
0x2	Symmetric passive
0x3	Client
0x4	Server
0x5	Broadcast
0x6	NTP control message
0x7	Private use

44.3.3 ntpStrat

The `ntpStrat` column is to be interpreted as follows:

<code>ntpStrat</code>	Description
0x00	Unspecified
0x01	Primary reference
0x02–0xff	Secondary reference

44.3.4 ntpRefStrId

The interpretation of the `ntpRefStrId` column depends on the value of `ntpStrat`. The following table lists some suggested identifiers:

<code>ntpStrat</code>	<code>ntpRefStrId</code>	Description
0x00	DCN	DCN routing protocol
0x00	NIST	NIST public modem
0x00	TSP	TSP time protocol
0x00	DTS	Digital Time Service
0x01	ATOM	Atomic clock (calibrated)
0x01	VLF	VLF radio
0x01	callsign	Generic radio
0x01	LORC	LORAN-C
0x01	GOES	GOES UHF environment satellite
0x01	GPS	GPS UHF positioning satellite

44.4 Monitoring Output

In monitoring mode, the `ntpDecode` plugin outputs the following columns:

Column	Type	Description	Flags
<code>ntpPkts</code>	U64	Number of NTP packets	

44.5 Plugin Report Output

The following information is reported:

- Aggregated `ntpStat`
- Number of NTP packets

44.6 Examples

- Extract the NTP leap indicator:

```
tawk 'NR > 1 { print rshift(and(strtonum($ntpLiVM), 0xc0), 6) }' out_flows.txt
```

- Extract the NTP version:

```
tawk 'NR > 1 { print rshift(and(strtonum($ntpLiVM), 0x38), 3) }' out_flows.txt
```

- Extract the NTP mode:

```
tawk 'NR > 1 { printf "%#x\n", and(strtonum($ntpLiVM), 0x7) }' out_flows.txt
```

45 ospfDecode

45.1 Description

This plugin analyzes OSPFv4/6 traffic and provides absolute and relative statistics to the `PREFIX_ospfStats.txt` file. In addition, the `rospf` script extracts the areas, networks and netmasks, along with the routers and their interfaces (Section 45.7).

45.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
OSPF_OUTPUT_HLO	1	Output hello messages	
OSPF_OUTPUT_DBDB	1	Output database description messages (routing tables)	
OSPF_OUTPUT_MSG	1	Output all other messages	
OSPF_OUTPUT_STATS	1	Output statistics file	
OSPF_MASK_AS_IP	1	Netmasks representation: 0: hex, 1: IPv4	
OSPF_AREA_AS_IP	0	Areas representation: 0: int, 1: IPv4, 2: hex	
OSPF_LSID_AS_IP	0	Link State ID representation: 0: int, 1: IPv4	
OSPF_TYP_STR	1	Message type representation: 0: hex, 1: string	
OSPF_LSTYP_STR	1	LS type representation: 0: int, 1: string	
OSPF_NEIGMAX	10	Maximum neighbors to store	
OSPF_NUMTYP	10	Maximum number of LS types to store	OSPF_TYP_STR=1

In addition, the suffix for the output files can be controlled with the following flags:

Name	Default	Description
OSPF_SUFFIX	"_ospfStats.txt"	Statistics
OSPF_HELLO_SUFFIX	"_ospfHello.txt"	OSPFv2/3 hello messages
OSPF_DBDB_SUFFIX	"_ospfDBD.txt"	OSPFv2/3 database description (routing tables)
OSPF2_MSG_SUFFIX	"_ospf2Msg.txt"	All other messages from OSPFv2 (Link State Request/Update/Ack)
OSPF3_MSG_SUFFIX	"_ospf3Msg.txt"	All other messages from OSPFv3 (Link State Request/Update/Ack)

45.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- OSPF_SUFFIX
- OSPF_HELLO_SUFFIX
- OSPF_DBDB_SUFFIX
- OSPF2_MSG_SUFFIX
- OSPF3_MSG_SUFFIX

45.3 Flow File Output

The ospfDecode plugin outputs the following columns:

Column	Type	Description	Flags
<code>ospfStat</code>	H8	Status	
<code>ospfVersion</code>	H8	Version	
<code>ospfType</code>	H8/RS	Message type	<code>OSPF_TYP_STR=0/1</code>
<code>ospfLSType</code>	H64	Update LS type	
<code>ospfAuType</code>	H16	Authentication type	
<code>ospfAuPass</code>	RS	Authentication password (if <code>ospfAuType == 0x4</code>)	
<code>ospfArea</code>	U32/IP4/H32	Area ID	<code>OSPF_AREA_AS_IP=0/1/2</code>
<code>ospfSrcRtr</code>	IP4	Hello source router	
<code>ospfBkupRtr</code>	IP4	Hello backup router	
<code>ospfNeighbors</code>	R(IP4)	Hello neighbor router	

45.3.1 `ospfStat`

The hex based status variable (`ospfStat`) is defined as follows:

<code>ospfStat</code>	Description
2^0 (=0x01)	OSPF detected
2^1 (=0x02)	OSPFv2 message had invalid TTL ($\neq 1$)
2^2 (=0x04)	OSPFv2 message had invalid destination
2^3 (=0x08)	OSPF message had invalid type
2^4 (=0x10)	OSPF unknown version
2^5 (=0x20)	—
2^6 (=0x40)	—
2^7 (=0x80)	OSPF message was malformed (snapped, covert channels?, ...)

The invalid checksum status 0x08 is currently not implemented.

The malformed status 0x10 is currently used to report cases such as possible covert channels, e.g., authfield used when auType was NULL.

45.3.2 `ospfType`

The hex based message type variable `ospfType` is defined as follows:

<code>ospfType</code>	Description
2^0 (=0x01)	Not valid
2^1 (=0x02)	Hello
2^2 (=0x04)	Database Description
2^3 (=0x08)	Link State Request

ospfType	Description
2^4 (=0x10)	Link State Update
2^5 (=0x20)	Link State Acknowledgement
2^6 (=0x40)	—
2^7 (=0x80)	—

45.3.3 ospfLSType

The hex based message type variable `ospfLSType` is defined as follows:

ospfLSType	Description
2^0 (=0x0000 0000 0000 0001)	Reserved
2^1 (=0x0000 0000 0000 0002)	OSPFv2/3 Router-LSA
2^2 (=0x0000 0000 0000 0004)	OSPFv2/3 Network-LSA
2^3 (=0x0000 0000 0000 0008)	OSPFv2 Summary-LSA (IP network) OSPFv3 Inter-Area-Prefix-LSA
2^4 (=0x0000 0000 0000 0010)	OSPFv2 Summary-LSA (ASBR) OSPFv3 Inter-Area-Router-LSA
2^5 (=0x0000 0000 0000 0020)	OSPFv2/3 AS-External-LSA
2^6 (=0x0000 0000 0000 0040)	OSPFv2 Multicast group LSA (not implemented by Cisco) Deprecated in OSPFv3
2^7 (=0x0000 0000 0000 0080)	OSPFv2 Not-so-stubby area (NSSA) External LSA OSPFv3 NSSA-LSA
2^8 (=0x0000 0000 0000 0100)	OSPFv2 External attribute LSA for BGP OSPFv3 Link-LSA
2^9 (=0x0000 0000 0000 0200)	OSPFv2 Opaque LSA: Link-local scope OSPFv3 Intra-Area-Prefix-LSA
2^{10} (=0x0000 0000 0000 0400)	OSPFv2 Opaque LSA: Area-local scope OSPFv3 Intra-Area-TE-LSA
2^{11} (=0x0000 0000 0000 0800)	OSPFv2 Opaque LSA: autonomous system scope OSPFv3 GRACE-LSA
2^{12} (=0x0000 0000 0000 1000)	OSPFv3 Router Information (RI)
2^{13} (=0x0000 0000 0000 2000)	OSPFv3 Inter-AS-TE-v3 LSA
2^{14} (=0x0000 0000 0000 4000)	OSPFv3 L1VPN LS
2^{15} (=0x0000 0000 0000 8000)	OSPFv3 Autoconfiguration (AC) LSA
2^{16} (=0x0000 0000 0001 0000)	OSPFv3 Dynamic Flooding LSA
2^{17} – 2^{32} are unassigned	
2^{33} (=0x0000 0002 0000 0000)	OSPFv3 E-Router-LSA
2^{34} (=0x0000 0004 0000 0000)	OSPFv3 E-Network-LSA
2^{35} (=0x0000 0008 0000 0000)	OSPFv3 E-Inter-Area-Prefix-LSA

ospfLSType	Description
2^{36} (=0x0000 0010 0000 0000)	OSPFv3 E-Inter-Area-Router-LSA
2^{37} (=0x0000 0020 0000 0000)	OSPFv3 E-AS-External-LSA
2^{38} (=0x0000 0040 0000 0000)	Unused (not to be allocated)
2^{39} (=0x0000 0080 0000 0000)	OSPFv3 E-Type-7-LSA
2^{40} (=0x0000 0100 0000 0000)	OSPFv3 E-Link-LSA
2^{41} (=0x0000 0200 0000 0000)	OSPFv3 E-Intra-Area-Prefix-LSA

45.3.4 ospfAuType

The hex based authentication type variable `ospfAuType` is defined as follows:

ospfAuType	Description
2^1 (=0x0002)	Null authentication
2^2 (=0x0004)	Simple password
2^3 (=0x0008)	Cryptographic authentication

45.4 Packet File Output

In packet mode (-s option), the `ospfDecode` plugin outputs the following columns:

Column	Type	Description	Flags
<code>ospfStat</code>	H8	Status	
<code>ospfVersion</code>	U8	Version	
<code>ospfArea</code>	U32/IP4/H32	Area ID	<code>OSPF_AREA_AS_IP=0/1/2</code>
<code>ospfType</code>	S	Message Type	
<code>ospfLSType</code>	H64	Update LS Type	

45.5 Plugin Report Output

The following information is reported:

- Aggregated `ospfStat`
- Aggregated `ospfType` for OSPFv2 and OSPFv3
- Number of OSPFv2 packets
- Number of OSPFv3 packets

45.6 Additional Output

- `PREFIX_ospfStats.txt`: global statistics about OSPF traffic
- `PREFIX_ospfHello.txt` Hello messages (see Section 45.7)

- PREFIX_ospfDBD.txt: Routing tables (see OSPF_OUTPUT_DB in Section 45.2)
- PREFIX_ospf2Msg.txt: All other messages from OSPFv2 (see OSPF_OUTPUT_MSG in Section 45.2)
- PREFIX_ospf3Msg.txt: All other messages from OSPFv3 (see OSPF_OUTPUT_MSG in Section 45.2)

45.7 Post-Processing

45.7.1 rospf

Hello messages can be used to discover the network topology and are stored in the PREFIX_ospfHello.txt file. The script rospf extracts the areas, networks, netmasks, routers and their interfaces:

```
./scripts/rospf PREFIX_ospfHello.txt
```

Name	Area	Network	Netmask
N1	0	192.168.21.0	0xffffffff00
N2	1	192.168.16.0	0xffffffff00
N3	1	192.168.22.0	0xfffffffffc
...			
Router	Interface_n	Network_n	
R1	192.168.22.29	N11	192.168.21.4
R2	192.168.22.5	N12	192.168.16.1
R3	192.168.22.10	N13	192.168.21.2
...			
Router	Connected Routers		
R0	R2 R4 R6 R7 R8		
R1	R2 R4		
R2	R0 R1 R4 R8		
...			

45.7.2 dbd

If OSPF_OUTPUT_DB is activated (Section 45.2), database description messages are stored in a file PREFIX_ospfDBD.txt. The dbd script formats this file to produce an output similar to that of standard routers:

```
./scripts/dbd PREFIX_ospfDBD.txt
```

OSPF Router with ID (192.168.22.10)				
Router Link States (Area 1)				
Link ID	ADV Router	Age	Seq#	Checksum
192.168.22.5	192.168.22.5	4	0x80000002	0x38ce
192.168.22.10	192.168.22.10	837	0x80000002	0x6b0f
192.168.22.9	192.168.22.9	837	0x80000002	0x156c
Net Link States (Area 1)				
Link ID	ADV Router	Age	Seq#	Checksum
192.168.22.6	192.168.22.10	4	0x80000001	0x150b
192.168.22.9	192.168.22.9	838	0x80000001	0x39e0

```
Summary Net Link States (Area 1)
```

Link ID	ADV Router	Age	Seq#	Checksum
192.168.17.0	192.168.22.9	735	0x80000001	0x5dd9
192.168.17.0	192.168.22.10	736	0x80000001	0x57de
192.168.18.0	192.168.22.9	715	0x80000001	0x52e3

```
...
```

46 p0f

46.1 Description

The p0f plugin tries to fingerprint OS and applications.

46.2 Dependencies

46.2.1 Other Plugins

This plugin requires the **sslDecode** plugin with the following flags activated, i.e., set to 1:

- SSL_EXT_LIST
- SSL_CIPHER_LIST

46.2.2 Required Files

The file p0f-ssl.txt is required.

46.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
P0F_SSL_VER	1	Consider the version for fingerprint match	
P0F_SSL_NCIPHER	1	Consider the number of ciphers for fingerprint match	
P0F_SSL_NUMEXT	1	Consider the number of extensions for fingerprint match	
P0F_SSL_FLAGS	1	Consider flags for fingerprint match	
P0F_SSL_CIPHER	1	Consider ciphers for fingerprint match	
P0F_SSL_EXT	1	Consider extensions for fingerprint match	
P0F_SSL_ELEN	6	Maximum length of cipher or extension	
P0F_SSL_NSIG	64	Maximum number of signatures to read	
P0F_SSL_SLEN	128	Maximum length of a string (os, browser, comment)	
P0F_SSL_DB	"p0f-ssl.txt"	Name of the database to use	

46.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCTRL>0`):

- P0F_SSL_DB

46.4 Flow File Output

The p0f plugin outputs the following columns:

Column	Type	Description	Flags
p0fSSLRule	U16	p0f SSL fingerprint rule number	
p0fSSLOS	S	p0f SSL OS fingerprint	
p0fSSLOS2	S	p0f SSL OS fingerprint (2)	
p0fSSLBrowser	S	p0f SSL browser fingerprint	
p0fSSLComment	S	p0f SSL fingerprint comment	

46.5 References

- <https://idea.popcount.org/2012-06-17-ssl-fingerprinting-for-p0f>

47 payloadDumper

47.1 Description

The payloadDumper plugin dumps the payload of layer 2, TCP, UDP or SCTP flows to files. It provides features similar to [tcpflow](#).

47.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
PLDUMP_L2	0	Extract payload for layer 2 flows	ETH_ACTIVATE>0
PLDUMP_ETHERTYPES	{}	Only extract L2 payloads for those ethertypes e.g., {0x2000, 0x2003}	PLDUMP_L2=1
PLDUMP_TCP	1	Extract payload for TCP flows	
PLDUMP_TCP_PORTS	{}	Only extract TCP payloads on those ports, e.g., {80, 8080}	PLDUMP_TCP=1
PLDUMP_UDP	1	Extract payload for UDP flows	
PLDUMP_UDP_PORTS	{}	Only extract UDP payloads on those ports, e.g., {80, 8080}	PLDUMP_UDP=1
PLDUMP_SCTP	0	Extract payload for TCP flows	SCTP_ACTIVATE=1
PLDUMP_SCTP_PORTS	{}	Only extract SCTP payloads on those ports, e.g., {80, 8080}	PLDUMP_SCTP=1 & SCTP_ACTIVATE=1
PLDUMP_MAX_BYTES	0	Max. number of bytes per flow to dump (use 0 for no limits)	
PLDUMP_START_OFFSET	0	Start dumping bytes at a specific offset (Layer 2 and UDP only)	PLDUMP_L2=1 PLDUMP_UDP=1
PLDUMP_RMDIR	1	Empty PLDUMP_FOLDER before starting	
PLDUMP_NAMES	0	Format for filenames: 0: flowInd_[AB] 1: srcIP.srcPort-dstIP.dstPort-l4Proto Extra suffix for SCTP: _sctpStream For L2: srcMac-dstMac-etherType 2: Same as 1, but prefixed with timestampT	
PLDUMP_FOLDER	"/tmp/payloadDumper/"	Output folder for saved files	
PLDUMP_PREFIX	" "	Prefix for output files	
PLDUMP_SUFFIX	" "	Suffix for output files	

47.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- `PLDUMP_RMDIR`
- `PLDUMP_FOLDER`
- `PLDUMP_PREFIX`
- `PLDUMP_SUFFIX`

47.3 Flow File Output

The payloadDumper plugin outputs the following columns:

Column	Type	Description	Flags
<code>pldStat</code>	H8	Status	

47.3.1 pldStat

The `pldStat` column is to be interpreted as follows:

pldStat	Description
2^0 (=0x01)	Match for this flow
2^1 (=0x02)	Dump payload for this flow
2^2 (=0x04)	SCTP init TSN diff engine
2^3 (=0x08)	SCTP payload truncated
2^4 (=0x10)	TCP sequence numbers out of order or roll-over or TCP keep-alive
2^5 (=0x20)	SCTP TSN out of order or roll-over
2^6 (=0x40)	Filename truncated
2^7 (=0x80)	Failed to open file

47.4 Packet File Output

In packet mode (-s option), the payloadDumper plugin outputs the following columns:

Column	Type	Description	Flags
<code>pldStat</code>	H8	Status	

47.5 Plugin Report Output

The following information is reported:

- Aggregated `pldStat`
- Number of non zero content dumped flows

47.6 Additional Output

The payload of the layer 2, TCP, UDP and/or SCTP flows is extracted in `PLDUMP_FOLDER`. Each file is named according to the value of `PLDUMP_NAMES`, `PLDUMP_SUFFIX` and `PLDUMP_SUFFIX`.

48 pcapd

48.1 Description

The pcapd plugin can be used to create PCAP files based on some criteria such as flow indexes (Section 48.3.2) or alarms raised by other plugins (Section 48.3.3).

48.2 Dependencies

If PD_MODE_OUT=1 (one pcap per flow), the libpcap version must be at least 1.7.2. (In this mode, the plugin uses the pcap_dump_open_append() function which was introduced in the libpcap in February 12, 2015.)

48.3 Configuration Flags

The following flags can be used to configure the plugin:

Variable	Default	Description	Flags
PD_MODE_PKT	0	0: all packets, 1: activate packet range selection	
PD_STRTPKT	1	Packet at which processing starts	PD_MODE_PKT=1
PD_ENDPKT	10	Packet at which processing ends (0: end of flow)	PD_MODE_PKT=1
PD_MODE_IN	0	0: if -e option was used, extract flows listed in FILE, otherwise, extract flows whose alarm bit is set 1: dump all packets	
PD_EQ	1	Save matching (1) or non-matching (0) flows	PD_MODE_IN=0
PD_OPP	0	1: extract also the opposite flow, 0: don't	PD_MODE_IN=0
PD_DIRSEL	0	2: extract A flow, 3: extract B flow, 0: off	PD_MODE_IN=0
PD_MODE_OUT	0	0: one pcap 1: one pcap per flow	
PD_SPLIT	1	Split the output file (Tranalyzer -W option)	
PD_LBSRCH	0	Search algorithm (-e option): 0: linear search 1: binary search	
PD_TSHFT	0	Time stamp shift in packets	
PD_TTSFTS	0	Time stamp increment seconds	PD_TSHFT=1
PD_TTSFTNMS	1	Time stamp increment micro/nano seconds (depends on TSTAMP_PREC in <i>tranalyzer.h</i>)	PD_TSHFT=1
PD_MACSHFT	0	MAC shift in packets	
PD_MACSSHFT	1	Src MAC increment in packets last byte	PD_MACSHFT=1
PD_MACDSHFT	1	Src MAC increment in packets last byte	PD_MACSHFT=1
PD_VLNSHFT	0	VLAN shift in packets	
PD_VLNISHFT	1	VLAN increment in packets	PD_VLNSHFT=1
PD_IPSHFT	0	IPv4/6 increment in packets	
PD_IP4SHFT	0x00000001	IPv4 shift 32 bit network order	PD_IPSHFT=1
PD_IP6SHFT	0x0000000000000001	IPv6 shift last 64 bit network order	PD_IPSHFT=1

Variable	Default	Description	Flags
PD_TTLSHFT	0	0: no TTL change, 1: TTL shift, 2: random shift	
PD_TTL	8	sub value from TTL	PD_TTLSHFT=1
PD_TTLMOD	128	TTL modulo	PD_TTLSHFT>0
PD_CHKSUML3	0	Correct checksum in IPv4 header	
PD_MAX_FD	128	Max. number of simultaneously open file descriptors	PD_MODE_OUT=1
PD_SUFFIX	"_pcapd.pcap"	Suffix for output pcap file	

48.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- `PD_MAX_FD`
- `PD_SUFFIX`

48.3.2 PD_MODE_IN=0, -e option used

The idea behind this mode (`PD_MODE_IN=0` and Tranalyzer `-e` option used) is to use awk to extract flows of interest and then the pcapd plugin to create one or more PCAP with all those flows. The input file (`-e` option) consists of one column listing the flow indexes (one entry per row). Lines starting with '`%`', '`#`', a space or a tab are ignored, along with empty lines:

```
# This file is passed to tranalyzer via -e option: t2 -e file.txt ...
# Each row lists a different flow index to extract.
123    # Extract flow 123
456    # Extract flow 456
#789   # Do NOT extract flow 789
```

Flows whose index appears in the `-e` file will be dumped in a file named `PREFIX_PD_SUFFIX`, where `PREFIX` is the value given to Tranalyzer `-e` option. Note that if `PD_EQ=0`, then flows whose index does **not** appear in the file will be dumped.

48.3.3 PD_MODE_IN=0, -e option not used

In this mode (`PD_MODE_IN=0` and Tranalyzer `-e` option **NOT** used), every flow whose status bit `FL_ALARM=0x20000000` is set (`PD_EQ=1`) or not set (`PD_EQ=0`) will be dumped in a file named `PREFIX_PD_SUFFIX`, where `PREFIX` is the value given to Tranalyzer `-w` or `-W` option.

48.3.4 PD_MODE_IN=1

In this mode, all the packets are dumped into one or more PCAP files. If Tranalyzer `-W` option is used, then the pcap files will be split accordingly. For example, the following command will create PCAP files of 100MB each: `tranalyzer -i eth0 -W out:100M`

48.3.5 PD_MODE_OUT=1

In this mode, every flow will have its own PCAP file, whose name will end with the flow index.

48.4 Plugin Report Output

The following information is reported:

- Number of packets extracted

48.5 Additional Output

A PCAP file with suffix PD_SUFFIX will be created. The prefix and location of the file depends on the configuration of the plugin.

- If Tranalyzer -e option was used, the file is named according to the -e option.
- Otherwise the file is named according to the -w or -W option.

48.6 Examples

For the following examples, it is assumed that Tranalyzer was run as follows, with the `basicFlow` and `txtSink` plugins in their default configuration:

```
tranalyzer -r file.pcap -w out
```

The column numbers can be obtained by looking in the file `out_headers.txt` or by using `tawk`.

48.6.1 Extracting ICMP Flows

To create a PCAP file containing ICMP flows only, proceed as follows:

1. Identify the “*Layer 4 protocol*” column in `out_headers.txt` (column 14):
`grep "Layer 4 protocol" out_headers.txt`
2. Extract all flow indexes whose protocol is ICMP (1):
`awk -F'\t' '$14 == 1 { print $2 }' out_flows.txt > out_icmp.txt`
3. Configure `pcapd.h` as follows: `PD_MODE_IN=0, PD_EQ=1`
4. Build the `pcapd` plugin: `cd $T2HOME/pcapd/; ./autogen.sh`
5. Re-run Tranalyzer with the -e option:
`tranalyzer -r file.pcap -w out -e out_icmp.txt`
6. The file `out_icmp.txt.pcap` now contains all the ICMP flows.

48.6.2 Extracting Non-ICMP Flows

To create a PCAP file containing non-ICMP flows only, use the same procedure as that of Section 48.6.1, but replace `PD_EQ=1` with `PD_EQ=0` in step 3. Alternatively, replace `$14==1` with `$14!=1` in step 2. Or if an entire flow file is preferred to the flow indexes only, set `PD_FORMAT=1` and replace `print $2` with `print $0` in step 2.

49 pktSIATHisto

49.1 Description

The pktSIATHisto plugin records the packet size (PS) and inter-arrival time (IAT) of a flow. While the PS reflects the bin, the IAT is divided by default into statistical bins to conserve memory / flow (see example below). Where the low precision is reserved for the most prominent IAT of all known codecs. Nevertheless, it can be configured by the user in any arbitrary way. If the memory is not sufficient then decrease `HASHCHAINTABLE_BASE_SIZE` in `tranalyzer.h`.

Bin	Range of IAT (default)
0 – 199	0 ms (incl.) – 200 ms (excl.), partitioned into bins of 1 ms
200 – 239	200 ms (incl.) – 400 ms (excl.), partitioned into bins of 5 ms
240 – 299	400 ms (incl.) – 1 sec. (excl.), partitioned into bins of 10 ms
300	for all IAT higher than 1 sec.

49.2 Configuration Flags

Classifying tasks may require other IAT binning. Then the bin limit `IATBINBu` and the binsize `IATBINWu` constants in `pktSIATHisto.h` need to be adapted as being indicated below using 6 different classes of bins:

```
#define IATSECMAX 6 // max # of section in statistics;
                    // last section comprises all elements > IATBINBu6

#define IATBINBu1    50// bin boundary of section one: [0, 50)ms
#define IATBINBu2    200
#define IATBINBu3    1000
#define IATBINBu4    10000
#define IATBINBu5    100000
#define IATBINBu6    1000000

#define IATBINWu1    10// bin width 1ms
#define IATBINWu2    5
#define IATBINWu3    10
#define IATBINWu4    20
#define IATBINWu5    50
#define IATBINWu6    100

#define IATBINNu1    IATBINBu1 / IATBINWu1// # of bins in section one
#define IATBINNu2    (IATBINBu2 - IATBINBu1) / IATBINWu2 + IATBINNu1
#define IATBINNu3    (IATBINBu3 - IATBINBu2) / IATBINWu3 + IATBINNu2
#define IATBINNu4    (IATBINBu4 - IATBINBu3) / IATBINWu4 + IATBINNu3
#define IATBINNu5    (IATBINBu5 - IATBINBu4) / IATBINWu5 + IATBINNu4
#define IATBINNu6    (IATBINBu6 - IATBINBu5) / IATBINWu6 + IATBINNu5
```

The number of bin sections is defined by `IATSECMAX`, default is 3. The static fields `IATBINBu` and `IATBINWu` need to be adapted when `IATSECMAX` is changed. The static definition in curly brackets of the constant fields `IATBINBu[]`, `IATBINBu[]` and `IATBINBu[]` must adapted as well to the maximal bin size. The constant `IATBINUMAX` including his two dimensional packet length, IAT statistics is being used by the descriptive statistics plugin and can suit as a raw input for subsequent statistical classifiers, such as Bayesian networks or C5.0 trees.

The user is able to customize the output by changing several define statements in the header file `pktSIATHisto.h`. Every change requires a recompilation of the plugin using the `autogen.sh` script.

`HISTO_PRINT_BIN == 0`, the default case, selects the number of the IAT bin, while 1 supplies the lower bound of the IAT bin's range.

As being outlined in the Descriptive Statistics plugin the output of the plugin can be suppressed by defining `PRINT_HISTO` to zero.

For specific applications in the AI regime, the distribution can be directed into a separate file if the value `PRINT_HISTO_IN_SEPARATE_FILE` is different from zero. The suffix for the distribution file is defined by the `HISTO_FILE_SUFFIX` define. All switches are listed below:

Name	Default	Description	Flags
<code>PSIAT_NDPLF</code>	17	multiplication factor red-black tree nodepool	
<code>PRINT_HISTO</code>	1	print histo to flow file	
<code>HISTO_PRINT_BIN</code>	0	Bin number; 0: Minimum of assigned inter arrival time Example: Bin = 10 \Rightarrow iat = [50:55] \Rightarrow min(iat) = 50ms	
<code>HISTO_EARLY_CLEANUP</code>	0	after <code>t2OnFlowTerminate()</code> tree information is destroyed (MUST be 0 if dependent plugins are loaded)	
<code>PSI_XCLD</code>	0	1: include <code>(PSI_XMIN, UINT16_MAX]</code>	
<code>PSI_XMIN</code>	1	minimal packet length starts at <code>PSI_XMIN</code>	<code>PSI_XCLD=1</code>
<code>PSI_MOD</code>	0	> 1 : Modulo factor of packet length	
<code>IATSECMAX</code>	3	max # of sections in statistics, last section comprises all elements $>$ <code>IATBINBuN</code>	<code>PSI_XCLD=1</code>

49.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- `PSIAT_NDPLF`

49.3 Flow File Output

The `pktSIATHisto` plugin outputs the following columns:

Column	Type	Description	Flags
<code>tCnt</code>	U32	Number of tree entries for PS and IAT	
<code>Ps_IatBin_Cnt_</code> <code>PsCnt_IatCnt</code>	R(U16_4xU32)	Packet size and inter-arrival time of bin histogram	<code>HISTO_PRINT_BIN=0</code>
<code>Ps_Iat_Cnt_</code> <code>PsCnt_IatCnt</code>	R(U16_4xU32)	Packet size and min inter-arrival time of bin histogram	<code>HISTO_PRINT_BIN=1</code>

All PS-IAT bins greater than zero are appended for each flow in the `PREFIX_flows.txt` file using the following format:

```
[PS]_[IAT]_[# packets]_[# of packets PS]_[# of packets IAT]
```

the PS-IAT bins are separated by semicolons. The IAT value is the lower bound of the IAT range of a bin.

49.4 Post-Processing

The `statGplt` script can be used to transform the packet length and IAT statistics from `pktSIATHisto` to `gnuplot` or `t2plot` format. The format is:

- For the 3D case: **PS <tab> IAT <tab> count**
- For the 2D case: **PS <tab> count**

49.5 Example Output

Consider a single flow with the following PS and IAT values:

Packet number	PS (bytes)	IAT (ms)	IAT bin
1	50	0	0
2	70	88.2	17
3	70	84.3	16
4	70	92.9	18
5	70	87.1	17
6	60	91.6	18

Packet number two and five have the same PS-IAT combination. Packets number two to five have the same PS and number two and five as well as the number four and six fall within the same IAT bin. Therefore the following sequence is generated:

```
50_0_1_1_1 ; 60_90_1_1_2 ; 70_80_1_4_1 ; 70_85_2_4_2 ; 70_90_1_4_2
```

Note that for better readability spaces are inserted around the semicolons which will not exist in the text based flow file!

50 popDecode

50.1 Description

The popDecode plugin processes MAIL header and content information of a flow. The idea is to identify certain POP mail features and save content.

50.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
POP_SAVE	0	Save content to <code>POP_F_PATH</code>	
POP_BTFLD	1	Enable bitfields output	
POP_MXNMLN	65	Maximal name length	
POP_MXUNM	5	Maximal number of users	
POP_MXPNM	5	Maximal number of passwords/parameters	
POP_MXCNM	10	Maximal number of content	
POP_RMDIR	1	Empty <code>POP_F_PATH</code> before starting	POP_SAVE=1
POP_F_PATH	<code>"/tmp/POPFILES/"</code>	Path for extracted content	POP_SAVE=1
POP_NONAME	<code>"nudel"</code>	No name file name	POP_SAVE=1

50.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- `POP_RMDIR`
- `POP_F_PATH`
- `POP_NONAME`

50.3 Flow File Output

The popDecode plugin outputs the following columns:

Column	Type	Description	Flags
<code>popStat</code>	H16	Status	
<code>popCBF</code>	H16	POP command codes bitfield	<code>POP_BTFLD=1</code>
<code>popCC</code>	RSC	POP command codes	
<code>popRM</code>	RU16	POP response codes	
<code>popUsrNum</code>	U8	POP number of users	
<code>popUsr</code>	RS	POP users	
<code>popPwNum</code>	U8	POP number of passwords	
<code>popPw</code>	RS	POP passwords	
<code>popCNum</code>	U8	POP number of parameters	
<code>popC</code>	RS	POP content	

50.3.1 popStat

The `popStat` column describes the errors encountered during the flow lifetime:

popStat	Description	Flags
2^0 (=0x0001)	POP2 port found	
2^1 (=0x0002)	POP3 port found	
2^2 (=0x0004)	Response +OK	
2^3 (=0x0008)	Response -ERR	
2^4 (=0x0010)	Data storage exists	POP_SAVE=1
2^5 (=0x0020)	Data storage in progress	POP_SAVE=1
2^6 (=0x0040)	Response not valid or data	
2^7 (=0x0080)	Array overflow	
2^8 (=0x0100)	Authentication pending	
2^9 (=0x0200)	Return path pending	
2^{10} (=0x0400)	—	
2^{11} (=0x0800)	—	
2^{12} (=0x1000)	—	
2^{13} (=0x2000)	—	
2^{14} (=0x4000)	—	
2^{15} (=0x8000)	—	

50.3.2 popCBF

The `popCBF` column describes the commands encountered during the flow lifetime:

popCBF	Description
2^0 (=0x0001)	Login with MD5 signature
2^1 (=0x0002)	Authentication request
2^2 (=0x0004)	Get a list of capabilities supported by the server
2^3 (=0x0008)	Mark the message as deleted
2^4 (=0x0010)	Get a scan listing of one or all messages
2^5 (=0x0020)	Return a +OK reply
2^6 (=0x0040)	Cleartext password entry
2^7 (=0x0080)	Exit session. Remove all deleted messages from the server
2^8 (=0x0100)	Retrieve the message
2^9 (=0x0200)	Remove the deletion marking from all messages
2^{10} (=0x0400)	Get the drop listing
2^{11} (=0x0800)	Begin a TLS negotiation
2^{12} (=0x1000)	Get the top n lines of the message

popCBF	Description
2^{13} (=0x2000)	Get a unique-id listing for one or all messages
2^{14} (=0x4000)	Mailbox login
2^{15} (=0x8000)	

50.4 Packet File Output

In packet mode (-s option), the popDecode plugin outputs the following columns:

Column	Type	Description	Flags
popStat	H16	Status	

50.5 Plugin Report Output

The following information is reported:

- POP status
- Number of POP packets
- Number of files extracted (POP_SAVE=1)

50.6 TODO

- fragmentation

51 portClassifier

51.1 Description

The portClassifier plugin classifies the flow according to the destination port meaning. It accepts a default port list `portmap.txt`, automatically installed with the plugin.

51.2 Dependencies

51.2.1 Required Files

The file `portmap.txt` is required.

51.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
PBC_NUM	1	Print string representation of port classification
PBC_STR	1	Print numeric representation of port classification
PBC_CLASSFILE	"portmap.txt"	input file for the mapping between ports and applications
PBC_UNKNOWN	"unknown"	Label for unknown ports

51.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- PBC_CLASSFILE
- PBC_UNKNOWN

51.4 Flow File Output

The portClassifier plugin outputs the following columns:

Column	Type	Description	Flags
dstPortClassN	U16	Port based classification of the destination port number	PBC_NUM=0
dstPortClass	SC	Port based classification of the destination port name	PBC_STR=1

51.5 Packet File Output

In packet mode (-s option), the portClassifier plugin outputs the following columns:

Column	Type	Description	Flags
dstPortClassN	U16	Port based classification of the destination port number	PBC_NUM=0
dstPortClass	SC	Port based classification of the destination port name	PBC_STR=1

52 protoStats

52.1 Description

The protoStats plugin provides protocol/port sorted frequency statistics about the observed OSI layer 4 protocols and ports to the file named PREFIX_protocols. Protocols numbers are decoded via a proto.txt file, automatically installed with the plugin.

52.2 Dependencies

52.2.1 Required Files

The following files are required:

- PST_L2ETHFILE ("ethertypes.txt")
- PST_PORTFILE ("portmap.txt")
- PST_PROTOFILE ("proto.txt")

52.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
PST_ETH_STAT	1	Output layer 2 statistics
PST_SCTP_STAT	0	Output SCTP statistics
PST_UDPLITE_STAT	0	Output UDP-Lite statistics
PST_SUFFIX	"_protocols.txt"	Suffix for output file

52.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- PST_SUFFIX
- PST_L2ETHFILE
- PST_PORTFILE
- PST_PROTOFILE

52.4 Flow File Output

None.

52.5 Additional Output

- PREFIX_protocols.txt: protocol statistics

52.6 Post-Processing

The `protStat` script can be used to sort the `PREFIX_protocols.txt` file for the most or least occurring protocols (in terms of number of packets or bytes). It can output the top or bottom N protocols or only those with at least a given percentage.

- list all the options: `protStat --help`
- for better readability, use `protStat` with `tcol`: `protStat ... | tcol`
- sorted list of protocols (by packets): `protStat PREFIX_protocols.txt`
- sorted list of protocols (by bytes): `protStat PREFIX_protocols.txt -b`
- top 10 protocols (by packets): `protStat PREFIX_protocols.txt -n 10`
- bottom 5 protocols (by bytes): `protStat PREFIX_protocols.txt -n -5 -b`
- protocols with packets percentage greater than 20%: `protStat PREFIX_protocols.txt -p 20`
- protocols with bytes percentage smaller than 5%: `protStat PREFIX_protocols.txt -b -p -5`
- TCP and UDP statistics only: `protStat PREFIX_protocols.txt -udp -tcp`

53 psqlSink

53.1 Description

The psqlSink plugin outputs flows to a PostgreSQL database.

53.2 Dependencies

53.2.1 External Libraries

This plugin depends on the **libpq** library.

Ubuntu:	sudo apt-get install libpq-dev
Arch:	sudo pacman -S postgresql-libs
Gentoo:	sudo emerge postgresql
openSUSE:	sudo zypper install postgresql-devel
Red Hat/Fedora ²⁶ :	sudo dnf install libpq-devel
macOS ²⁷ :	brew install postgresql

53.2.2 Core Configuration

This plugin requires the following core configuration:

- \$T2HOME/tranalyzer2/src/tranalyzer.h:
 - BLOCK_BUF=0

53.3 Database Initialization

The psqlSink plugin requires a PostgreSQL server running on PSQL_HOST, e.g., 127.0.0.1 on port PSQL_PORT, e.g., 5432. In addition, a user with name PSQL_USER, e.g., postgres, and password PSQL_PASS, e.g., postgres **MUST** exist and be allowed to create databases. This can be achieved with the following commands:

Linux: \$ sudo -u postgres psql

macOS: \$ psql postgres

```
postgres=# CREATE ROLE postgres WITH LOGIN PASSWORD 'postgres';
CREATE ROLE
postgres=# ALTER ROLE postgres CREATEDB;
ALTER ROLE
```

53.4 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
PSQL_OVERWRITE_DB	2	0: abort if DB already exists

²⁶If the dnf command could not be found, try with yum instead

²⁷Brew is a packet manager for macOS that can be found here: <https://brew.sh>

Name	Default	Description
PSQL_OVERWRITE_TABLE	2	1: overwrite DB if it already exists 2: reuse DB if it already exists 0: abort if table already exists 1: overwrite table if it already exists 2: append to table if it already exists
PSQL_TRANSACTION_NFLOWS	40000	0: one transaction > 0: one transaction every <i>n</i> flows
PSQL_QRY_LEN	32768	Max length for query
PSQL_HOST	"127.0.0.1"	Address of the database
PSQL_PORT	5432	Port of the database
PSQL_USER	"postgres"	Username to connect to DB
PSQL_PASS	"postgres"	Password to connect to DB
PSQL_DBNAME	"tranalyzer"	Name of the database
PSQL_TABLE_NAME	"flow"	Name of the table
PSQL_SELECT	0	Only insert specific fields into the DB
PSQL_SELECT_FILE	"pgsql-columns.txt"	Filename of the field selector (one column name per line)

53.4.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- PSQL_HOST
- PSQL_PORT
- PSQL_USER
- PSQL_PASS
- PSQL_DBNAME
- PSQL_TABLE_NAME
- PSQL_SELECT_FILE (require PSQL_SELECT=1)

53.5 Insertion of Selected Fields Only

When PSQL_SELECT=1, the columns to insert into the DB can be customized with the help of PSQL_SELECT_FILE. The filename defaults to psql-columns.txt in the user plugin folder, e.g., `~/.tranalyzer/plugins`. The format of the file is simply one field name per line with lines starting with a '#' being ignored. For example, to only insert source and destination addresses and ports, create the following file:

```
# Lines starting with a '#' are ignored and can be used to add comments
srcIP
srcPort
dstIP
dstPort
```

53.6 Post-Processing

The following queries can be used to analyze bitfields in PostgreSQL:

- Select all A flows:

```
SELECT to_hex("flowStat"::bigint), *
FROM flow
WHERE ("flowStat"::bigint & 1) = 0::bigint
```

- Select all IPv4 flows:

```
SELECT *
FROM flow
WHERE ("flowStat"::bigint & x'4000'::bigint) != 0::bigint
```

- Select all IPv6 flows:

```
SELECT to_hex("flowStat"::bigint), *
FROM flow
WHERE ("flowStat"::bigint & x'8000'::bigint) != 0::bigint
```

53.7 Example

```
# Run Tranalyzer
$ t2 -r file.pcap

# Connect to the PostgreSQL database
$ psql -U postgres -d tranalyzer

# Number of flows
tranalyzer=# SELECT COUNT(*) FROM flow;

# 10 first srcIP/dstIP pairs
tranalyzer=# SELECT "srcIP", "dstIP" FROM flow LIMIT 10;

# All flows from 1.2.3.4 to 1.2.3.5
tranalyzer=# SELECT * FROM flow WHERE "srcIP" = '1.2.3.4' AND "dstIP" = '1.2.3.5';
```

For examples of more complex queries, have a look in \$T2HOME/scripts/t2fm/sql/.

53.7.1 Clean up an existing database

```
# Connect to the PostgreSQL database
$ psql -U postgres

# Drop the database
tranalyzer=# DROP DATABASE tranalyzer;
```

54 pwX

54.1 Description

The pwX plugin extracts usernames and passwords from different plaintext protocols. This plugin produces only output to the flow file. Configuration is achieved by user defined compiler switches in `src/pwX.h`.

54.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Variable	Default	Description
PWX_USERNAME	1	Output the username
PWX_PASSWORD	1	Output the password
PWX_FTP	1	Extract FTP authentication
PWX_POP3	1	Extract POP3 authentication
PWX_IMAP	1	Extract IMAP authentication
PWX_SMTP	1	Extract SMTP authentication
PWX_HTTP_BASIC	1	Extract HTTP Basic Authorization
PWX_HTTP_PROXY	1	Extract HTTP Proxy Authorization
PWX_HTTP_GET	1	Extract HTTP GET authentication
PWX_HTTP_POST	1	Extract HTTP POST authentication
PWX_IRC	1	Extract IRC authentication
PWX_TELNET	1	Extract Telnet authentication
PWX_LDAP	1	Extract LDAP bind request authentication
PWX_PAP	1	Extract PAP (Password Authentication Protocol) authentication
PWX_STATUS	1	Extract authentication status (success, error, ...).
PWX_DEBUG	0	Activate debug output.

54.3 Flow File Output

The pwX plugin outputs the following columns:

Name	Type	Description	Flags
pwxType	U8	Authentication type	
pwxUser	S	Extracted username	PWX_USERNAME!=0
pwxPass	S	Extracted password	PWX_PASSWORD!=0
pwxStatus	U8	Authentication status	PWX_STATUS!=0

54.3.1 pwxType

The pwxType column is to be interpreted as follows:

pwxType	Description
0	No password or username extracted
1	FTP authentication
2	POP3 authentication
3	IMAP authentication
4	SMTP authentication
5	HTTP Basic Authorization
6	HTTP Proxy Authorization
7	HTTP GET authentication
8	HTTP POST authentication
9	IRC authentication
10	Telnet authentication
11	LDAP authentication
12	PAP authentication

54.3.2 pwxStatus

The pwxStatus column is to be interpreted as follows:

pwxStatus	Description
0	Authentication status is unknown
1	Authentication was successful
2	Authentication failed

54.4 Plugin Report Output

The number of passwords extracted is reported.

55 quicDecode

55.1 Description

The quicDecode plugin analyzes QUIC (IETF) traffic.

55.2 Dependencies

If QUIC_DECODE_TLS=1, then **libssl** is required.

QUIC_DECODE_TLS=1		
Ubuntu:	sudo apt-get install	libssl-dev
Arch:	sudo pacman -S	openssl
openSUSE:	sudo zypper install	libopenssl-devel
Red Hat/Fedora ²⁸ :	sudo dnf install	openssl-devel
macOS ²⁹ :	brew install	openssl@1.1

55.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
QUIC_SPKT_TYPE_STR	1	Format of packet type in packet mode 0: number, 1: string
QUIC_DECODE_TLS	1	0: do not decrypt QUIC Initial packets 1: decrypt TLS 1.3 handshake in QUIC Initial packets Use with the sslDecode plugin to extract the SNI and JA3 fingerprint
QUIC_DEBUG	0	0: do not print any debug messages 1: print debug messages

55.4 Flow File Output

The quicDecode plugin outputs the following columns:

Column	Type	Description	Flags
quicStat	H8	Status	
quicVersion	H32	Version	
quicFlags	H8	Flags	
quicPktTypes	H8	Packet Types	
quicDCID	SC	Destination Connection ID	
quicSCID	SC	Source Connection ID	

²⁸If the `dnf` command could not be found, try with `yum` instead

²⁹Brew is a packet manager for macOS that can be found here: <https://brew.sh>

Column	Type	Description	Flags
quicDCID	SC	Original Destination Connection ID (Retry)	

55.4.1 quicStat

The quicStat column is to be interpreted as follows:

quicStat	Description
0x01	Flow is QUIC
0x02	Handshake (Packet Type is 2)
0x04	Version negotiation (version is 0)
0x08	Version changed
0x10	Destination Connection ID changed
0x20	Source Connection ID changed
0x40	Original Destination Connection ID changed
0x80	Packet was snapped (t2buf failed)

55.4.2 quicVersion

The quicVersion column is to be interpreted as follows³⁰:

quicVersion	Description
0x00000000[0001-ffff]	Standardized versions of QUIC
0x454747[00-ff]	NetApp quant
0x50435130	Private Octopus Picoquic internal test version
0x5130303[1-9]	Google QUIC 01–09 (Q001–Q009)
0x5130313[0-9]	Google QUIC 10–19 (Q010–Q019)
0x5130323[0-9]	Google QUIC 20–29 (Q020–Q029)
0x5130333[0-9]	Google QUIC 30–39 (Q030–Q039)
0x5130343[0-9]	Google QUIC 40–49 (Q040–Q049)
0x51474f[00-ff]	quic-go (QGO[0–255])
0x91c170[00-ff]	quicly (qicly0[0–255])
0abcd000[0-f]	Microsoft WinQuic
0xf10000[00-ff]	IETF QUIC-LB
0xf123f0c[0-f]	Mozilla MozQuic
0xfaceb00[0-f]	Facebook mvfst
0xff[00000-ffffff]	IETF QUIC draft-xx ³¹
0xf0f0f0f[0-f]	ETH Zürich Measurability experiments
0xf0f0f1f[0-f]	Telecom Italia Measurability experiments

³⁰For a more exhaustive list, refer to <https://github.com/quicwg/base-drafts/wiki/QUIC-Versions>

³¹The latest draft is draft-ietf-quic-transport-34 with version 0xff000022

55.4.3 quicFlags

The `quicFlags` column is to be interpreted as follows:

<code>quicFlags</code>	Description
0x03	Packet Number Length (Long Header)
0x0c	Reserved (Long Header)
0x20	Spin Bit (Short Header)
0x30	Packet Type (Long Header)
0x40	Fixed Bit
0x80	Long Header

55.4.4 quicPktTypes

The `quicPktTypes` column is to be interpreted as follows:

<code>quicPktTypes</code>	Description
2^0 (=0x01)	Initial
2^1 (=0x02)	0-RTT
2^2 (=0x04)	Handshake
2^3 (=0x08)	Retry

55.5 Packet File Output

In packet mode (-s option), the `quicDecode` plugin outputs the following columns:

Column	Type	Description	Flags
<code>quicFlags</code>	H8	Flags	
<code>quicPktType</code>	S/U8	Packet Type	
<code>quicVersion</code>	SC	Version	
<code>quicDCID</code>	SC	Destination Connection ID	
<code>quicSCID</code>	SC	Source Connection ID	
<code>quicODCID</code>	SC	Original Destination Connection ID	
<code>quicPktNum</code>	U32	Packet Number	

55.6 Plugin Report Output

The following information is reported:

- Number of QUIC packets
- Number of QUIC Initial packets
- Number of QUIC 0-RTT packets
- Number of QUIC Handshake packets
- Number of QUIC Retry packets

55.7 Known Bugs and Limitations

- The quicDecode plugin assumes every UDP packet on port 443 or 4433 from after 2015³² is QUIC...

55.8 References

- [draft-ietf-quic-applicability](#): Applicability of the QUIC Transport Protocol
- [draft-ietf-quic-bit-grease](#): Greasing the QUIC Bit
- [draft-ietf-quic-datagram](#): An Unreliable Datagram Extension to QUIC
- [draft-ietf-quic-http](#): Hypertext Transfer Protocol Version 3 (HTTP/3)
- [draft-ietf-quic-invariants](#): Version-Independent Properties of QUIC
- [draft-ietf-quic-load-balancers](#): QUIC-LB: Generating Routable QUIC Connection IDs
- [draft-ietf-quic-manageability](#): Manageability of the QUIC Transport Protocol
- [draft-ietf-quic-qpack](#): QPACK: Header Compression for HTTP/3
- [draft-ietf-quic-recovery](#): QUIC Loss Detection and Congestion Control
- [draft-ietf-quic-tls](#): Using TLS to Secure QUIC
- [draft-ietf-quic-transport](#): QUIC: A UDP-Based Multiplexed and Secure Transport
- [draft-ietf-quic-version-negotiation](#): Compatible Version Negotiation for QUIC

³²QUIC was submitted for standardization to the IETF in 2015

56 radiusDecode

56.1 Description

The radiusDecode plugin analyzes RADIUS traffic.

56.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
RADIUS_CNTS	1	Output counts, necessary for FORCE_MODE
RADIUS_AVPTYPE	1	Output AVP Types
RADIUS_NAS	1	Output NAS info
RADIUS_FRAMED	1	Output framed info
RADIUS_TUNNEL	1	Output tunnel info
RADIUS_ACCT	1	Output accounting info
RADIUS_NMS	0	Codes and AVP types format: 0: No code/type output 1: Values, 2: Names
RAD_CNTMX	20	Maximum number of codes/AVP types
RADIUS_STRMAX	128	Maximum length for strings

56.3 Flow File Output

The radiusDecode plugin outputs the following columns:

Column	Type	Description	Flags
radiusStat	H8	Status	
radiusAxsReq_Acc_Rej_Chal	4xU16	Access-Request/Accept/Reject/Challenge	RADIUS_CNTS=1
radiusAccReq_Resp	U16_U16	Accounting-Request/Response	RADIUS_CNTS=1
radiusAccStart_Stop	U16_U16	Accounting Start/Stop	RADIUS_CNTS=1
radiusCodes	R(U8)	Radius codes	RADIUS_NMS=1
radiusCodeNms	R(S)	Radius code names	RADIUS_NMS=2
radiusAVPTypes	R(U8)	AVP types	RADIUS_AVPTYPE=1 && RADIUS_NMS=1
radiusAVPTypeNms	R(S)	AVP type names	RADIUS_AVPTYPE=1 && RADIUS_NMS=2
radiusUser	S	Username	
radiusPW	S	Password	
radiusServiceType	U32	Service type	
radiusLoginService	U32	Login-Service	
radiusVendor	U32	Vendor Id (SMI)	

If RADIUS_NAS=1, the following columns are displayed:

Column	Type	Description	Flags
radiusNasId	S	NAS Identifier	
radiusNasIp	IP4	NAS IP address	
radiusNasPort	U32	NAS IP port	
radiusNasPortType	U32	NAS port type	
radiusNasPortId	S	NAS port Id	

If RADIUS_FRAMED=1, the following columns are displayed:

radiusFramedIp	IP4	Framed IP address
radiusFramedMask	IP4	Framed IP netmask
radiusFramedProto	U32	Framed protocol
radiusFramedComp	U32	Framed compression
radiusFramedMtu	U32	Framed MTU

If RADIUS_TUNNEL=1, the following columns are displayed:

radiusTunnel_Medium	U32_U32	Tunnel type and medium type
radiusTunnelCli	S	Tunnel client endpoint
radiusTunnelSrv	S	Tunnel server endpoint
radiusTunnelCliAId	S	Tunnel client authentication Id
radiusTunnelSrvAId	S	Tunnel server authentication Id
radiusTunnelPref	S	Tunnel preference

If RADIUS_ACCT=1, the following columns are displayed:

radiusAcctSessId	S	Accounting session Id
radiusAcctSessTime	U32	Accounting session time (seconds)
radiusAcctStatType	U32	Accounting status type
radiusAcctTerm	U32	Accounting terminate cause
radiusAcctInOct_OutOct	U32_U32	Accounting input/output octets
radiusAcctInPkt_OutPkt	U32_U32	Accounting input/output packets
radiusAcctInGw_OutGw	U32_U32	Accounting input/output gigawords
radiusConnInfo	S	User connection info
radiusFilterId	S	Filter Identifier
radiusCalledId	S	Called Station Identifier
radiusCallingId	S	Calling Station Identifier
radiusReplyMsg	S	Reply message

56.3.1 radiusStat

The radiusStat column is to be interpreted as follows:

radiusStat	Description
2^0 (=0x01)	Flow is RADIUS

radiusStat	Description
2^1 (=0x02)	Authentication and configuration traffic
2^2 (=0x04)	Accounting traffic
2^3 (=0x08)	—
2^4 (=0x10)	Connection successful
2^5 (=0x20)	Connection failed
2^6 (=0x40)	—
2^7 (=0x80)	Malformed packet

56.3.2 radiusServiceType

The `radiusServiceType` column is to be interpreted as follows:

radiusServiceType	Description
1	Login
2	Framed
3	Callback Login
4	Callback Framed
5	Outbound
6	Administrative
7	NAS Prompt
8	Authenticate Only
9	Callback NAS Prompt
10	Call Check
11	Callback Administrative
12	Voice
13	Fax
14	Modem Relay
15	IAPP-Register
16	IAPP-AP-Check
17	Authorize Only
18	Framed-Management
19	Additional-Authorization

56.3.3 radiusLoginService

The `radiusLoginService` column is to be interpreted as follows:

radiusLoginService	Description
0	Telnet
1	Rlogin
2	TCP Clear
3	PortMaster (proprietary)
4	LAT

radiusLoginService	Description
5	X25-PAD
6	X25-T3POS
7	Unassigned
8	TCP Clear Quiet (suppresses any NAS-generated connect string)

56.3.4 radiusVendor

The radiusVendor column represents the SMI Network Management Private Enterprise Codes which can be found at <https://www.iana.org/assignments/enterprise-numbers>. Alternatively use grep on the file vendor.txt as follows: grep id vendor.txt, where id is the actual Id reported by Tranalyzer, e.g., 4874 for Juniper.

56.3.5 radiusNasPortType

The radiusNasPortType column is to be interpreted as follows:

radiusNasPortType	Description
0	Async
1	Sync
2	ISDN Sync
3	ISDN Async V.120
4	ISDN Async V.110
5	Virtual
6	PIAFS
7	HDLC Clear Channel
8	X.25
9	X.75
10	G.3 Fax
11	SDSL - Symmetric DSL
12	ADSL-CAP - Asymmetric DSL, Carrierless Amplitude Phase Modulation
13	ADSL-DMT - Asymmetric DSL, Discrete Multi-Tone
14	IDSL - ISDN Digital Subscriber Line
15	Ethernet
16	xDSL - Digital Subscriber Line of unknown type
17	Cable
18	Wireless - Other
19	Wireless - IEEE 802.11
20	Token-Ring
21	FDDI
22	Wireless - CDMA2000
23	Wireless - UMTS
24	Wireless - 1X-EV
25	IAPP
26	FTTP - Fiber to the Premises
27	Wireless - IEEE 802.16
28	Wireless - IEEE 802.20

radiusNasPortType	Description
29	Wireless - IEEE 802.22
30	PPPoA - PPP over ATM
31	PPPoEoA - PPP over Ethernet over ATM
32	PPPoEoE - PPP over Ethernet over Ethernet
33	PPPoEoVLAN - PPP over Ethernet over VLAN
34	PPPoEoQinQ - PPP over Ethernet over IEEE 802.1QinQ
35	xPON - Passive Optical Network
36	Wireless - XGP
37	WiMAX Pre-Release 8 IWK Function
38	WiMAX-WIFI-IWK: WiMAX WIFI Interworking
39	WiMAX-SFF: Signaling Forwarding Function for LTE/3GPP2
40	WiMAX-HA-LMA: WiMAX HA and or LMA function
41	WiMAX-DHCP: WiMAX DHCP service
42	WiMAX-LBS: WiMAX location based service
43	WiMAX-WVS: WiMAX voice service

56.3.6 radiusFramedProto

The `radiusFramedProto` column is to be interpreted as follows:

radiusFramedProto	Description
1	PPP
2	SLIP
3	AppleTalk Remote Access Protocol (ARAP)
4	Gandalf proprietary SingleLink/MultiLink protocol
5	Xylogics proprietary IPX/SLIP
6	X.75 Synchronous
7	GPRS PDP Context

56.3.7 radiusFramedComp

The `radiusFramedComp` column is to be interpreted as follows:

radiusFramedComp	Description
0	None
1	VJ TCP/IP header compression
2	IPX header compression
3	Stac-LZS compression

56.3.8 radiusTunnel_Medium

The `radiusTunnel_Medium` column is to be interpreted as follows:

radiusTunnel	Description
1	Point-to-Point Tunneling Protocol (PPTP)
2	Layer Two Forwarding (L2F)
3	Layer Two Tunneling Protocol (L2TP)
4	Ascend Tunnel Management Protocol (ATMP)
5	Virtual Tunneling Protocol (VTP)
6	IP Authentication Header in the Tunnel-mode (AH)
7	IP-in-IP Encapsulation (IP-IP)
8	Minimal IP-in-IP Encapsulation (MIN-IP-IP)
9	IP Encapsulating Security Payload in the Tunnel-mode (ESP)
10	Generic Route Encapsulation (GRE)
11	Bay Dial Virtual Services (DVS)
12	IP-in-IP Tunneling
13	Virtual LANs (VLAN)

radiusMedium	Description
1	IPv4 (IP version 4)
2	IPv6 (IP version 6)
3	NSAP
4	HDLC (8-bit multidrop)
5	BBN 1822
6	802 (includes all 802 media plus Ethernet “canonical format”)
7	E.163 (POTS)
8	E.164 (SMDS, Frame Relay, ATM)
9	F.69 (Telex)
10	X.121 (X.25, Frame Relay)
11	IPX
12	Appletalk
13	Decnet IV
14	Banyan Vines
15	E.164 with NSAP format subaddress

56.3.9 radiusAcctStatType

The `radiusAcctStatType` column is to be interpreted as follows:

radiusAcctStatType	Description
1	Start
2	Stop
3	Interim-Update
7	Accounting-On
8	Accounting-Off
9	Tunnel-Start
10	Tunnel-Stop

radiusAcctStatType	Description
11	Tunnel-Reject
12	Tunnel-Link-Start
13	Tunnel-Link-Stop
14	Tunnel-Link-Reject
15	Failed

56.3.10 radiusAcctTerm

The `radiusAcctTerm` column is to be interpreted as follows:

radiusAcctTerm	Description
1	User Request
2	Lost Carrier
3	Lost Service
4	Idle Timeout
5	Session Timeout
6	Admin Reset
7	Admin Reboot
8	Port Error
9	NAS Error
10	NAS Request
11	NAS Reboot
12	Port Unneeded
13	Port Preempted
14	Port Suspended
15	Service Unavailable
16	Callback
17	User Error
18	Host Request
19	Supplicant Restart
20	Reauthentication Failure
21	Port Reinitialized
22	Port Administratively Disabled
23	Lost Power

56.4 Packet File Output

In packet mode (`-s` option), the `radiusDecode` plugin outputs the following columns:

Column	Type	Description	Flags
<code>radiusStat</code>	H8	Status	
<code>radiusCode</code>	U8	Code	RADIUS_NMS=1
<code>radiusCodeNm</code>	S	Code name	RADIUS_NMS=2
<code>radiusAVPTypes</code>	R(U8)	AVP types	RADIUS_AVPTYPE=1&&RADIUS_NMS=1

Column	Type	Description	Flags
radiusAVPTypeNms	R(S)	AVP type names	RADIUS_AVPTYPE=1&&RADIUS_NMS=2

56.5 Monitoring Output

In monitoring mode, the radiusDecode plugin outputs the following columns:

Column	Type	Description	Flags
radiusPkts	U64	Number of RADIUS packets	
radiusAxsPkts	U64	Number of Access	
radiusAxsAccPkts	U64	Number of Access-Accept	
radiusAxsRejPkts	U64	Number of Access-Reject	
radiusAccPkts	U64	Number of Accounting packets	

56.6 Plugin Report Output

The following information is reported:

- Aggregated `radiusStat`
- Number of RADIUS packets
- Number of Access, Access-Accept, Access-Reject and Accounting packets

56.7 References

- [RFC2865](#): Remote Authentication Dial In User Service (RADIUS)
- [RFC2866](#): RADIUS Accounting
- [RFC2867](#): RADIUS Accounting Modifications for Tunnel Protocol Support
- [RFC2868](#): RADIUS Attributes for Tunnel Protocol Support
- [RFC2869](#): RADIUS Extensions
- <https://www.iana.org/assignments/radius-types/radius-types.xhtml>

57 regexHyperscan

57.1 Description

This plugin applies regexes on the network traffic using the [Hyperscan library](#)³³. The regexes can be applied on the whole flow or per packet from layer 7.

57.2 Dependencies

57.2.1 External Libraries

This plugin depends on the Hyperscan library which is included in this plugin. In order to compile it, the following tools and libraries are needed.

Ubuntu:	sudo apt-get install cmake g++ libboost-dev ragel
Arch:	sudo pacman -S boost cmake ragel
Red Hat/Fedora ³⁴ :	sudo dnf install boost-devel cmake gcc-c++ ragel

57.2.2 Required Files

The file `hsregexes.txt` contains the regexes and their corresponding ID. The lines starting with % are comments. The other lines must contain two or three columns:

Column	Description
1	A string ID which will appear in the flow output if the flow matches the regex in column 2.
2	A regex in the Hyperscan format describe in Section 57.3 .
3	Optional. Whether (1) to extract flows matching regex using the liveXtr plugin, or not (0).

57.3 Hyperscan regex format

Each regex must have the following format: /pattern flags

The **pattern** use the PCRE syntax with some limitation explained in the [Hyperscan documentation](#)³⁵.

The **flags** are optional and are described in the [Hyperscan documentation](#)³⁶. The following table provides a correspondence between the letters used in this plugin regex format and the values in the Hyperscan documentation.

Flag	Description
i	HS_FLAG_CASELESS
s	HS_FLAG_DOTALL
m	HS_FLAG_MULTILINE

³³<https://github.com/intel/hyperscan>

³⁴If the dnf command could not be found, try with yum instead

³⁵<https://intel.github.io/hyperscan/dev-reference/compilation.html#pattern-support>

³⁶https://intel.github.io/hyperscan/dev-reference/api_constants.html#pattern-flags

Flag	Description
H	HS_FLAG_SINGLEMATCH (enabled by default)
V	HS_FLAG_ALLOWEMPTY
8	HS_FLAG_UTF8
W	HS_FLAG_UCP

57.4 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
RHS_STREAMING	1	1: Apply the regexes on the whole flow as a stream. 0: Apply the regexes per packet.
RHS_RELOADING	1	Automatically reload the regex file when modified.
RHS_EXTRACT_OPPOSITE	1	Also extract the opposite flow when regex match.
RHS_MAX_FLOW_MATCH	16	Max. number of regexes which can match on a flow.
RHS_REGEX_FILE	"hsregexes.txt"	The name of the file described in Section 57.2.2 .

57.5 Flow File Output

The regexHyperscan plugin outputs the following columns:

Column	Type	Description
hsregexes	RS	IDs of all regexes matching this flow

58 regex_pcre

58.1 Description

The regex_pcre plugin provides a full PCRE compatible regex engine.

58.2 Dependencies

58.2.1 External Libraries

This plugin depends on the **pcre** library.

Ubuntu:	sudo apt-get install libpcre3-dev
Arch:	sudo pacman -S pcre pcre2
openSUSE:	sudo zypper install pcre-devel
Red Hat/Fedora ³⁷ :	sudo dnf install pcre-devel
macOS ³⁸ :	brew install pcre

58.2.2 Required Files

The file `regexfile.txt` is required (automatically generated from `scripts/regfile.txt`). Refer to Section 58.3.4 for more details.

58.3 Configuration Flags

58.3.1 regfile_pcre.h

The compiler constants in `regfile_pcre.h` control the pre-processing and compilation of the rule sets supplied in the regex file during the initialization phase of Tranalyzer.

Name	Default	Description	Flags
RULE_OPTIMIZE	1	0: No opt rules allocated 1: Allocate opt rule structure and compile regex	
REGEX_MODE	PCRE_DOTALL	Regex compile time options	
PREIDMX	4	Max number of node predecessors	

58.3.2 regex_pcre.h

The compiler constants in `regex_pcre.h` control the execution and the output the rule matches.

Variable	Default	Description	Flags
EXPERTMODE	0	0: Alarm with highest severity: class type and severity, 1: full info	

³⁷If the `dnf` command could not be found, try with `yum` instead

³⁸Brew is a packet manager for macOS that can be found here: <https://brew.sh>

Variable	Default	Description	Flags
PKTTIME	0	0: no time, 1: timestamp when rule matched	
AGGR	0	1: Aggregate alarms	
SALRMFLG	0	1: enable sending FL_ALARM for pcapd	
MAXREGPOS	30	Maximal # of matches stored / flow	
RGX_POSIX_FILE	"regexfile.txt"	Name of regex file under ./tranalyzer/plugins	
OVECCOUNT	3	Value % 3	

58.3.3 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- RGX_POSIX_FILE

58.3.4 regexfile.txt

The *scripts/regexfile.txt* file has the following format:

```
#ID      PreID    Flags     ClassID Severity          Sel       Regexmode      FlwStat Proto      srcPort
dstPort offset   Regex

# standalone rule: Alarm, start L7, Regexmode: default, select FlwStat: Req; Proto, dstPort
1       0        0x10     15      3        0x8000000d      0x00000000      0x00000000      6        0
           80      0        (OPTIONS|GET|HEAD|POST|PUT|DELETE|TRACE|CONNECT) [^\r\n]*\u7avi*\.\bin

# standalone rule: Alarm, disabled, start L7, select Regexmode: (PCRE_CASELESS|PCRE_DOTALL),
#                   FlwStat: Teredo, IPv6, Vlan, Repl; Proto, srcPort
3       0        0x10     15      3        0x0800000e      0x00000005      0x00088101      6        80
           0        0        \x31\xDB\x8D\x43\x0D\xCD\x80\x66.*\x31

# standalone rule: Alarm, start L7, Regexmode: default, FlwStat: IPv4, Rply
4       0        0x10     15      3        0x8000000c      0x00000000      0x00004001      6        80
           0        20      \x38\x55\x42\x66\xe2\xb5\x34.*\xb5\x95\xbb

# standalone rule, Alarm, start L7, select Regexmode: (PCRE_CASELESS|PCRE_DOTALL)
100    0        0x10     1      0        0x88000000      0x00000005      0x00000000      6        0
           80      0        ^http/1.0

# root rules to following tree, Reset if leaf fires
202    0        0x40     10     4        0x80000000      0x00000000      0x00000001      6        0
           80      0        (GET|PUT).*update/u7avi1777u1705ff.bin
203    202,4    0x41     20     4        0x88000000      0x00000005      0x00000001      6        0
           80      0        302 (?i)Found

# successors and predecessors, Reset if leaf fires
204    202,203 0x41     43     5        0x80000000      0x00000000      0x00000001      6        0
           21      0        (?i)\.exe

# successors 206 & 205 to 204 AND ruleset, don't reset tree if 205 fires
205    204    0x16     40     4        0x80000002      0x00000000      0x00000000      6        0
           20      0        ^get .*porno.*
206    204    0x56     35     6        0x8000000c      0x00000000      0x00000001      6        0
           21      0        igfxzoom\.exe
```

Lines starting with a '#' denote a comment line and will be ignored. All kind of rule trees can be formed using rules also acting on multiple packets using different ID's and Predecessor as outlined in the example above. Regex rules with the same ID denote combined predecessors to other rules. Default is an OR operation unless ANDPin bits are set. These bits denote the different inputs to a bitwise AND. The output is then provided to the successor rule which compares with the ANDMask bit field whether all necessary rules are matched. Then an evaluation of the successor rule can take place. Thus, arbitrary rule trees can be constructed and results of predecessors can be used for multiple successor rules. The variable Flags controls the basic PCRE rule interpretation and the flow alarm production (see the table below), e.g. only if bit eight is set and alarm flow output is produced. ClassID and Severity denote information being printed in the flow file if the rule fires.

Flags	Description
2^0 (=0x01)	Predecessor OPS
2^1 (=0x02)	Predecessor OPS
2^2 (=0x04)	Leaf
2^3 (=0x08)	—
2^4 (=0x10)	Print alarm to flow file
2^5 (=0x20)	Rule active only in flow boundary
2^6 (=0x40)	Reset REG_F_MTCH tree if match
2^7 (=0x80)	Internal: Regex match

Predecessor OPS	OP	Description
0x00	NONE	None, solitary rule
0x01	AND	and(pred1, pred2, ...)
0x02	OR	or(pred1, pred2, ...)
0x03	XOR	xor(pred1, pred2, ...)

The Sel column controls the header selection of a rule in the lower nibble and the start of regex evaluation in the higher nibble. The position of the bits in the control byte are outlined below:

Sel	Description
2^0 (=0x00000001)	Activate srcPort field
2^1 (=0x00000002)	Activate dstPort field
2^2 (=0x00000004)	Activate L4Proto field
2^3 (=0x00000008)	Activate flowStat field
2^{27} (=0x08000000)	PCRE mode active; otherwise default
2^{28} (=0x10000000)	Header start: Layer 2
2^{29} (=0x20000000)	Header start: Layer 3
2^{30} (=0x40000000)	Header start: Layer 4
2^{31} (=0x80000000)	Header start: Layer 7

Bit 0 - 27 selects the first 32 bit of flowStat, the protocol, source and destination port will be evaluated per rule, all others will be ignored. The flowStat field might contain other bits meaning more selection options in future. The

offset column depicts the start of the regex evaluation from the selected header start, default value 0. The Regex column accepts a full PCRE regex term. If the regex is not correct, the rule will be discarded displaying an error message in the Tranalyzer report.

The regexmode column denotes the mode of regex compilation and execution, listed below. If **0x00000000** then the default defined by REGEX_MODE is used.

regexmode	Name	Description
2^0 (=0x00000001)	PCRE_CASELESS	Compile
2^1 (=0x00000002)	PCRE_MULTILINE	Compile
2^2 (=0x00000004)	PCRE_DOTALL	Compile
2^3 (=0x00000008)	PCRE_EXTENDED	Compile
2^4 (=0x00000010)	PCRE_ANCHORED	Compile, DFA exec
2^5 (=0x00000020)	PCRE_DOLLAR_ENDONLY	Compile
2^6 (=0x00000040)	PCRE_EXTRA	Compile
2^7 (=0x00000080)	PCRE_NOTBOL	Exec, DFA exec
2^8 (=0x00000100)	PCRE_NOTEOL	Exec, DFA exec
2^9 (=0x00000200)	PCRE_UNGREEDY	Compile
2^{10} (=0x00000400)	PCRE_NOTEEMPTY	Exec, DFA exec
2^{11} (=0x00000800)	PCRE_UTF8	Compile
2^{12} (=0x00001000)	PCRE_NO_AUTO_CAPTURE	Compile
2^{13} (=0x00002000)	PCRE_NO_UTF8_CHECK	Compile, DFA exec
2^{14} (=0x00004000)	PCRE_AUTO_CALLOUT	Compile
2^{15} (=0x00008000)	PCRE_PARTIAL_SOFT	Exec, DFA exec
2^{16} (=0x00010000)	PCRE_DFA_SHORTEST	DFA exec
2^{17} (=0x00020000)	PCRE_DFA_RESTART	DFA exec
2^{18} (=0x00040000)	PCRE_FIRSTLINE	Compile
2^{19} (=0x00080000)	PCRE_DUPNAMES	Compile
2^{20} (=0x00100000)	PCRE_NEWLINE_CR	Compile, DFA exec
2^{21} (=0x00200000)	PCRE_NEWLINE_LF	Compile, DFA exec
2^{22} (=0x00400000)	PCRE_NEWLINE_ANY	Compile, DFA exec
2^{23} (=0x00800000)	PCRE_BSR_ANYCRLF	Compile, DFA exec
2^{24} (=0x01000000)	PCRE_BSR_UNICODE	Compile, DFA exec
2^{25} (=0x02000000)	PCRE_JAVASCRIPT_COMPAT	Compile
2^{26} (=0x04000000)	PCRE_NO_START_OPTIMIZE	Compile, DFA exec
2^{27} (=0x08000000)	PCRE_PARTIAL_HARD	Exec, DFA exec
2^{28} (=0x10000000)	PCRE_NOTEEMPTY_ATSTART	Exec, DFA exec
2^{29} (=0x20000000)	PCRE_UCP	Compile

58.4 Flow File Output

The regex_pcre plugin outputs the following columns:

Column name	Type	Description	Flags
rgxCnt	U16	Number of regex alarms	
rgxRID_cType_sev	R(U16_U8_U8)	Regex ID, class type and severity	EXPERTMODE=0
If EXPERTMODE=1, the following columns are displayed:			
rgxRID_cType_sev_	R(U16_U8_U8_	Regex ID, class type, severity,	PKTTIME=0
pktN_bPos	U32_U16)	packet number and byte position	
rgxRID_cType_sev_	R(U16_U8_U8_	Regex ID, class type, severity,	PKTTIME=1
pktN_bPos_time	U32_U16_TS)	packet number, byte position and time)	

58.5 Packet File Output

In packet mode (-s option), the regex_pcre plugin outputs the following columns:

Column	Type	Description	Flags
rgxCnt	U16	Number of regex alarms	
rgxRID_cType_sev	R(U16_U8_U8)	Regex ID, class type and severity	

58.6 Plugin Report Output

The following information is reported:

- Number of alarms in number of flows with max severity

59 regex_re2

59.1 Description

This plugin applies regexes on the network traffic using the RE2 library: <https://github.com/google/re2>. The regexes are applied per packet from the layer 7.

59.2 Dependencies

59.2.1 External Libraries

This plugin depends on the RE2 library.

Ubuntu 15.10+:	sudo apt-get install g++ libre2-dev
Arch:	sudo pacman -S re2
Gentoo:	sudo emerge re2
openSUSE:	sudo zypper install re2-devel
Red Hat/Fedora ³⁹ :	sudo dnf install gcc-c++ re2-devel
macOS ⁴⁰ :	brew install re2

Alternatively (or for older Ubuntu versions), compile it from source:

```
sudo apt-get install git g+
git clone https://github.com/google/re2.git
cd re2
make
sudo make install
sudo ldconfig
```

59.2.2 Required Files

The file `re2file.txt` contains the regexes and their corresponding ID. The lines starting with % are comments. The other lines must contain two columns:

Column	Description
1	A string ID which will appear in the <code>flow</code> output if the flow matches the regex in column 2.
2	A regex in the RE2 syntax: https://github.com/google/re2/wiki/Syntax

When the regex file located in the plugin folder is modified, it is automatically reloaded by tranalyzer.

59.3 Configuration Flags

The following flags can be used to control the output of the plugin:

³⁹If the `dnf` command could not be found, try with `yum` instead

⁴⁰Brew is a packet manager for macOS that can be found here: <https://brew.sh>

Name	Default	Description
RE2_DEBUG_MESSAGES	0	Activate debug output
RE2_REGEX_FILE	"re2file.txt"	The name of the file described in Section 59.2.2.
RE2_MAX_MEMORY	2^{31} (= 2GB)	Max. memory RE2 can use to build the regex automaton.
RE2_MERGE	1	0: Compiles each regex separately → slower, less memory. 1: Merge all regexes into one automaton → faster, more memory.
RE2_RELOADING	1	Automatically reload regex file when modified. (disabled for macOS, refer to Section 59.6)
RE2_MAX_MATCH_PER_PACKET	8	Max. number of regexes which can match on a single packet.
RE2_MAX_MATCH_PER_FLOW	32	Max. number of regexes which can match on a single flow.

59.4 Flow File Output

The regex_re2 plugin outputs the following columns:

Column	Type	Description	Flags
re2match	RS	IDs of all regexes matching this flow	

59.5 Plugin Report Output

The following information is reported:

- Number of signatures matched
- Number of flows with matched signatures

59.6 Known Bugs and Limitations

- macOS does not provide the inotify library, therefore modified regex files cannot be automatically reloaded.

60 sctpDecode

60.1 Description

The sctpDecode plugin produces a flow based view of SCTP operations between computers for anomaly detection and troubleshooting purposes.

60.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
SCTP_CRCADL32CHK	0	Checksum computation: 0: No checksum computation 1: CRC32 2: ADLER32	
SCTP_CHNKVAL	0	Chunk type representation: 0: chunk type bit field 1: chunk type value 2: chunk type as string	
SCTP_CHNKAGGR	0	Aggregate chunk types	SCTP_CHNKVAL>0
SCTP_TSNREL	0	1: Relative TSN, 0: Absolute TSN	
SCTP_MAXCTYPE	15	Maximum chunk types to store/flow	SCTP_CHNKVAL>0
SCTP_ASMX	10	Maximum ASCONF IP	
SCTP_MXADDR	5	Maximum number of addresses to print in packet mode	

60.3 Flow File Output

The sctpDecode plugin outputs the following columns:

Column	Type	Description Flags	
sctpStat	H8	Status	
sctpDSNum	U16	Data stream number	SCTP_ACTIVATE=1
sctpMaxDSNum	U16	Max. number of data streams	SCTP_ACTIVATE=0
sctpPID	U32	Payload ID	
sctpVTag	H32	Verification tag	
sctpTypeBF	H16	Aggregated type bit field	SCTP_CHNKVAL=0
sctpType	R(U8)	Unique types values	SCTP_CHNKVAL=1
sctpTypeN	R(SC)	Unique types names	SCTP_CHNKVAL=2
sctpCnd_I_A	U16_U16_U16	DATA, INIT and ABORT count	
sctpCFlags	H8	Aggregated chunk flag	
sctpCCBF	H16	Aggregated error cause code bit field	
sctpASIP4	R(SC)	ASCONF IPv4	
sctpASIP6	R(SC)	ASCONF IPv6	
sctpIS	U16	Inbound streams	
sctpOS	U16	Outbound streams	
sctpIARW	U32	Initial Advertised Receiver Window	

Column	Type	Description Flags
sctpIARWMin	U32	Initial Advertised Receiver Window Minimum
sctpIARWMax	U32	Initial Advertised Receiver Window Maximum
sctpARW	F	Advertised Receiver Window

60.3.1 sctpStat

The sctpStat column is to be interpreted as follows:

sctpStat	Description
2^0 (=0x01)	Adler32 error
2^1 (=0x02)	CRC32 error
2^2 (=0x04)	Chunk padded
2^3 (=0x08)	Chunk truncated
2^6 (=0x10)	3 ACK
2^7 (=0x20)	Type Field overflow
2^4 (=0x40)	Do not report
2^5 (=0x80)	Stop processing of the packet

60.3.2 sctpTypeN, sctpTypeBF and sctpType

The sctpTypeN, sctpTypeBF and sctpType columns are to be interpreted as follows:

sctpTypeN	sctpTypeBF	sctpType	Description
0	0x0001	DATA	Payload data
1	0x0002	INIT	Initiation
2	0x0004	INIT_ACK	Initiation acknowledgement
3	0x0008	SACK	Selective acknowledgement
4	0x0010	HEARTBEAT	Heartbeat request
5	0x0020	HEARTBEAT_ACK	Heartbeat acknowledgement
6	0x0040	ABORT	Abort
7	0x0080	SHUTDOWN	Shutdown
8	0x0100	SHUTDOWN_ACK	Shutdown acknowledgement
9	0x0200	ERROR	Operation error
10	0x0400	COOKIE_ECHO	State cookie
11	0x0800	COOKIE_ACK	Cookie acknowledgement
12	0x1000	ECNE	Explicit congestion notification echo (reserved)
13	0x2000	CWR	Congestion window reduced (reserved)
14	0x4000	SHUTDOWN_COMPLETE	Shutdown complete
15	0x8000	AUTH	Authentication chunk

60.3.3 sctpCFlags

The sctpCFlags column is to be interpreted as follows:

sctpCFlags	Description
2^0 (=0x01)	Last segment
2^1 (=0x02)	First segment
2^2 (=0x04)	Ordered delivery
2^3 (=0x08)	Possibly delay SACK
2^4 (=0x10)	—
2^5 (=0x20)	—
2^6 (=0x40)	Transmission sequence number error
2^7 (=0x80)	Association sequence number error

60.3.4 sctpCCBF

The sctpCCBF column is to be interpreted as follows:

sctpCCBF	Description
2^0 (=0x0001)	—
2^1 (=0x0002)	Invalid Stream Identifier
2^2 (=0x0004)	Missing Mandatory Parameter
2^3 (=0x0008)	Stale Cookie Error
2^4 (=0x0010)	Out of Resource
2^5 (=0x0020)	Unresolvable Address
2^6 (=0x0040)	Unrecognized Chunk Type
2^7 (=0x0080)	Invalid Mandatory Parameter
2^8 (=0x0100)	Unrecognized Parameters
2^9 (=0x0200)	No User Data
2^{10} (=0x0400)	Cookie Received While Shutting Down
2^{11} (=0x0800)	Restart of an Association with New Addresses
2^{12} (=0x1000)	User Initiated ABORT
2^{13} (=0x2000)	Protocol Violation
2^{14} (=0x4000)	—
2^{15} (=0x8000)	Error code > 14

60.4 Packet File Output

In packet mode (-s option), the sctpDecode plugin outputs the following columns:

Column	Type	Description	Flags
sctpVTag	H32	Verification tag	
sctpChkSum	H32	Checksum	
sctpCalCRCChkSum	H32	Computed CRC checksum	SCTP_CRCADL32CHK=1
sctpCalADLChkSum	H32	Computed ADLER32 checksum	SCTP_CRCADL32CHK=2

Column	Type	Description	Flags
sctpChunkType_	U8_	Chunk type,	
sid_	U16_	stream identifier,	
flags_	H8_	chunk flags,	
numDPkts_	U16_	DATA count,	
len_	I32_	chunk length,	
pid	U32	Payload ID	
sctpNChunks	U8	Number of chunks	
sctpCCBF	H16	Aggregated error cause code bit field	
sctpARW	U32	Advertised Receiver Window	
sctpPID	U32	Payload ID	
sctpStat	U8	Status	
sctpTSN	U32	Transmission Sequence Number (TSN)	SCTP_TSNREL=0
sctpTSNAck	U32	TSN Acknowledgement	SCTP_TSNREL=0
sctpRelTSN	U32	Relative Transmission Sequence Number (TSN)	SCTP_TSNREL=1
sctpRelTSNAck	U32	Relative TSN Acknowledgement	SCTP_TSNREL=1
sctpASIP4	R(IP4)	ASCONF IPv4	
sctpASIP6	R(IP6)	ASCONF IPv6	

60.5 Plugin Report Output

The following information is reported:

- Aggregated `sctpStat`
- Aggregated `sctpCFlags`
- Aggregated `sctpTypeBF`

61 smbDecode

61.1 Description

The smbDecode plugin analyzes SMBv1 and SMBv2 traffic.

61.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
SMB1_DECODE	0	decode SMB1 (beta)	
SMB_SECBLOB	0	decode security blob (beta)	
SMB_NUM_FNAME	5	number of unique filenames to store	
SMB2_NUM_DIALECT	3	number of SMB2 dialects to store	
SMB1_NUM_DIALECT	3	number of SMB1 dialects to store	SMB1_DECODE=1
SMB1_DIAL_MAXLEN	32	maximum length for SMB1 dialects	SMB1_DECODE=1
SMB2_NUM_STAT	18	number of unique SMB2 header status to store	
SMB1_SAVE_DATA	0	save files (SMB1)	SMB1_DECODE=1
SMB2_SAVE_DATA	0	save files (SMB2)	
SMB_SAVE_AUTH	0	save NTLM authentications	
SMB_NATIVE_NAME_LEN	64	Maximum length for names	
SMB_SAVE_DIR	"/tmp/TranSMB/"	Folder for saved data	SMB_SAVE_DATA=1
SMB_FILE_ID	"File_Id_"	Prefix for output files	SMB_SAVE_DATA=1
SMB_AUTH_FILE	"smb_auth.txt"	File where to store NTLM authentications	SMB_SAVE_AUTH=1
SMB_RM_DATADIR	1	Remove SMB_SAVE_DIR before starting	SMB_SAVE_DATA=1
SMB_FNAME_LEN	512	Maximum length for filenames	
SMB_STRCPY_BEHAVIOR	T2_STRCPY_TRUNC	If data to copy is bigger than buffer: T2_STRCPY_EXIT: exit T2_STRCPY_EMPTY: return an empty buffer ("abcdefg" → "") T2_STRCPY_TRUNC: truncate destination ("abcdefg" → "abcdef") T2_STRCPY_ELLIPSIS: truncate destination with an ellipsis ("abcdefg" → "abc...")	

When saving files, the plugin uses a combination of the file ID and the flow index as name. The file ID can be replaced with the real filename by using the `smbrename` script and the `SMB_GUID_MAP_FILE` (`smb_filenames.txt`) file (See Section [61.5](#)).

61.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- SMB_RM_DATADIR
- SMB_AUTH_FILE
- SMB_SAVE_DIR
- SMB_MAP_FILE
- SMB_FILE_ID

61.3 Flow File Output

The smbDecode plugin outputs the following columns:

Column	Type	Description	Flags
smbStat	H16	Status	
smb1NDialects	U32	Number of requested dialects (SMB1)	
smb1Dialects	RS	SMB1 requested dialects (client: supported, server: chosen)	
smb2NDialects	U32	Number of dialects (SMB2)	
smb2Dialects	RH16	SMB2 dialect revision (client: supported, server: chosen)	
smbNHdrStat	U32	Number of unique SMB2 header status values	
smbHdrStat	RH32	SMB2 list of unique header status	
smbOpcodes	H32	Opcodes	
smbNOpcodes	19x(U32)	Number of records per opcode	
smbPrevSessId	H64	SMB previous session ID	
smbNativeOS	S	SMB native OS	
smbNativeLanMan	S	SMB native LAN Manager	
smbPrimDom	S	SMB primary domain	
smbTargName	S	SMB target name	
smbDomName	S	SMB domain name	
smbUserName	S	SMB user name	
smbHostName	S	SMB host name	
smbNTLMServChallenge	S	SMB NTLM server challenge	
smbNTProofStr	S	SMB NT proof string	
smbSessionKey	S	SMB session key	
smbGUID	S	Client/Server GUID	
smbSessFlags_	H16_	Session flags, secM_	
caps	H8_	Security mode,	
caps	H32	Capabilities	
smbBootT	TS	Server start time	
smbMaxSizeT_R_W	U32_U32_U32	Max transaction/read/write size	
smbPath	S	Full share path name	
smbShareT	H8	Type of share being accessed	
smbShareFlags	H32_	Share flags, caps	
acc	H32_	Capabilities, Access mask	
smbNFiles	U32	Number of accessed files	

Column	Type	Description	Flags
smbFiles	RS	Accessed files	

61.3.1 smbStat

The `smbStat` column is to be interpreted as follows:

smbStat	Description
0x0001	Flow is SMB
0x0002	SMB2 header status list truncated...increase <code>SMB2_NUM_STAT</code>
0x0004	Dialect name truncated...increase <code>SMB1_DIAL_MAXLEN</code>
0x0008	SMB1 dialect list truncated...increase <code>SMB1_NUM_DIALECT</code>
0x0010	SMB2 dialect list truncated...increase <code>SMB_NUM_DIALECT</code>
0x0020	List of accessed files truncated...increase <code>SMB_NUM_FNAME</code>
0x0040	Selected dialect index out of bound...increase <code>SMB1_NUM_DIALECT</code>
0x0080	Selected dialect index out of bound (error or reverse flow not found)
0x0100	Filename truncated...increase <code>SMB_FNAME_LEN</code>
0x0200	—
0x0400	—
0x0800	—
0x1000	Authentication information extracted
0x2000	—
0x4000	—
0x8000	Malformed packets

61.3.2 smb2Dialects

The `smb2Dialects` column is to be interpreted as follows:

smb2Dialects	Description
0x0202	SMB 2.0.2
0x0210	SMB 2.1
0x02ff	Wildcard revision number (≥ 2.1)
0x0300	SMB 3
0x0302	SMB 3.0.2
0x0311	SMB 3.1.1

61.3.3 smbHdrStat

The `smbHdrStat` column is to be interpreted as follows:

smbHdrStat	Description
0x00000000	STATUS_SUCCESS
0x00000103	STATUS_PENDING
0x0000010b	STATUS_NOTIFY_CLEANUP
0x0000010c	STATUS_NOTIFY_ENUM_DIR
0x80000005	STATUS_BUFFER_OVERFLOW
0x80000006	STATUS_NO_MORE_FILES
0xc0000003	STATUS_INVALID_INFO_CLASS
0xc000000d	STATUS_INVALID_PARAMETER
0xc000000f	STATUS_NO_SUCH_FILE
0xc0000010	STATUS_INVALID_DEVICE_REQUEST
0xc0000011	STATUS_END_OF_FILE
0xc0000016	STATUS_MORE_PROCESSING_REQUIRED
0xc0000022	STATUS_ACCESS_DENIED
0xc0000023	STATUS_BUFFER_TOO_SMALL
0xc0000034	STATUS_OBJECT_NAME_NOT_FOUND
0xc0000035	STATUS_OBJECT_NAME_COLLISION
0xc000003a	STATUS_OBJECT_PATH_SYNTAX_BAD
0xc0000043	STATUS_SHARING_VIOLATION
0xc0000061	STATUS_PRIVILEGE_NOT_HELD
0xc000006a	STATUS_WRONG_PASSWORD
0xc000006d	STATUS_LOGON_FAILURE
0xc0000071	STATUS_PASSWORD_EXPIRED
0xc00000ac	STATUS_PIPE_NOT_AVAILABLE
0xc00000ba	STATUS_FILE_IS_A_DIRECTORY
0xc00000bb	STATUS_NOT_SUPPORTED
0xc00000c9	STATUS_NETWORK_NAME_DELETED
0xc00000cc	STATUS_BAD_NETWORK_NAME
0xc0000101	STATUS_DIRECTORY_NOT_EMPTY
0xc0000120	STATUS_CANCELLED
0xc0000128	STATUS_FILE_CLOSED
0xc000019c	STATUS_FS_DRIVER_REQUIRED
0xc0000203	STATUS_USER_SESSION_DELETED
0xc0000225	STATUS_NOT_FOUND
0xc0000234	STATUS_ACCOUNT_LOCKED_OUT
0xc0000257	STATUS_PATH_NOT_COVERED
0xc0000275	STATUS_NOT_A_REPARSE_POINT

For a comprehensive list of the possible status and more extensive description, refer to [\[MS-ERREF\]](#), Section 2.3.1.

61.3.4 smbOpcodes

The smbOpcodes column is to be interpreted as follows:

smbOpcodes	Description
2^0 (=0x00000001)	SMB2_NEGOTIATE

smbOpcodes	Description
2^1 (=0x00000002)	SMB2_SESSION_SETUP
2^2 (=0x00000004)	SMB2_LOGOFF
2^3 (=0x00000008)	SMB2_TREE_CONNECT
2^4 (=0x00000010)	SMB2_TREE_DISCONNECT
2^5 (=0x00000020)	SMB2_CREATE
2^6 (=0x00000040)	SMB2_CLOSE
2^7 (=0x00000080)	SMB2_FLUSH
2^8 (=0x00000100)	SMB2_READ
2^9 (=0x00000200)	SMB2_WRITE
2^{10} (=0x00000400)	SMB2_LOCK
2^{11} (=0x00000800)	SMB2_IOCTL
2^{12} (=0x00001000)	SMB2_CANCEL
2^{13} (=0x00002000)	SMB2_ECHO
2^{14} (=0x00004000)	SMB2_QUERY_DIRECTORY
2^{15} (=0x00008000)	SMB2_CHANGE_NOTIFY
2^{16} (=0x00010000)	SMB2_QUERY_INFO
2^{17} (=0x00020000)	SMB2_SET_INFO
2^{18} (=0x00040000)	SMB2_OPLOCK_BREAK

61.3.5 smbNOpcodes

The `smbNOpcodes` column reports the number of records of each type separated by underscores.

smbNOpcodes	Description
1	Number of SMB2_NEGOTIATE records
2	Number of SMB2_SESSION_SETUP records
3	Number of SMB2_LOGOFF records
4	Number of SMB2_TREE_CONNECT records
5	Number of SMB2_TREE_DISCONNECT records
6	Number of SMB2_CREATE records
7	Number of SMB2_CLOSE records
8	Number of SMB2_FLUSH records
9	Number of SMB2_READ records
10	Number of SMB2_WRITE records
11	Number of SMB2_LOCK records
12	Number of SMB2_IOCTL records
13	Number of SMB2_CANCEL records
14	Number of SMB2_ECHO records
15	Number of SMB2_QUERY_DIRECTORY records
16	Number of SMB2_CHANGE_NOTIFY records
17	Number of SMB2_QUERY_INFO records

smbNOpcodes	Description
18	Number of SMB2_SET_INFO records
19	Number of SMB2_OPLOCK_BREAK records

61.3.6 smbSessFlags_secM_caps

The `smbSessFlags_secM_caps` column is to be interpreted as follows:

smbSessFlags	Description
0x01	Client authenticated as guest user
0x02	Client authenticated as anonymous user
0x04	Server requires encryption of messages on this session (SMB 3.x)

smbSecM	Description
0x01	Security signatures enabled on the server
0x02	Security signatures required by the server

smbCaps	Description
0x01	Server supports the Distributed File System (DFS)
0x02	Server supports leasing
0x04	Server supports multi-credit operation (Large MTU)
0x08	Server supports establishing multiple channels for a single session
0x10	Server supports persistent handles
0x20	Server supports directory leasing
0x40	Server supports encryption

61.3.7 smbShareT

The `smbShareT` column is to be interpreted as follows:

smbShareT	Description
0x01	Physical disk share
0x02	Named pipe share
0x03	Printer share

61.3.8 smbShareFlags_caps_acc

The `smbShareFlags_caps_acc` column is to be interpreted as follows:

smbShareFlags	Description
0x00000001	Specified share is present in a Distributed File System (DFS) tree structure
0x00000002	Specified share is present in a DFS tree structure (DFS root)
If none of the following three bits is set, then the caching policy is “manual”	
0x00000010	Auto caching
0x00000020	VDO Caching
0x00000030	Offline caching MUST NOT occur
0x00000100	Restrict exclusive opens
0x00000200	Force shared delete
0x00000400	Allow namespace caching
0x00000800	Server will filter directory entries based on access permissions of the client
0x00001000	Server will not issue exclusive caching rights on this share
0x00002000	Enable hash V1
0x00004000	Enable hash V2
0x00008000	Encrypt data required

smbShareCaps	Description
0x00000008	Specified share is present in a DFS tree structure
0x00000010	Continuous availability
0x00000020	Scaleout
0x00000040	Cluster
0x00000080	Asymmetric

smbShareAcc	Description
0x00000001	Read access
0x00000002	Write access
0x00000004	Append access
0x00000008	Read extended attributes access
0x00000010	Write extended attributes access
0x00000020	Execute access
0x00000040	Delete child access
0x00000080	Read attributes access
0x00000100	Write attributes access
0x00010000	Delete access
0x00020000	Read access to owner, group and ACL of the SID

smbShareAcc	Description
0x00040000	Owner may write the DAC
0x00080000	Can write owner (take ownership)
0x00100000	Can wait on handle to synchronize on completion of I/O
0x01000000	System security is NOT set
0x02000000	Maximum allowed is NOT set
0x10000000	Generic all is NOT set
0x20000000	Generic execute is NOT set
0x40000000	Generic write is NOT set
0x80000000	Generic read is NOT set

61.4 Plugin Report Output

The following information is reported:

- Aggregated `smbStat`
- Number of SMBv1, SMBv2 and SMBv3 records
- Number of NetNTLMv2 hashes extracted (`SMB_SAVE_AUTH=1`)

61.5 Post-Processing

61.5.1 smbrename

The `smbrename` script can be used to rename and organize the files extracted by the plugin. It must be run from within the `SMB_SAVE_DIR` folder (where the file `smb_filenames.txt` is located). By default, it will replace the file ID with the real filename and organize the files into folders according to their mimetype. Either operation can be performed or not. Try '`smbrename -help`' for more information.

61.5.2 SMB Authentications

When `SMB1_DECODE=1`, `SMB_SECBLOB=1` and `SMB_SAVE_AUTH=1`, the plugin produces a file with suffix `SMB_AUTH_FILE` containing all the NetNTLMv2 hashes extracted from the traffic. The hashes can then be reversed using JohnTheRipper⁴¹ or Hashcat⁴² as follows:

```
john --wordlist=password.lst -format=netntlmv2 FILE_smb_auth.txt
hashcat -m 5600 FILE_smb_auth.txt wordlist.txt
```

61.6 References

- [MS-CIFS]: Common Internet File System (CIFS) Protocol
- [MS-SMB]: Server Message Block (SMB) Protocol

⁴¹<https://github.com/magnumripper/JohnTheRipper>

⁴²<https://hashcat.net>

- [\[MS-SMB2\]](#): Server Message Block (SMB) Protocol Versions 2 and 3
- [\[MS-ERREF\]](#): Windows Error Codes
- [\[MS-SPNG\]](#): Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) Extension
- [\[MS-AUTHSOD\]](#): Authentication Services Protocols Overview
- [\[MS-DTYP\]](#): Windows Data Types
- [\[RFC4178\]](#): The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism

62 smtpDecode

62.1 Description

The smtpDecode plugin processes MAIL header and content information of a flow. The idea is to identify certain mail features and CNAMES.

62.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
SMTP_SAVE	0	1: save content to SMTP_F_PATH	
SMTP_RMDIR	1	Empty SMTP_F_PATH before starting	SMTP_SAVE=1
SMTP_BTFLD	0	1: Bitfield coding of SMTP commands	
SMTP_RCTXT	1	1: print response code text	
SMTP_MXNMLN	70	maximal name length	
SMTP_MXUNMLN	25	maximal user length	
SMTP_MXPNMLN	15	maximal PW length	
SMTP_MAXCNM	8	maximal number rec,trans codes	
SMTP_MAXUNM	5	maximal number server names	
SMTP_MAXPNM	5	maximal number server names	
SMTP_MAXSNM	8	maximal number of server addresses	
SMTP_MAXRNM	8	maximal number of rec EMail addresses	
SMTP_MAXTNM	8	maximal number of trans EMail addresses	
SMTP_F_PATH	"/tmp/SMTPFILES/"	Path for extracted content	
SMTP_NONAME	"nudel"	No name file name	

62.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCTRL>0`):

- `SMTP_RMDIR`
- `SMTP_F_PATH`
- `SMTP_NONAME`

62.3 Flow File Output

The smtpDecode plugin outputs the following columns:

Column	Type	Description	Flags
<code>smtpStat</code>	H8	Status	
<code>smtpCBF</code>	H16	Command codes bitfield	SMTP_BTFLD=1
<code>smtppCC</code>	RSC	Command codes	
<code>smtppRC</code>	RU16	Response codes	
<code>smtppUsr</code>	RS	Users	

Column	Type	Description	Flags
smtppW	RS	Passwords	
smtpSANum	U8	Number of server addresses	
smtpESANum	U8	Number of email sender addresses	
smtpERANum	U8	Number of email receiver addresses	
smtpSA	RS	Server send addresses	
smtpESA	RS	Email send addresses	
smtpERA	RS	Email receive addresses	

62.3.1 smtpStat

The `smtpStat` column is to be interpreted as follows:

smtpStat	Description	Flags
2^0 (=0x01)	SMTP ports found	
2^1 (=0x02)	Authentication pending	
2^2 (=0x04)	Data download pending	SMTP_SAVE=1
2^3 (=0x08)	User password pending	
2^4 (=0x10)	flow write finished	SMTP_SAVE=1
2^5 (=0x20)	—	
2^6 (=0x40)	File error	SMTP_SAVE=1
2^7 (=0x80)	Array overflow	

62.3.2 smtpCBF

The `smtpCBF` column is to be interpreted as follows:

smtpCBF	Description	smtpCBF	Description
2^0 (=0x0001)	HELO	2^8 (=0x0100)	SAML
2^1 (=0x0002)	EHLO	2^9 (=0x0200)	VRFY
2^2 (=0x0004)	MAIL	2^{10} (=0x0400)	EXPN
2^3 (=0x0008)	RCPT	2^{11} (=0x0800)	HELP
2^4 (=0x0010)	DATA	2^{12} (=0x1000)	NOOP
2^5 (=0x0020)	RSET	2^{13} (=0x2000)	QUIT
2^6 (=0x0040)	SEND	2^{14} (=0x4000)	TURN
2^7 (=0x0080)	SOML	2^{15} (=0x8000)	AUTH

62.4 Packet File Output

In packet mode (-s option), the `smtpDecode` plugin outputs the following columns:

Column	Type	Description	Flags
smtpStat	H8	Status	

62.5 Plugin Report Output

The following information is reported:

- Aggregated [smtpStat](#)
- Number of SMTP packets
- Number of SMTP files

62.6 TODO

- fragmentation

63 snmpDecode

63.1 Description

The snmpDecode plugin analyzes SNMP traffic.

63.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
SNMP_STRLEN	64	Maximum length for strings

63.3 Flow File Output

The snmpDecode plugin outputs the following columns:

Column	Type	Description	Flags
snmpStat	H8	Status	
snmpVersion	U8	Version	
snmpCommunity	S	Community (SNMPv1-2)	
snmpUser	S	Username (SNMPv3)	
snmpMsgT	H16	Message types bitfield	
snmpNumReq_	U64_	Number of GetRequest, Next_, Resp_, Set_, Trap1_, Bulk_, Info_, Trap2_, Rep	GetNextRequest, GetResponse, SetRequest, Trapv1, GetBulkRequest, InformRequest, Trapv2, Report packets

63.3.1 snmpStat

The `snmpStat` column is to be interpreted as follows:

snmpStat	Description
0x01	Flow is SNMP
0x02	—
0x04	—
0x08	—
0x10	—
0x20	—
0x40	String was truncated... increase <code>SNMP_STRLEN</code>

snmpStat	Description
0x80	Packet was malformed

63.3.2 snmpVersion

The snmpVersion column is to be interpreted as follows:

snmpVersion	Description
0	SNMPv1
1	SNMPv2c
3	SNMPv3

63.3.3 snmpMsgT and snmpType

The snmpMsgT and snmpType columns are to be interpreted as follows:

snmpMsgT	snmpType	Description
0x0001	0xa0	GetRequest
0x0002	0xa1	GetNextRequest
0x0004	0xa2	GetResponse
0x0008	0xa3	SetRequest
0x0010	0xa4	Trap (v1)
0x0020	0xa5	GetBulkRequest (v2c, v3)
0x0040	0xa6	InformRequest
0x0080	0xa7	Trap (v2c, v3)
0x0100	0xa8	Report

63.4 Packet File Output

In packet mode (-s option), the snmpDecode plugin outputs the following columns:

Column	Type	Description	Flags
snmpVersion	U8	Version	
snmpCommunity	S	Community	
snmpUser	S	Username (SNMPv3)	
snmpType	H8	Message type	

63.5 Plugin Report Output

The following information is reported:

- Number of SNMP packets

- Number of SNMP GetRequest packets
- Number of SNMP GetNextRequest packets
- Number of SNMP GetResponse packets
- Number of SNMP SetRequest packets
- Number of SNMP Trap v1 packets
- Number of SNMP GetBulkRequest packets
- Number of SNMP InformRequest packets
- Number of SNMP Trap v2 packets
- Number of SNMP Report packets

64 socketSink

64.1 Description

This plugin is a socket interface of Tranalyzer. The idea is to interface one or many distributed Tranalyzer instances with a central server post-processing and visualizing its data. The plugin also implements the Alarm Mode being activated by ALARM_MODE=1 in the core *tranalyzer.h* file. Prepending information such as data length, checksum, or an id is controlled by the BUF_DATA_SHFT variable in the Tranalyzer core: *outputBuffer.h*. The user needs to configure the destination port, socket type and whether host info is transmitted in the first record. The socketSink plugin produces output directly into the Ethernet interface.

64.2 Dependencies

64.2.1 External Libraries

If gzip compression is activated (`SKS_GZ_COMPRESS=1`), then **zlib** must be installed.

SKS_GZ_COMPRESS=1		
Ubuntu:	<code>sudo apt-get install zlib1g-dev</code>	
Arch:	<code>sudo pacman -S zlib</code>	
Gentoo:	<code>sudo emerge zlib</code>	
openSUSE:	<code>sudo zypper install zlib-devel</code>	
Red Hat/Fedora⁴³:	<code>sudo dnf install zlib-devel</code>	
macOS⁴⁴:	<code>brew install zlib</code>	

64.2.2 Core Configuration

This plugin requires the following core configuration:

- `$T2HOME/tranalyzer2/src/tranalyzer.h`:
 - `BLOCK_BUF=0`

64.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
<code>SKS_SERVADD</code>	"127.0.0.1"	destination address	
<code>SKS_DPORT</code>	6666	destination port (host order)	
<code>SKS_SOCKTYPE</code>	1	Socket type: 0: UDP; 1: TCP	
<code>SKS_GZ_COMPRESS</code>	0	Compress (gzip) the output	<code>SKS_SOCKTYP=1</code>
<code>SKS_CONTENT_TYPE</code>	1	0: binary; 1: text; 2: JSON	
<code>SKS_HOST_INFO</code>	0	0: no info; 1: all info about host	<code>SKS_CONTENT_TYPE=0</code>

⁴³If the `dnf` command could not be found, try with `yum` instead

⁴⁴Brew is a packet manager for macOS that can be found here: <https://brew.sh>

64.3.1 bin2txt.h

`bin2txt.h` controls the conversion from internal binary format to standard text output.

Variable	Default	Description	Flags
<code>HEX_CAPITAL</code>	0	Hex number representation: 0: lower case, 1: upper case	
<code>IP4_NORMALIZE</code>	0	IPv4 addresses representation: 0: normal, 1: normalized (padded with 0)	
<code>IP6_COMPRESS</code>	1	IPv6 addresses representation: 0: full 128 bit length 1: compressed	
<code>TFS_EXTENDED_HEADER</code>	0	Print an extended header in the flow file (number of rows, columns, columns type)	
<code>B2T_LOCALTIME</code>	0	Time representation: 0: UTC, 1: localtime	
<code>B2T_NANOSECS</code>	0	Time precision: 0: microsecs, 1: nanosecs	
<code>HDR_CHR</code>	"%"	start character of comments in flow file	
<code>SEP_CHR</code>	"\t"	character to use to separate the columns in the flow file	

64.3.2 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- `SKS_SERVADD`
- `SKS_DPORT`

64.4 Additional Output

The output buffer normally being written to the flow file will be directed to the socket.

If `SKS_HOST_INFO=1` then the following header is transmitted as a prelude.

Parameter	Type	Description	Flags
1	U32	Message length	<code>BUF_DATA_SHFT>0</code>
2	U32	Checksum	<code>BUF_DATA_SHFT>1</code>
3	U32	Sensor ID	
4	U64.U32	Present Unix timestamp	
5	RS	OS;Machine Name;built;OS type;HW;	
6	RS	Ethername(address) or IPInterfacename(address/netmask)	

After the prelude all flow based binary buffer will be directed to the socket interface according to the format shown in the following table:

Column	Type	Description	Flags
1	U32	Message length	BUF_DATA_SHFT>0
2	U32	Checksum	BUF_DATA_SHFT>1
3	RU32	Binary buffer output	

64.5 Example

1. Open a socket, e.g., with netcat: nc -l 127.0.0.1 6666
2. Start T2 with the socketSink plugin, e.g., t2 -r file.pcap
3. You should now see the flows on your netcat terminal

To simulate a server collecting data from many T2 or save the transmitted flows into a file, use the following command:
 nc -l 127.0.0.1 6666 > flowfile.txt

64.6 Post-Processing

64.7 t2b2t

The program t2b2t can be used to transform binary Tranalyzer files generated by the [binSink](#) or [socketSink](#) plugin into text or JSON files. The converted files use the same format as the ones generated by the [txtSink](#) or [jsonSink](#) plugin.

The program can be found in \$T2HOME/utils/t2b2t and can be compiled by typing make.

The use of the program is straightforward:

- bin→txt: t2b2t -r FILE_flows.bin -w FILE_flows.txt
- bin→JSON: t2b2t -r FILE_flows.bin -j -w FILE_flows.json
- bin→compressed txt: t2b2t -r FILE_flows.bin -c -w FILE_flows.txt.gz
- bin→compressed JSON: t2b2t -r FILE_flows.bin -c -j -w FILE_flows.json.gz

If the -w option is omitted, the destination is inferred from the input file, e.g., the examples above would produce the same output files with or without the -w option. Note that -w - can be used to output to stdout.

Additionally, the -n option can be used **not** to print the name of the columns as the first row.

Try t2b2t -h for more information.

65 sqliteSink

65.1 Description

The sqliteSink plugin outputs flows to a SQLite database.

65.2 Dependencies

65.2.1 External Libraries

This plugin depends on the **sqlite** library.

Ubuntu:	sudo apt-get install	libssqlite3-dev
Arch:	sudo pacman -S	sqlite
openSUSE:	sudo zypper install	sqlite3-devel
Red Hat/Fedora ⁴⁵ :	sudo dnf install	sqlite-devel
macOS ⁴⁶ :	brew install	sqlite

65.2.2 Core Configuration

This plugin requires the following core configuration:

- \$T2HOME/tralyzer2/src/tralyzer.h:
 - BLOCK_BUF=0

65.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
SQLITE_OVERWRITE	2	0: abort if table already exists 1: overwrite table if it already exists 2: append to table if it already exists
SQLITE_HEX_AS_INT	0	0: store hex numbers (bitfields) as text 1: store hex numbers (bitfields) as int
SQLITE_TRANSACTION_NFLOWS	40000	0: one transaction > 0: one transaction every <i>n</i> flows
SQLITE_QRY_LEN	32768	Initial length for query
SQLITE_QRY_MAXLEN	4194304	Maximal length for query
SQLITE_DB_SUFFIX	".db"	Suffix for the database name
SQLITE_DBNAME	"/tmp/t2.db"	Name of the database
SQLITE_TABLE_NAME	"flow"	Name of the table
T2_SQLITE_SELECT	0	Only insert specific fields into the DB
SQLITE_SELECT_FILE	"sqlite-columns.txt"	Filename of the field selector

⁴⁵If the `dnf` command could not be found, try with `yum` instead

⁴⁶Brew is a packet manager for macOS that can be found here: <https://brew.sh>

Name	Default	Description
(one column name per line)		

65.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- `SQLITE_QRY_LEN`
- `SQLITE_QRY_MAXLEN`
- `SQLITE_DB_SUFFIX`
- `SQLITE_TABLE_NAME`
- `SQLITE_SELECT_FILE`

65.3.2 Database Name

The database name is extracted from Tranalyzer input and/or `-w/-W` option. `SQLITE_DB_SUFFIX` is simply appended. Alternatively, an absolute path may be provided by uncommenting the `SQLITE_DBNAME` macro in `src/sqliteSink.h`.

65.4 Insertion of Selected Fields Only

When `T2_SQLITE_SELECT=1`, the columns to insert into the DB can be customized with the help of `SQLITE_SELECT_FILE`. The filename defaults to `sqlite-columns.txt` in the user plugin folder, e.g., `~/.tranalyzer/plugins`. The format of the file is simply one field name per line with lines starting with a '#' being ignored. For example, to only insert source and destination addresses and ports, create the following file:

```
# Lines starting with a '#' are ignored and can be used to add comments
srcIP
srcPort
dstIP
dstPort
```

65.5 Plugin Report Output

The following information is reported:

- Number of flows discarded due to main buffer problems

65.6 Example

```
# Run Tranalyzer
$ t2 -r file.pcap
# Connect to the SQLite database
$ sqlite3 file.db
# Number of flows
sqlite> select count(*) from flow;
# 10 first srcIP/dstIP pairs
```

```
sqlite> select "srcIP", "dstIP" from flow limit 10;  
# All flows from 1.2.3.4 to 1.2.3.5  
sqlite> select * from flow where "srcIP" = '1.2.3.4' and "dstIP" = '1.2.3.5';
```

66 sshDecode

66.1 Description

This plugin analyzes SSH traffic.

66.2 Dependencies

This plugin requires the **libssl**.

Ubuntu:	sudo apt-get install libssl-dev
Arch:	sudo pacman -S openssl
openSUSE:	sudo zypper install libopenssl-devel
Red Hat/Fedora ⁴⁷ :	sudo dnf install openssl-devel
macOS ⁴⁸ :	brew install openssl@1.1

66.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
SSH_USE_PORT	0	1: Count all packets to/from SSH_PORT as SSH (useful if version exchange was not captured)	
SSH_DECODE	2	0: Do not decode SSH handshake messages 1: Only decode SSH Key Exchange Init messages 2: Decode all SSH Exchange messages	SSH_DECODE>0
SSH_FINGERPRINT	1	Algorithm to use for the fingerprint: 0: No fingerprint, 1: MD5, 1: SHA256	SSH_DECODE=2
SSH_ALGO	1	Output chosen algorithms	SSH_DECODE>0
SSH_LISTS	0	Output lists of supported algorithms	SSH_DECODE>0
SSH_HASSH	1	Output HASSH fingerprint (hash and description)	
SSH_HASSH_STR	0	Also output HASSH fingerprint before hashing	SSH_HASSH=1
SSH_HASSH_DLEN	512	Max length for HASSH descriptions	SSH_HASSH=1
SSH_HASSH_STR_LEN	1024	Max length for uncompressed HASSH signatures	SSH_HASSH=1
SSH_BUF_SIZE	512	Max length for strings	
SSH_HKT_SIZE	48	Max length for host key type and chosen algorithms	
SSH_DEBUG	0	Activate debug output	

In addition, the name of the HASSH database is controlled by the `SSH_HASSH_NAME` flag and defaults to "hassh_fingerprints.tsv".

⁴⁷If the `dnf` command could not be found, try with `yum` instead

⁴⁸Brew is a packet manager for macOS that can be found here: <https://brew.sh>

66.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- `SSH_HASSH_NAME`

66.4 Flow File Output

The sshDecode plugin outputs the following columns:

Column	Type	Description	Flags
<code>sshStat</code>	H16	Status	
<code>sshVersion</code>	R(S)	SSH version and software	
<code>sshHostKeyType</code>	R(SC)	SSH host key type	<code>SSH_DECODE=2</code>
<code>sshFingerprint</code>	R(SC)	SSH public key fingerprint	<code>SSH_DECODE=2 && SSH_FINGERPRINT>0</code>
<code>sshCookie</code>	R(SC)	SSH cookie	<code>SSH_DECODE>0</code>

If `SSH_DECODE>0 && SSH_ALGO=1`, the following columns are displayed:

<code>sshKEX</code>	R(S)	SSH chosen KEX algorithm
<code>sshSrvHKeyAlgo</code>	R(S)	SSH chosen server host key algorithm
<code>sshEncCS</code>	R(S)	SSH chosen encryption algorithm client to server
<code>sshEncSC</code>	R(S)	SSH chosen encryption algorithm server to client
<code>sshMacCS</code>	R(S)	SSH chosen MAC algorithm client to server
<code>sshMacSC</code>	R(S)	SSH chosen MAC algorithm server to client
<code>sshCompCS</code>	R(S)	SSH chosen compression algorithm client to server
<code>sshCompSC</code>	R(S)	SSH chosen compression algorithm server to client
<code>sshLangCS</code>	R(S)	SSH chosen language client to server
<code>sshLangSC</code>	R(S)	SSH chosen language server to client

If `SSH_DECODE>0 && SSH_LISTS=1`, the following columns are displayed:

<code>sshKEXList</code>	R(S)	SSH KEX algorithms
<code>sshSrvHKeyAlgoList</code>	R(S)	SSH server host key algorithms
<code>sshEncCSList</code>	R(S)	SSH encryption algorithms client to server
<code>sshEncSCList</code>	R(S)	SSH encryption algorithms server to client
<code>sshMacCSList</code>	R(S)	SSH MAC algorithms client to server
<code>sshMacSCList</code>	R(S)	SSH MAC algorithms server to client
<code>sshCompCSList</code>	R(S)	SSH compression algorithms client to server
<code>sshCompSCList</code>	R(S)	SSH compression algorithms server to client
<code>sshLangCSList</code>	R(S)	SSH languages client to server
<code>sshLangSCList</code>	R(S)	SSH languages server to client

If `SSH_HASSH=1`, the following columns are displayed:

<code>sshHassh</code>	R(SC)	SSH HASSH fingerprint
<code>sshHasshDesc</code>	R(S)	SSH HASSH description

Column	Type	Description	Flags
sshHasshStr	R(S)	SSH HASSH string	

66.4.1 sshStat

The sshStat column is to be interpreted as follows:

sshStat	Description
2 ⁰ (=0x0001)	Flow contains SSH protocol
2 ¹ (=0x0002)	Keeps track of who sent the SSH banner first
2 ² (=0x0004)	Banner does not end with CRLF or contains NULL byte
2 ³ (=0x0008)	Key Exchange Init message seen
2 ⁴ (=0x0010)	Diffie-Hellman Key Exchange Init message seen
2 ⁵ (=0x0020)	Diffie-Hellman Key Exchange Reply message seen
2 ⁶ (=0x0040)	Elliptic Curve Diffie-Hellman Key Exchange Init message seen
2 ⁷ (=0x0080)	Elliptic Curve Diffie-Hellman Key Exchange Reply message seen
2 ⁸ (=0x0100)	Diffie-Hellman Group Exchange Group message seen
2 ⁹ (=0x0200)	Diffie-Hellman Group Exchange Init message seen
2 ¹⁰ (=0x0400)	Diffie-Hellman Group Exchange Request message seen
2 ¹¹ (=0x0800)	Diffie-Hellman Group Exchange Reply message seen
2 ¹² (=0x1000)	New Keys message seen
2 ¹³ (=0x2000)	String truncated... increase SSH_BUF_SIZE
2 ¹⁴ (=0x4000)	Host key type or chosen algorithm truncated... increase SSH_HKT_SIZE
2 ¹⁵ (=0x8000)	Malformed (decoding error, encrypted, ...)

66.4.2 sshFingerprint

The fingerprint of a public key can be computed as follows:

```
ssh-keygen -lf id_rsa.pub
```

To compute the fingerprint of each host listed in `~/.ssh/known_hosts`, use the following command:

```
ssh-keygen -lf ~/.ssh/known_hosts
```

Note that the default SHA256 algorithm can be changed with the `-E md5` option.

66.5 Packet File Output

In packet mode (`-s` option), the sshDecode plugin outputs the following columns:

Column	Type	Description	Flags
sshStat	H16	Status	

66.6 Monitoring Output

In monitoring mode, the sshDecode plugin outputs the following columns:

Column	Type	Description	Flags
sshNFlows	U64	Number of SSH flows	
sshStat	H16	Status	

66.7 Plugin Report Output

The following information is reported:

- Aggregated [sshStat](#)
- Number of SSH flows
- Number of HASSH signatures matched

67 sslDecode

67.1 Description

This plugin analyzes SSL/TLS and OpenVPN traffic.

67.2 Dependencies

If `SSL_ANALYZE_CERT` is activated, then **libssl** is required.

SSL_ANALYZE_CERT=1		
Ubuntu:	<code>sudo apt-get install libssl-dev</code>	
Arch:	<code>sudo pacman -S openssl</code>	
openSUSE:	<code>sudo zypper install libopenssl-devel</code>	
Red Hat/Fedora ⁴⁹ :	<code>sudo dnf install openssl-devel</code>	
macOS ⁵⁰ :	<code>brew install openssl@1.1</code>	

67.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
<code>SSL_ANALYZE_OVPN</code>	0	Analyze OpenVPN (Experimental)
<code>SSL_ANALYZE_QUIC</code>	0	Analyze TLS 1.3 client/server hello in decrypted QUIC initial packets Requires the quicDecode plugin with <code>QUIC_DECODE_TLS = 1</code>
<code>SSL_REC_VER</code>	1	Output the list and number of record versions
<code>SSL_MAX_REC_VER</code>	3	Maximum number of record versions to store
<code>SSL_HAND_VER</code>	1	Output the list and number of handshake versions
<code>SSL_MAX_HAND_VER</code>	2	Maximum number of handshake versions to store
<code>SSL_EXT_LIST</code>	1	Output the list and number of extensions
<code>SSL_MAX_EXT</code>	20	Maximum number of extensions to store
<code>SSL_SUPP_VER</code>	1	Output the list and number of supported versions
<code>SSL_MAX_SUPP_VER</code>	4	Maximum number of supported versions to store
<code>SSL_SIG_ALG</code>	1	Output the list and number of signature hash algorithms
<code>SSL_MAX_SIG_ALG</code>	15	Maximum number of signature hash algorithms to store
<code>SSL_EC</code>	1	Output the list and number of elliptic curves
<code>SSL_MAX_EC</code>	6	Maximum number of elliptic curves to store
<code>SSL_EC_FORMATS</code>	1	Output the list and number of elliptic curve formats

⁴⁹If the `dnf` command could not be found, try with `yum` instead

⁵⁰Brew is a packet manager for macOS that can be found here: <https://brew.sh>

Name	Default	Description
SSL_MAX_EC_FORMATS	6	Maximum number of elliptic curve formats to store
SSL_ALPN_LIST	1	Output the list and number of protocols (ALPN)
SSL_ALPS_LIST	1	Output the list and number of protocols (ALPS)
SSL_NPN_LIST	1	Output the list and number of protocols (NPN)
SSL_MAX_PROTO	6	Maximum number of protocols (ALPN/ALPS/NPN) to store
SSL_PROTO_LEN	16	Maximum number of characters per protocol (ALPN)
SSL_CIPHER_LIST	1	Output the list and number of supported ciphers
SSL_MAX_CIPHER	3	Maximum number of ciphers to store
SSL_ANALYZE_CERT	1	Analyze certificates
If SSL_ANALYZE_CERT > 0, the following flags are available:		
SSL_CERT_SERIAL	1	Print the certificate serial number
SSL_CERT_FINGERPRINT	1	0: no certificate fingerprint, 1: SHA1, 2: MD5
SSL_CERT_VALIDITY	1	Print certificates validity (Valid from/to, lifetime)
SSL_CERT_SIG_ALG	1	Print the certificate signature algorithm
SSL_CERT_PUBKEY_ALG	1	Print the certificate public key algorithm
SSL_CERT_ALG_NAME_LONG	0	Use short (0) or long (1) names for algorithms
SSL_CERT_PUBKEY_TS	1	Print certificates public key type and size
SSL_CERT_SUBJECT	2	0: no info about cert subject, 1: whole subject as one string, 2: selected fields (see below)
SSL_CERT_ISSUER	2	0: no info about cert issuer, 1: whole issuer as one string, 2: selected fields (see below)
SSL_CERT_COMMON_NAME	1	Print the common name of the issuer/subject
SSL_CERT_ORGANIZATION	1	Print the organization name of the issuer/subject
SSL_CERT_ORG_UNIT	1	Print the organizational unit of the issuer/subject
SSL_CERT_LOCALITY	1	Print the locality name of the issuer/subject
SSL_CERT_STATE	1	Print the state/province name of the issuer/subject
SSL_CERT_COUNTRY	1	Print the country of the issuer/subject (iso3166)
SSL_RM_CERTDIR	1	Remove SSL_CERT_PATH before starting
SSL_SAVE_CERT	0	Save certificates
SSL_CERT_NAME_FINDEX	0	Prepend the flowIndex to the certificate name
SSL_DETECT_TOR	0	Detect likely Tor connections
SSL_BLIST	0	Flag blacklisted certificates
SSL_BLIST_LEN	41	Max length for blacklist descriptions

Name	Default	Description
SSL_JA4	1	Output JA4/JA4S fingerprints
SSL_JA4_O	0	Output JA4/JA4S_o fingerprints (original order)
SSL_JA4_R	0	Output JA4/JA4S_r fingerprints (raw)
SSL_JA4_RO	0	Output JA4/JA4S_ro fingerprints (raw, original order)
SSL_JA4_STR_LEN	254	Max length for uncompressed JA4 signatures (JA4/JA4S_r, JA4/JA4S_ro)
SSL_JA4_DLEN	64	Max length for JA4/JA4S descriptions (sslJA4Desc)
SSL_JA3	1	Output JA3 fingerprints (hash and description)
SSL_JA3_STR	0	Also output JA3 fingerprints before hashing
SSL_JA3_DLEN	64	Max length for JA3 descriptions
SSL_JA3_STR_LEN	1024	Max length for uncompressed JA3 signatures (ja3_str)

If SSL_SAVE_CERT==1 then, certificates are saved under SSL_CERT_PATH (default: "/tmp/TranCerts/") with the extension SSL_CERT_EXT (default: ".pem") and the SHA1 or MD5 fingerprint as filename.

67.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- SSL_RM_CERTDIR
- SSL_CERT_PATH
- SSL_CERT_EXT

67.4 Flow File Output

The sslDecode plugin outputs the following columns:

Column	Type	Description	Flags
sslStat	H32	Status	
sslProto	H32	Protocol	
ovpnType	H16	OpenVPN message types	SSL_ANALYZE_OVPN=1
ovpnSessionID	U64	OpenVPN session ID	SSL_ANALYZE_OVPN=1
sslFlags	H8	SSL flags	
sslVersion	H16	SSL/TLS Version	
sslNumRecVer	U16	Number of record versions	SSL_REC_VER=1
sslRecVer	R(H16)	List of record versions	SSL_REC_VER=1
sslNumHandVer	U16	Number of handshake versions	SSL_HAND_VER=1
sslHandVer	R(H16)	List of handshake versions	SSL_HAND_VER=1
sslVuln	H8	Vulnerabilities	
sslAlert	H64	Alert type	
sslCipher	H16	Preferred (Client)/Negotiated (Server) cipher	
sslNumExt	U16	Number of extensions	SSL_EXT_LIST=1

Column	Type	Description	Flags
sslExtList	R(H16)	List of extensions	SSL_EXT_LIST=1
sslNumSuppVer	U16	Number of supported versions	SSL_SUPP_VER=1
sslSuppVer	R(H16)	List of supported versions (client) (Server: negotiated version)	SSL_SUPP_VER=1
sslNumSigAlg	U16	Number of signature hash algorithms	SSL_SIG_ALG=1
sslSigAlg	R(H16)	List of signature hash algorithms	SSL_SIG_ALG=1
sslNumECPt	U16	Number of elliptic curve points	SSL_EC=1
sslECPt	R(H16)	List of elliptic curve points	SSL_EC=1
sslNumECFormats	U8	Number of EC point formats	SSL_EC_FORMATS=1
sslECFormats	R(H8)	List of EC point formats	SSL_EC_FORMATS=1
sslNumALPN	U16	Number of protocols (ALPN)	SSL_ALPN_LIST=1
sslALPNList	R(S)	List of protocols (ALPN)	SSL_ALPN_LIST=1
sslNumALPS	U16	Number of protocols (ALPS)	SSL_ALPS_LIST=1
sslALPSList	R(S)	List of protocols (ALPS)	SSL_ALPS_LIST=1
sslNumNPN	U16	Number of protocols (NPN)	SSL_NPN_LIST=1
sslNPNList	R(S)	List of protocols (NPN)	SSL_NPN_LIST=1
sslNumCipher	U16	Number of supported ciphers	SSL_CIPHER_LIST=1
sslCipherList	R(H16)	List of supported ciphers	SSL_CIPHER_LIST=1
sslNumCC_	U16_	Number of change_cipher records, A_ Number of alert records, H_ Number of handshake records, AD_ Number of application data records, HB Number of heartbeat records	
sslSessIdLen	U8	Session ID length	
sslGMTTime	R(TS)	GMT Unix Time	
sslServerName	R(S)	server name	

If SSL_ANALYZE_CERT == 1, the following columns are output:

sslCertVersion	R(U8)	Certificate version	SSL_CERT_FINGERPRINT=1
sslCertSerial	R(SC)	Certificate serial number	SSL_CERT_SERIAL=1
sslCertSha1FP	R(SC)	Certificate SHA1 fingerprint	SSL_CERT_FINGERPRINT=1
sslCertMd5FP	R(SC)	Certificate MD5 fingerprint	SSL_CERT_FINGERPRINT=2
sslCNotValidBefore_	TS_	Certificate validity: not valid before,	SSL_CERT_VALIDITY=1
after_	R(TS_	not valid after,	
lifetime	U64)	lifetime	
sslCSigAlg	RS	Certificate signature algorithm	SSL_CERT_SIG_ALG=1
sslCKeyAlg	RS	Certificate public key algorithm	SSL_CERT_PUBKEY_ALG=1
sslCPKeyType_	SC_	Certificate public key type,	SSL_CERT_PUBKEY_TS=1
Size	U16	Certificate public key size (bits)	

If SSL_CERT_SUBJECT > 0, the following columns are output:

sslCSubject	R(S)	Certificate subject	SSL_CERT SUBJECT=1
sslCSubjectCommonName	R(S)	Certificate subject common name	SSL_CERT SUBJECT=2

Column	Type	Description	Flags
sslCSubjectOrgName	R(S)	Certificate subject organization name	SSL_CERT SUBJECT=2
sslCSubjectOrgUnit	R(S)	Certificate subject organizational unit name	SSL_CERT SUBJECT=2
sslCSubjectLocality	R(S)	Certificate subject locality name	SSL_CERT SUBJECT=2
sslCSubjectState	R(S)	Certificate subject state or province name	SSL_CERT SUBJECT=2
sslCSubjectCountry	R(S)	Certificate subject country name	SSL_CERT SUBJECT=2
If SSL_CERT_ISSUER > 0, the following columns are output:			
sslCIssuer	R(S)	Certificate issuer	SSL_CERT_ISSUER=1
sslCIssuerCommonName	R(S)	Certificate issuer common name	SSL_CERT_ISSUER=2
sslCIssuerOrgName	R(S)	Certificate issuer organization name	SSL_CERT_ISSUER=2
sslCIssuerOrgUnit	R(S)	Certificate issuer organizational unit name	SSL_CERT_ISSUER=2
sslCIssuerLocality	R(S)	Certificate issuer locality name	SSL_CERT_ISSUER=2
sslCIssuerState	R(S)	Certificate issuer state or province name	SSL_CERT_ISSUER=2
sslCIssuerCountry	R(S)	Certificate issuer country name	SSL_CERT_ISSUER=2
sslBlistCat	R(S)	Blacklisted certificate category	SSL_BLIST=1&& (SSL_SAVE_CERT=1 SSL_CERT_FINGERPRINT=1)
sslJA3Hash	R(SC)	JA3 fingerprint	SSL_JA3=1
sslJA3Desc	R(S)	JA3 description	SSL_JA3=1
sslJA3Str	R(S)	JA3 string	SSL_JA3=1&& SSL_JA3_STR=1
sslJA4	R(SC)	JA4/JA4S fingerprint	SSL_JA4=1
sslJA4Desc	R(S)	JA4/JA4S description	SSL_JA4=1
sslJA4O	R(SC)	JA4/JA4S_o fingerprint (original order)	SSL_JA4_O=1
sslJA4R	R(SC)	JA4/JA4S_r fingerprint (raw)	SSL_JA4_R=1
sslJA4RO	R(SC)	JA4/JA4S_ro fingerprint (raw, original order)	SSL_JA4_RO=1
sslTorFlow	U8	Tor flow	SSL_DETECT_TOR=1

If SSL_CERT SUBJECT=2 or SSL_CERT_ISSUER=2, then the columns displayed are controlled by the following self-explanatory flags:

- SSL_CERT_COMMON_NAME,
- SSL_CERT_ORGANIZATION,
- SSL_CERT_ORG_UNIT,
- SSL_CERT_LOCALITY,
- SSL_CERT_STATE,
- SSL_CERT_COUNTRY.

67.4.1 sslStat

The hex based status variable `sslStat` is defined as follows:

<code>sslStat</code>	Description
0x0000 0001	Message had mismatched version
0x0000 0002	Record was too long (max 16384)
0x0000 0004	Record was malformed, eg, invalid value
0x0000 0008	Certificate had expired
0x0000 0010	Connection was closed due to fatal alert
0x0000 0020	Connection was renegotiated (existed before)
0x0000 0040	Peer not allowed to send heartbeat requests
0x0000 0080	Cipher list truncated... increase <code>SSL_MAX_CIPHER</code>
0x0000 0100	Extension list truncated... increase <code>SSL_MAX_EXT</code>
0x0000 0200	Protocol (ALPN/NPN/ALPS) list truncated... increase <code>SSL_MAX_PROTO</code>
0x0000 0400	Protocol (ALPN/NPN/ALPS) name truncated... increase <code>SSL_PROTO_LEN</code>
0x0000 0800	EC or EC formats list truncated... increase <code>SSL_MAX_EC</code> or <code>SSL_MAX_EC_FORMATS</code>
0x0000 1000	Certificate is blacklisted
0x0000 2000	Insecure or weak cipher detected (Null, DES, RC4 (RFC7465), ADH, 40/56 bits)
0x0000 4000	Weak protocol detected (SSL 2.0, SSL 3.0)
0x0000 8000	Weak key detected
0x0001 0000	Signature hash algorithms list truncated... increase <code>SSL_MAX_SIG_ALG</code>
0x0002 0000	Supported versions list truncated... increase <code>SSL_MAX_SUPP_VER</code>
0x0004 0000	Packet snapped, decoding failed
0x0008 0000	Failed to compute JA3 fingerprint... increase <code>SSL_JA3_STR_LEN</code>
0x0010 0000	Failed to compute JA4/JA4S fingerprint
0x0020 0000	JA4/JA4S_a successfully computed
0x0040 0000	JA4/JA4S_b successfully computed
0x0080 0000	JA4/JA4S_c successfully computed
0x0100 0000	Insecure cipher (should NEVER be used)
0x0200 0000	Weak cipher (should not be used)
0x0400 0000	Secure cipher
0x0800 0000	Perfect Forward Secrecy (PFS) ciphers
0x1000 0000	JA4/JA4S fingerprint truncated... increase <code>SSL_JA4_STR_LEN</code>
0x2000 0000	Record versions list truncated... increase <code>SSL_MAX_REC_VER</code>
0x4000 0000	Handshake versions list truncated... increase <code>SSL_MAX_HAND_VER</code>
0x8000 0000	—

67.4.2 sslProto

The hex based protocol variable `sslProto` is defined as follows:

sslProto	Description
0x0000 0001	HTTP/0.9, HTTP/1.0 or HTTP/1.1 (ALPN starts with http)
0x0000 0002	HTTP/2 (h2 or h2c)
0x0000 0004	HTTP/3 (h3 or HTTP/0.9/1.1 over QUIC (hq)
0x0000 0008	SPDY/1, SPDY/2 or SPDY/3 (ALPN starts with spdy)
0x0000 0010	IMAP
0x0000 0020	POP3
0x0000 0040	FTP
0x0000 0080	XMPP jabber
0x0000 0100	STUN/TURN
0x0000 0200	Apple Push Notification Service (APNS))
0x0000 0400	WebRTC Media and Data or Confidential WebRTC Media and Data
0x0000 0800	Constrained Application Protocol (CoAP)
0x0000 1000	ManageSieve
0x0000 2000	RTP or RTCP ⁵¹
0x0000 4000	OpenVPN ⁵²
0x0000 8000	OASIS Message Queuing Telemetry Transport (MQTT)
0x0001 0000	acme-tls/1
0x0002 0000	DICOM
0x0004 0000	NNTP (reading) or NNTP (transit)
0x0008 0000	SIP
0x0010 0000	Tabular Data Stream Protocol (TDS)
0x0020 0000	DNS over Dedicated QUIC Connections (DoQ)
0x0040 0000	DNS-over-TLS (DoT)
0x0080 0000	IRC
0x0100 0000	SMB
0x0200 0000	SUNRPC
0x0400 0000	Network Time Security Key Establishment
0x0800 0000	—
0x1000 0000	—
0x2000 0000	—
0x4000 0000	GREASE value
0x8000 0000	Unknown protocol (ALPN matched none of the above)

67.4.3 ovpnType

The ovpnType column is to be interpreted as follows:

⁵¹Guessed by the presence of the use-srtp hello extension

⁵²Guessed by being able to decode the protocol

ovpnType	Description
2^1 (=0x0002)	P_CONTROL_HARD_RESET_CLIENT_V1
2^2 (=0x0004)	P_CONTROL_HARD_RESET_SERVER_V1
2^3 (=0x0008)	P_CONTROL_SOFT_RESET_V1
2^4 (=0x0010)	P_CONTROL_V1
2^5 (=0x0020)	P_ACK_V1
2^6 (=0x0040)	P_DATA_V1
2^7 (=0x0080)	P_CONTROL_HARD_RESET_CLIENT_V2
2^8 (=0x0100)	P_CONTROL_HARD_RESET_SERVER_V2
2^9 (=0x0200)	P_DATA_V2

67.4.4 sslFlags

The `sslFlags` is defined as follows:

sslFlags	Description
0x01	request is SSLv2
0x02	SSLv3 version on ‘request’ layer different than on ‘record’ layer
0x04	gmt_unix_time is small (less than 1 year since epoch, probably seconds since boot)
0x08	gmt_unix_time is more than 5 years in the future (probably random)
0x10	random data (28 bytes) is not random
0x20	compression (deflate) is enabled

67.4.5 sslVersion, sslRecVer, sslHandVer and sslSuppVer

The hex based version variables `sslVersion`, `sslRecVer`, `sslHandVer` and `sslSuppVer` are defined as follows:

sslVersion	Description	sslVersion	Description
0x0200	SSL 2.0	0x7f15	TLS 1.3 (draft 21)
0x0300	SSL 3.0	0x7f16	TLS 1.3 (draft 22)
0x0301	TLS 1.0	0x7f17	TLS 1.3 (draft 23)
0x0302	TLS 1.1	0x7f18	TLS 1.3 (draft 24)
0x0303	TLS 1.2	0x7f19	TLS 1.3 (draft 25)
0x0304	TLS 1.3	0x7f1a	TLS 1.3 (draft 26)
0xa0a	GREASE value	0x7f1b	TLS 1.3 (draft 27)
0x1a1a	GREASE value	0x7f1c	TLS 1.3 (draft 28)
0x2a2a	GREASE value	0x8a8a	GREASE value
0x3a3a	GREASE value	0x9a9a	GREASE value
0x4a4a	GREASE value	0xaaaa	GREASE value
0x5a5a	GREASE value	0xbaba	GREASE value
0x6a6a	GREASE value	0xcaca	GREASE value
0x7a7a	GREASE value	0xdada	GREASE value
0x7f0e	TLS 1.3 (draft 14)	0xeaea	GREASE value
0x7f0f	TLS 1.3 (draft 15)	0xfafa	GREASE value
0x7f10	TLS 1.3 (draft 16)	0xfb17	TLS 1.3 (Facebook draft 23)
0x7f11	TLS 1.3 (draft 17)	0xfb1a	TLS 1.3 (Facebook draft 26)
0x7f12	TLS 1.3 (draft 18)	0xfefc	DTLS 1.3
0x7f13	TLS 1.3 (draft 19)	0xfefd	DTLS 1.2
0x7f14	TLS 1.3 (draft 20)	0xefff	DTLS 1.0

67.4.6 sslVuln

The hex based vulnerability variable `sslVuln` is defined as follows:

sslVuln	Description
0x01	vulnerable to BEAST
0x02	vulnerable to BREACH
0x04	vulnerable to CRIME
0x08	vulnerable to FREAK
0x10	vulnerable to POODLE
0x20	HEARTBLEED attack attempted
0x40	HEARTBLEED attack successful (Not implemented)
0x80	—

67.4.7 sslAlert

The hex based alert variable `sslAlert` is defined as follows (red is fatal):

<code>sslAlert</code>	Description	<code>sslAlert</code>	Description
0x00000000 00000001	close notify	0x00000001 00000000	unknown PSK identity (fatal)
0x00000000 00000002	unexpected message (fatal)	0x00000002 00000000	no application protocol (fatal)
0x00000000 00000004	bad record MAC (fatal)	0x00000004 00000000	—
0x00000000 00000008	decryption failed	0x00000008 00000000	—
0x00000000 00000010	record overflow	0x00000010 00000000	—
0x00000000 00000020	decompression failed (fatal)	0x00000020 00000000	—
0x00000000 00000040	handshake failed (fatal)	0x00000040 00000000	—
0x00000000 00000080	no certificate	0x00000080 00000000	—
0x00000000 00000100	bad certificate	0x00000100 00000000	—
0x00000000 00000200	unsupported certificate	0x00000200 00000000	—
0x00000000 00000400	certificate revoked	0x00000400 00000000	—
0x00000000 00000800	certificate expired	0x00000800 00000000	—
0x00000000 00001000	certificate unknown	0x00001000 00000000	—
0x00000000 00002000	illegal parameter (fatal)	0x00002000 00000000	—
0x00000000 00004000	unknown CA (fatal)	0x00004000 00000000	—
0x00000000 00008000	access denied (fatal)	0x00008000 00000000	—
0x00000000 00010000	decode error (fatal)	0x00010000 00000000	—
0x00000000 00020000	decrypt error	0x00020000 00000000	—
0x00000000 00040000	export restriction (fatal)	0x00040000 00000000	—
0x00000000 00080000	protocol version (fatal)	0x00080000 00000000	—
0x00000000 00100000	insufficient security (fatal)	0x00100000 00000000	—
0x00000000 00200000	internal error (fatal)	0x00200000 00000000	—
0x00000000 00400000	user canceled	0x00400000 00000000	—
0x00000000 00800000	no renegotiation	0x00800000 00000000	—
0x00000000 01000000	unsupported extension	0x01000000 00000000	—
0x00000000 02000000	inappropriate fallback (fatal)	0x02000000 00000000	—
0x00000000 04000000	certificate unobtainable	0x04000000 00000000	—
0x00000000 08000000	unrecognized name	0x08000000 00000000	—
0x00000000 10000000	bad certificate status response	0x10000000 00000000	—
0x00000000 20000000	bad certificate hash value	0x20000000 00000000	—
0x00000000 40000000	unknown PSK identity (fatal)	0x40000000 00000000	—
0x00000000 80000000	no application protocol (fatal)	0x80000000 00000000	Fatal

67.4.8 sslCipher

The `sslCipher` variable represents the preferred cipher for the client and the negotiated cipher for the server. The corresponding name can be found in the `src/sslCipher.h` file. All values following the `0x[0-9a-f]a[0-9a-f]a` pattern are GREASE values.

67.4.9 sslNumCC_A_H_AD_HB

The number of message variable `sslNumCC_A_H_AD_HB` decomposed as follows:

sslNumCC_A_H_AD_HB	Description
sslNumCC	number of change cipher records
sslNumA	number of alerts records
sslNumH	number of handshake records
sslNumAD	number of application data records
sslNumHB	number of heartbeat records

67.4.10 sslExtList

The list of extensions is to be interpreted as follows:

sslExt	Description	sslExt	Description
0x0000	Server name	0x002b	Supported versions
0x0001	Max fragment length	0x002c	Cookie
0x0002	Client certificate URL	0x002d	PSK key exchange modes
0x0003	Trusted CA keys	0x002f	Certificate authorities
0x0004	Truncated HMAC	0x0030	OID filters
0x0005	Status request	0x0031	Post handshake auth
0x0006	User mapping	0x0032	Signature algorithms cert
0x0007	Client authz	0x0033	Key Share
0x0008	Server authz	0x0034	Transparency info
0x0009	Cert type	0x0035	Connection ID (deprecated)
0x000a	Supported groups (elliptic curves)	0x0036	Connection ID
0x000b	EC point formats	0x0037	External ID hash
0x000c	SRP	0x0038	External session ID
0x000d	Signature hash algorithms	0x0039	QUIC transport parameters
0x000e	Use SRTP	0x0040	Ticket request
0x000f	Heartbeat	0x0041	DNSSEC chain
0x0010	ALPN	0x0042	Sequence number encryption algo. (DTLS)
0x0011	Status request v2	0x0043	Return Routability Check (RRC) for DTLS
0x0012	Signed certificate timestamp	0x0a0a	GREASE value
0x0013	Client certificate type	0x1a1a	GREASE value
0x0014	Server certificate type	0x2a2a	GREASE value
0x0015	Padding	0x3374	NPN
0x0016	Encrypt then MAC	0x3377	Origin bound cert
0x0017	Extended master secret	0x337c	Encrypted client cert
0x0018	Token binding	0x3a3a	GREASE value
0x0019	Cached info	0x4469	Application settings (ALPS)
0x001a	TLS LTS	0x4a4a	GREASE value
0x001b	Compress certificate	0x5a5a	GREASE value
0x001c	Record size limit	0x6a6a	GREASE value
0x001d	Pwd protect	0x754f	Channel ID old
0x001e	Pwd clear	0x7550	Channel ID
0x001f	Password salt	0x7a7a	GREASE value
0x0020	Ticket pinning	0x8a8a	GREASE value
0x0021	TLS cert with extern PSK	0x9a9a	GREASE value
0x0022	Delegated credential	0xaaaa	GREASE value
0x0023	Session ticket	0xbaba	GREASE value
0x0024	TLMSP	0xcaca	GREASE value
0x0025	TLMSP proxying	0xdada	GREASE value
0x0026	TLMSP delegate	0xeaea	GREASE value
0x0027	Supported EKT ciphers	0xfafa	GREASE value
0x0028	Extended random	0xfd00	Encrypted Client Hello outer extensions
0x0029	Pre-Shared Key (PSK)	0xfe0d	Encrypted Client Hello
0x002a	Early data	0xff01	Renegotiation info

67.4.11 sslSigAlg

The list of signature hash algorithms is to be interpreted as follows:

sslSigAlg	Description	sslSigAlg	Description
0x0201	rsa_pkcs1_sha1	0xdada	GREASE value
0x0203	ecdsa_sha1	0xeaea	GREASE value
0x0401	rsa_pkcs1_sha256	0xfafa	GREASE value
0x0403	ecdsa_secp256r1_sha256	0xfea0	dilithium2
0x0420	rsa_pkcs1_sha256_legacy	0xfea1	p256_dilithium2
0x0501	rsa_pkcs1_sha384	0xfea2	rsa3072_dilithium2
0x0503	ecdsa_secp384r1_sha384	0xfea3	dilithium3
0x0520	rsa_pkcs1_sha384_legacy	0xfea4	p384_dilithium3
0x0601	rsa_pkcs1_sha512	0xfea5	dilithium5
0x0620	rsa_pkcs1_sha512_legacy	0xfea6	p521_dilithium5
0x0603	ecdsa_secp521r1_sha512	0xfea7	dilithium2_aes
0x0708	sm2sig_sm3	0xfea8	p256_dilithium2_aes
0x0709	gostr34102012_256a	0xfea9	rsa3072_dilithium2_aes
0x070a	gostr34102012_256b	0xfeaa	dilithium3_aes
0x070b	gostr34102012_256c	0xfeab	p384_dilithium3_aes
0x070c	gostr34102012_256d	0xfeac	dilithium5_aes
0x070d	gostr34102012_512a	0xfead	p521_dilithium5_aes
0x070e	gostr34102012_512b	0xfe0b	falcon512
0x070f	gostr34102012_512c	0xfe0c	p256_falcon512
0x0804	rsa_pss_rsae_sha256	0xfe0d	rsa3072_falcon512
0x0805	rsa_pss_rsae_sha384	0xfe0e	falcon1024
0x0806	rsa_pss_rsae_sha512	0xfe0f	p521_falcon1024
0x0807	ed25519	0xfe96	picnic1full
0x0808	ed448	0xfe97	p256_picnic1full
0x0809	rsa_pss_pss_sha256	0xfe98	rsa3072_picnic1full
0x080a	rsa_pss_pss_sha384	0xfe1b	picnic311
0x080b	rsa_pss_pss_sha512	0xfe1c	p256_picnic311
0x081a	ecdsa_brainpoolP256r1tls13_sha256	0xfe1d	rsa3072_picnic311
0x081b	ecdsa_brainpoolP384r1tls13_sha384	0xfe27	rainbowIclassic
0x081c	ecdsa_brainpoolP512r1tls13_sha512	0xfe28	p256_rainbowIclassic
0xa0a	GREASE value	0xfe29	rsa3072_rainbowIclassic
0x1a1a	GREASE value	0xfe3c	rainbowVclassic
0x2a2a	GREASE value	0xfe3d	p521_rainbowVclassic
0x3a3a	GREASE value	0xfe42	sphincsharaka128frobust
0x4a4a	GREASE value	0xfe43	p256_sphincsharaka128frobust
0x5a5a	GREASE value	0xfe44	rsa3072_sphincsharaka128frobust
0x6a6a	GREASE value	0xfe5e	sphincsha256128frobust
0x7a7a	GREASE value	0xfe5f	p256_sphincsha256128frobust
0x8a8a	GREASE value	0xfe60	rsa3072_sphincsha256128frobust
0x9a9a	GREASE value	0xfe7a	sphincsshake256128frobust
0aaaaa	GREASE value	0xfe7b	p256_sphincsshake256128frobust
0xbaba	GREASE value	0xfe7c	rsa3072_sphincsshake256128frobust
0xcaca	GREASE value		

67.4.12 sslCNotValidBefore_after_lifetime

The `sslCNotValidBefore_after_lifetime` indicates the validity period of the certificate, i.e., not valid before / after, and the number of seconds between those two dates.

67.5 Plugin Report Output

The following information is reported:

- Aggregated `sslStat`
- Number of OpenVPN flows (`SSL_ANALYZE_OVPN=1`)
- Number of Tor flows (`SSL_DETECT_TOR=1`)
- Number of SSL 2.0, 3.0
- Number of TLS 1.0, 1.1, 1.2 and 1.3
- Number of DTLS 1.0 (OpenSSL pre 0.9.8f), 1.0 and 1.2 flows.
- Aggregated `sslProto`
- Number of certificates saved (`SSL_SAVE_CERT=1`)
- Number of blacklisted certificates (`SSL_BLIST=1`)
- Number of JA3 signatures matched (`SSL_JA3=1`)
- Number of JA4 signatures matched (`SSL_JA4=1`)
- Number of JA4S signatures matched (`SSL_JA4=1`)

68 stpDecode

68.1 Description

The stpDecode plugin analyzes STP traffic.

68.2 Dependencies

68.2.1 Core Configuration

This plugin requires the following core configuration:

- `$T2HOME/tranalyzer2/src/networkHeaders.h`:
 - `ETH_ACTIVATE>0`

68.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
<code>STP_RTPREXT</code>	1	1: Priority Extension MAC, 0: BID hex	

68.4 Flow File Output

The stpDecode plugin outputs the following columns:

Column	Type	Description	Flags
<code>stpStat</code>	H8	Status	
<code>stpVer</code>	U8	Protocol version identifier	
<code>stpType</code>	H8	Aggregated BPDU types	
<code>stpFlags</code>	H8	Aggregated BPDU flags	
<code>stpRtCst</code>	U32	Root cost	
<code>stpRtBID</code>	H64	Root bridge ID	<code>STP_RTPREXT=0</code>
<code>stpRtPrio</code>	U16	Root priority	<code>STP_RTPREXT=1</code>
<code>stpRtExt</code>	U16	Root extension (VLAN)	<code>STP_RTPREXT=1</code>
<code>stpRtMAC</code>	MAC	Root MAC	<code>STP_RTPREXT=1</code>
<code>stpBrdgID</code>	H64	Bridge ID	<code>STP_RTPREXT=0</code>
<code>stpBrdgPrio</code>	U16	Bridge priority	<code>STP_RTPREXT=1</code>
<code>stpBrdgExt</code>	U16	Bridge extension (VLAN)	<code>STP_RTPREXT=1</code>
<code>stpBrdgMAC</code>	MAC	Bridge MAC	<code>STP_RTPREXT=1</code>
<code>stpFrwrd</code>	U16	Forward delay	

68.4.1 stpStat

The `stpStat` column is to be interpreted as follows:

stpStat	Description
0x01	Flow is STP

68.4.2 stpProto

The stpProto column is to be interpreted as follows:

stpProto	Description
0x0000	Spanning Tree Protocol

68.4.3 stpVer

The stpVer column is to be interpreted as follows:

stpVer	Description
0	Spanning Tree
2	Rapid Spanning Tree
3	Multiple Spanning Tree
4	Shortest Path Tree

68.4.4 stpType

The stpType column is to be interpreted as follows:

stpType	Description
0x00	Configuration BPDU or TCN BPDU
0x02	Rapid/Multiple Spanning Tree
0x80	Topology Change Notification (TCN) BPDU

68.4.5 stpFlags

The stpFlags column is to be interpreted as follows:

stpFlags	Description
2^0 (=0x01)	Topology Change
2^1 (=0x02)	Proposal
2^2 (=0x04)	Port Role: 0x00: Unknown, 0x10: Alternate or Backup, 0x20: Root, 0x30: Designated
2^3 (=0x08)	
2^4 (=0x10)	Learning
2^5 (=0x20)	Forwarding
2^6 (=0x40)	Agreement
2^7 (=0x80)	Topology Change Acknowledgment

68.5 Packet File Output

In packet mode (-s option), the stpDecode plugin outputs the following columns:

Column	Type	Description	Flags
stpStat	H8	Status	
stpProto	H16	Protocol identifier	
stpVer	U8	Protocol version identifier	
stpType	H8	BPDU type	
stpFlags	H8	BPDU flags	
stpRtCst	U32	Root cost	
stpRtBID	H64	Root bridge ID	STP_RTPREXT=0
stpRtPrio	U16	Root priority	STP_RTPREXT=1
stpRtExt	U16	Root extension (VLAN)	STP_RTPREXT=1
stpRtMAC	MAC	Root MAC	STP_RTPREXT=1
stpBrdgID	H64	Bridge BID	STP_RTPREXT=0
stpBrdgPrio	U16	Bridge priority	STP_RTPREXT=1
stpBrdgExt	U16	Bridge extension (VLAN)	STP_RTPREXT=1
stpBrdgMAC	MAC	Bridge MAC	STP_RTPREXT=1
stpPort	H16	Port identifier	
stpMsgAge	U16	Message age	
stpMaxAge	U16	Max age	

Column	Type	Description	Flags
stpHello	U16	Hello time	
stpFrwd	U16	Forward delay	
stpPvstOrigVlan	U16	Originating VLAN (PVSTP+)	

68.6 Monitoring Output

In monitoring mode, the stpDecode plugin outputs the following columns:

Column	Type	Description	Flags
stpPkts	U64	Number of STP packets	

68.7 Plugin Report Output

The following information is reported:

- Aggregated [stpStat](#)
- Aggregated BPDU [stpType](#)
- Aggregated BPDU [stpFlags](#)
- Number of STP packets

69 stunDecode

This plugin analyzes the following protocols:

- Session Traversal Utilities for Nat (STUN)
- Traversal Using Relays around NAT (TURN)
- Interactive Connectivity Establishment (ICE)
- NAT Port Mapping Protocol (NAT-PMP)

69.1 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
NAT_PMP	1	Analyze NAT-PMP

69.2 Flow File Output

The stunDecode plugin outputs the following columns:

Column	Type	Description	Flags
natStat	H32	status	
natErr	H32	error code	
natMCReq_Ind_Succ_Err	U16_U16_U16_U16	number of messages (Req, Ind, Succ, Err)	
natAddr_Port	IP4_U16	mapped address and port	
natXAddr_Port	IP4_U16	(xor) mapped address and port	
natPeerAddr_Port	IP4_U16	peer address and port	
natOrigAddr_Port	IP4_U16	response origin address and port	
natRelayAddr_Port	IP4_U16	relayed address and port	
natDstAddr_Port	IP4_U16	destination address and port	
natOtherAddr_Port	IP4_U16	other address and port	
natLifetime	U32	binding lifetime (seconds)	
natUser	S	username	
natPass	S	password	
natRealm	S	realm	
natSoftware	S	software	

If NAT_PMP=1, the following columns are displayed:

natPMPReqEA_MU_MT	U16_U16_U16	NAT-PMP num. of requests (External Address, Map UDP, Map TCP)
natPMPRespEA_MU_MT	U16_U16_U16	NAT-PMP num. of responses (External Address, Map UDP, Map TCP)
natPMPSSSOE	U32	NAT-PMP seconds since start of epoch

69.2.1 natStat

The natStat column is to be interpreted as follows:

natStat	Description
2 ⁰ (=0x0000 0001)	STUN protocol
2 ¹ (=0x0000 0002)	TURN protocol
2 ² (=0x0000 0004)	ICE protocol
2 ³ (=0x0000 0008)	SIP protocol
2 ⁴ (=0x0000 0010)	Microsoft Extension
2 ⁵ (=0x0000 0020)	Even Port
2 ⁶ (=0x0000 0040)	Reserve next port
2 ⁷ (=0x0000 0080)	Don't fragment
2 ⁸ (=0x0000 0100)	Nonce
2 ¹³ (=0x0000 2000)	Deprecated message attribute
2 ¹⁴ (=0x0000 4000)	STUN over non-standard port
2 ¹⁵ (=0x0000 8000)	malformed message
2 ¹⁶ (=0x0001 0000)	Port Mapping Protocol (PMP)
2 ³¹ (=0x8000 0000)	Packet snapped, analysis incomplete

69.2.2 natErr

The hex based error variable natErr is defined as follows (STUN):

natErr	Description
2 ⁰ (=0x00000001)	Try alt
2 ¹ (=0x00000002)	Bad request
2 ² (=0x00000004)	Unauthorized
2 ³ (=0x00000008)	Forbidden
2 ⁴ (=0x00000010)	Unknown attribute
2 ⁵ (=0x00000020)	Allocation mismatch
2 ⁶ (=0x00000040)	Stale nonce
2 ⁷ (=0x00000080)	Address family not supported
2 ⁸ (=0x00000100)	Wrong credentials
2 ⁹ (=0x00000200)	Unsupported transport protocol
2 ¹⁰ (=0x00000400)	Peer address family mismatch
2 ¹¹ (=0x00000800)	Connection already exists
2 ¹² (=0x00001000)	Connection timeout or failure

natErr	Description
2^{13} (=0x00002000)	Allocation quota reached
2^{14} (=0x00004000)	Role conflict
2^{15} (=0x00008000)	Server error
2^{16} (=0x00010000)	Insufficient capacity
2^{31} (=0x80000000)	Unhandled error

The hex based error variable `natErr` is defined as follows (NAT-PMP):

natErr	Description
2^1 (=0x00000002)	Unsupported version
2^2 (=0x00000004)	Not authorized/refused
2^3 (=0x00000008)	Network failure
2^4 (=0x00000010)	Out of resources
2^5 (=0x00000020)	Unsupported opcode

69.2.3 natMCReq_Ind_Succ_Err

The number of messages variable `natMCReq_Ind_Succ_Err` decomposed as follows:

natMCReq_Ind_Succ_Err	Description
natMCReq	number of requests
natMCInd	number of indications
natMCSucc	number of success response
natMCErr	number of error response

69.3 Plugin Report Output

The following information is reported:

- Aggregated `natStat`
- Aggregated `natErr`
- Number of NAT-PMP packets
- Number of STUN packets

69.4 TODO

Port Control Protocol (PCP)

70 syslogDecode

70.1 Description

The syslogDecode plugin analyzes Syslog traffic.

70.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
SYSLOG_FSN	0	Format for Syslog severity/facility messages: 0: Numbers, 1: Names	

70.3 Flow File Output

The syslogDecode plugin outputs the following columns:

Column	Type	Description	Flags
syslogStat	H8	Status	
syslogMCnt	U32	message count	
syslogSev_Fac_Cnt	RU8_U8_U32	Number of severity/facility messages	

70.3.1 syslogStat

The syslogStat column is to be interpreted as follows:

syslogStat	Description
0x01	Syslog detected
0x02	—
0x04	—
0x08	—
0x10	—
0x20	—
0x40	—
0x80	Counter for facility/severity overflowed

70.4 Packet File Output

In packet mode (-s option), the syslogDecode plugin outputs the following columns:

Column	Type	Description	Flags
syslogStat	H8	Status	
syslogSev	U8/S	Severity	SYSL_FSN=0/1
syslogFac	U8/S	Facility	SYSL_FSN=0/1

70.5 Plugin Report Output

The following information is reported:

- Aggregated [syslogStat](#)
- Number of Syslog packets
- Number of Syslog message types

70.6 TODO

- IPv6 tests

70.7 References

- <https://tools.ietf.org/html/rfc5424>

71 tcpFlags

71.1 Description

The tcpFlags plugin contains IP and TCP header information encountered during the lifetime of a flow.

71.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
IPTOS	0	IPv4 ToS / IPv6 Class representation: 0: IP ToS hex 1: DSCP_ECN decimal 2: Precedence(1-7)_ECN decimal	
RTT_ESTIMATE	1	Estimate Round trip time	
IPCHECKSUM	2	0: No checksums calculation 1: Calculation of L3 (IP) header checksum 2: Calculation of L3 (IP) and L4 (TCP, UDP, ...) checksum	
WINDOWSIZE	1	Output TCP window size parameters	
WINMIN	1	Min. window size threshold defining a healthy communication (only packets below the threshold are counted)	
SEQ_ACK_NUM	1	Output Sequence/Acknowledge number features	
FRAG_ANALYZE	1	Enable fragmentation analysis	
NAT_BT_EST	1	Estimate NAT boot time	
SCAN_DETECTOR	1	Enable scan flow detector	
MPTCP	1	Enable MPTCP dissection	
TCPJA4T	1	Output JA4T/JA4TS fingerprints	
JA4TOPTMX	20	Maximal options stored in flow	TCPJA4T=1
TCPFLGCNT	1	TCP flags counts	
SPKTMD_SEQACKREL	0	SEQ/ACK numbers representation (-s option): 0: absolute 1: relative	SEQ_ACK_NUM=1
SPKTMD_SEQACKHEX	0	SEQ/ACK numbers representation (-s option): 0: uint32_t 1: hex32	SEQ_ACK_NUM=1

71.2.1 WINMIN

WINMIN default 1 setting selects all packets/flow where communication came to a halt due to receiver buffer overflow. Literally the number of window size 0 packets to the sender are then counted. WINMIN can be set to any value defining a healthy communication, which depends on the network and application.

71.3 Flow File Output

The tcpFlags plugin outputs the following columns:

Column	Type	Description	Flags
tcpFStat	H16	Status	
ipMindIPID	U16	IP minimum delta IP ID	
ipMaxdIPID	U16	IP maximum delta IP ID	
ipMinTTL	U8	IP minimum TTL	
ipMaxTTL	U8	IP maximum TTL	
ipTTLChg	U8	IP TTL change count	
ipToS	H8	IP Type of Service (ToS)	IPTOS=0
ipToSDscp_ecn	U8_U8	IP ToS: DSCP and ECN	IPTOS=1
ipToSPrec_ecn	U8_U8	IP ToS: Precedence and ECN	IPTOS=2
ipFlags	H16	IP aggregated flags	
ipOptCnt	U16	IP options count	IPV6_ACTIVATE=0 2
ipOptCpCl_Num	H8_H32	IP aggregated options, copy-class and number	IPV6_ACTIVATE=0 2
ip6OptCntHH_D	U16_U16	IPv6 aggregated Hop-by-Hop dest. option counts	IPV6_ACTIVATE>0
ip6OptHH_D	H32_H32	IPv6 Hop-by-Hop destination options	IPV6_ACTIVATE>0
tcpISeqN	U32	TCP initial sequence number	SEQ_ACK_NUM=1
tcpPSeqCnt	U16	TCP packet sequence count	SEQ_ACK_NUM=1
tcpSeqSntBytes	U64	TCP sent seq diff bytes	SEQ_ACK_NUM=1
tcpSeqFaultCnt	U16	TCP sequence number fault count	SEQ_ACK_NUM=1
tcpPAckCnt	U16	TCP packet ACK count	SEQ_ACK_NUM=1
tcpFlwLssAckRcvdBytes	U64	TCP flawless ACK received bytes	SEQ_ACK_NUM=1
tcpAckFaultCnt	U16	TCP ACK number fault count	SEQ_ACK_NUM=1
tcpBFlgMx	U32	TCP max bytes in flight	SEQ_ACK_NUM=1
tcpInitWinSz	U32	TCP initial effective window size	WINDOWSIZE=1
tcpAvgWinSz	F	TCP average effective window size	WINDOWSIZE=1
tcpMinWinSz	U32	TCP minimum effective window size	WINDOWSIZE=1
tcpMaxWinSz	U32	TCP maximum effective window size	WINDOWSIZE=1
tcpWinSzDwnCnt	U16	TCP effective window size change down count	WINDOWSIZE=1
tcpWinSzUpCnt	U16	TCP effective window size change up count	WINDOWSIZE=1
tcpWinSzChgDirCnt	U16	TCP effective window size direction change count	WINDOWSIZE=1
tcpWinSzThRt	F	TCP packet count ratio below window size WINMIN	WINDOWSIZE=1
tcpFlags	H16	TCP aggregated protocol flags (FIN, SYN, RST, PSH, ACK, URG, ECE, CWR)	
tcpAnomaly	H16	TCP aggregated header anomaly flags	
tcpCntF_S_R_P_A_ _U_E_C_FA_SA_ _RA_N_SF_SFR_ _RF_X	16xU16	TCP flags counts: FIN, SYN, RST, PSH, ACK, URG, ECE, CWR, FIN-ACK, SYN-ACK, RST-ACK, none, SYN-FIN, SYN-FIN-RST RST-FIN, Xmas (FIN-PSH-URG)	TCPFLGCNT=1
tcpJA4T	SC	TCP JA4T/JA4TS fingerprint	TCPJA4T=1

Column	Type	Description	Flags
tcpOptPktCnt	U16	TCP options packet count	
tcpOptCnt	U16	TCP options count	
tcpOptions	H32	TCP aggregated options	
tcpMSS	U16	TCP maximum segment size	
tcpWS	U16	TCP window scale factor	
tcpMPTBF	H16	MPTCP type bitfield	MPTCP=1
tcpMPF	H8	MPTCP flags	MPTCP=1
tcpMPAID	U8	MPTCP address ID	MPTCP=1
tcpMPDSSF	H8	MPTCP DSS flags	MPTCP=1
tcpTmS	U32	TCP time stamp	NAT_BT_EST=1
tcpTmER	U32	TCP time echo reply	NAT_BT_EST=1
tcpEcI	F	TCP estimated counter increment	NAT_BT_EST=1
tcpUtm	D	TCP estimated up time	NAT_BT_EST=1
tcpBtm	TS	TCP estimated boot time	NAT_BT_EST=1
tcpSSASAATrip	F	TCP trip time: A flow: SYN, SYN-ACK B flow: SYN-ACK, ACK	RTT_ESTIMATE=1
tcpRTTAckTripMin	F	TCP ACK trip minimum	RTT_ESTIMATE=1
tcpRTTAckTripMax	F	TCP ACK trip maximum	RTT_ESTIMATE=1
tcpRTTAckTripAvg	F	TCP ACK trip average	RTT_ESTIMATE=1
tcpRTTAckTripJitAvg	F	TCP ACK trip jitter average	RTT_ESTIMATE=1
tcpRTTSseqAA	F	TCP round trip time: A flow: SYN, SYN-ACK, ACK B flow: ACK-ACK	RTT_ESTIMATE=1
tcpRTTAckJitAvg	F	TCP ACK round trip average jitter	RTT_ESTIMATE=1

71.3.1 tcpFStat

The `tcpFStat` column is to be interpreted as follows:

tcpFStat	Description
0x0001	Packet good for inter-distance assessment
0x0002	TCP option init
0x0004	Timestamp option decreasing
0x0008	L4 option field corrupt or not acquired
0x0010	Window state-machine initialized
0x0020	Window update
0x0040	Win 0 probe
0x0080	Win 0 probe ACK
0x0100	Min Window detected

tcpFStat	Description
0x0200	WS used
0x0400	Window Receiver full
0x0800	Window state-machine count up(1)/down(0)
0x1000	L4 Checksum calculation if present
0x2000	UDPLITE Checksum coverage error
0x4000	TCP Selective ACK Option
0x8000	MPTCP detected

71.3.2 ipToS

The ipToS column is to be interpreted as follows:

ipToS	Description
0x01	ECN0: enable ECN when requested by incoming connections, and also request ECN on outgoing connection attempts
0x02	ECN1: (default) enable ECN when requested by incoming connections, but do not request ECN on outgoing connections
0x04	Precedence 0
0x08	Precedence 1
0x10	Precedence 2
0x20	Precedence 3: Class Sel 0
0x40	Precedence 4: Class Sel 1
0x80	Precedence 5: Class Sel 2

71.3.3 ipToS precedence description

The precedence in ipToS, Mode 0–2 is defined by the following table according to RFC 2474:

DSCP	Hex Md 0 Values	Dec Md 1-2 Values	Precedence: Description
CS0	0x00	0	Best Effort
LE	0x01	1	—
CS1	0x20, 0x28, 0x30, 0x38	8, 10, 12, 14	1: Priority
CS2	0x40, 0x48, 0x50, 0x58	16, 18, 20, 22	2: Immediate
CS3	0x60, 0x68, 0x70, 0x78	24, 26, 28, 30	3: Flash - mainly used for voice signaling
CS4	0x80, 0x88, 0x90, 0x98	32, 34, 36, 38	4: Flash Override
CS5	0xa0, 0xb8	40, 46	5: Critical - mainly used for voice RTP
CS6	0xc0	48	6: Internetwork Control
CS7	0xe0	56	7: Network Control

71.3.4 ipFlags

The ipFlags column is to be interpreted as follows:

ipFlags	Description	ipFlags	Description
0x0001	IP options corrupt	0x0100	Fragmentation position error
0x0002	IPv4 packets out of order	0x0200	Fragmentation sequence error
0x0004	IPv4 ID roll over	0x0400	L3 checksum error
0x0008	IP fragment below minimum	0x0800	L4 checksum error
0x0010	IP fragment out of range	0x1000	Length in L3/4 header < actual L3/4 length
0x0020	More Fragment bit	0x2000	Length in UDP/UDP-Lite header ≠ actual UDP/UDP-Lite length
0x0040	IPv4: Don't Fragment bit IPv6: reserve bit	0x4000	Packet inter-distance = 0
0x0080	Reserve bit	0x8000	Packet inter-distance < 0

71.3.5 ipOptCpCl_Num

The aggregated IP options are coded as a bit field in hexadecimal notation where the bit position denotes the IP options type according to following format: [2^{Copy-Class}][2^{Number}]. If the field reads: 0x10_0x00100000 in an ICMP message it is a 0x94 = 148 router alert.

Refer to RFC for decoding the bitfield: <http://www.iana.org/assignments/ip-parameters>.

71.3.6 tcpFlags

The tcpFlags column is to be interpreted as follows:

tcpFlags	Description
2 ⁰ (=0x0001)	FIN: No more data, finish connection
2 ¹ (=0x0002)	SYN: Synchronize sequence numbers
2 ² (=0x0004)	RST: Reset connection
2 ³ (=0x0008)	PSH: Push data
2 ⁴ (=0x0010)	ACK: Acknowledgement field value valid
2 ⁵ (=0x0020)	URG: Urgent pointer valid
2 ⁶ (=0x0040)	ECE: ECN-Echo
2 ⁷ (=0x0080)	CWR: Congestion Window Reduced flag is set
2 ⁸ (=0x0100)	FIN-ACK: Acknowledgment of FIN
2 ⁹ (=0x0200)	SYN-ACK: Acknowledgment of SYN
2 ¹⁰ (=0x0400)	RST-ACK: Acknowledgment of RST
2 ¹¹ (=0x0800)	Potential NULL scan packet or malicious channel
2 ¹² (=0x1000)	SYN-FIN flag: potential scan packet or malicious packet
2 ¹³ (=0x2000)	SYN-FIN-RST flag: potential scan packet or malicious channel
2 ¹⁴ (=0x4000)	FIN-RST flag: abnormal flow termination

tcpFlags	Description
2^{15} (=0x8000)	Potential Xmas scan packet (FIN-PSH-URG) or malicious channel

71.3.7 tcpAnomaly

The `tcpAnomaly` column is to be interpreted as follows:

tcpAnomaly	Description
0x0001	SYN retransmission
0x0002	SEQ retransmission
0x0004	SEQ fast retransmission
0x0008	Duplicate ACK
0x0010	TCP Keep-Alive
0x0020	TCP Keep-Alive ACK
0x0040	Sequence number out-of-order
0x0080	Sequence mess in flow order due to pcap packet loss
0x0100	ACK for unseen packet
0x0200	Previous packet not captured
0x0100	-
0x0200	-
0x1000	Scan detected in flow
0x2000	Successful Scan detected in flow
0x4000	SYN flag with L7 content
0x8000	-

71.3.8 tcpOptions

The `tcpOptions` column is to be interpreted as follows:

tcpOptions	Description
2^0 (=0x00000001)	End of Option List
2^1 (=0x00000002)	No-Operation
2^2 (=0x00000004)	Maximum Segment Size
2^3 (=0x00000008)	Window Scale
2^4 (=0x00000010)	SACK Permitted
2^5 (=0x00000020)	SACK
2^6 (=0x00000040)	Echo (obsoleted by option 8)
2^7 (=0x00000080)	Echo Reply (obsoleted by option 8)
2^8 (=0x00000100)	Timestamps
2^9 (=0x00000200)	Partial Order Connection Permitted (obsolete)
2^{10} (=0x00000400)	Partial Order Service Profile (obsolete)
2^{11} (=0x00000800)	CC (obsolete)
2^{12} (=0x00001000)	CC.NEW (obsolete)

tcpOptions	Description
2^{13} (=0x00002000)	CC.ECHO (obsolete)
2^{14} (=0x00004000)	TCP Alternate Checksum Request (obsolete)
2^{15} (=0x00008000)	TCP Alternate Checksum Data (obsolete)
2^{16} (=0x00010000)	Skeeter
2^{17} (=0x00020000)	Bubba
2^{18} (=0x00040000)	Trailer Checksum Option
2^{19} (=0x00080000)	MD5 Signature Option (obsoleted by option 29)
2^{20} (=0x00100000)	SCPS Capabilities
2^{21} (=0x00200000)	Selective Negative Acknowledgements
2^{22} (=0x00400000)	Record Boundaries
2^{23} (=0x00800000)	Corruption experienced
2^{24} (=0x01000000)	SNAP
2^{25} (=0x02000000)	Unassigned (released 2000-12-18)
2^{26} (=0x04000000)	TCP Compression Filter
2^{27} (=0x08000000)	Quick-Start Response
2^{28} (=0x10000000)	User Timeout Option (also, other known unauthorized use)
2^{29} (=0x20000000)	TCP Authentication Option (TCP-AO)
2^{30} (=0x40000000)	Multipath TCP (MPTCP)
2^{31} (=0x80000000)	all options > 31

71.3.9 tcpMPTBF

The `tcpMPTBF` column is to be interpreted as follows:

tcpMPTBF	Description
2^0 (=0x0001)	TCP_MP_CAPABLE
2^1 (=0x0002)	TCP_MP_JOIN
2^2 (=0x0004)	TCP_MP_DSS
2^3 (=0x0008)	TCP_MP_ADD_ADDR
2^4 (=0x0010)	TCP_MP_Rem_ADDR
2^5 (=0x0020)	TCP_MP_PRIO
2^6 (=0x0040)	TCP_MP_FAIL
2^7 (=0x0080)	TCP_MP_FSTCLS
2^{15} (=0x8000)	TCP_MP_PRIV

71.4 Packet File Output

In packet mode (-s option), the `tcpFlags` plugin outputs the following columns:

Column	Type	Description	Flags
ipToS	H8	IP Type of Service (ToS)	IPTOS=0
ipToSDcsp_ecn	U8_U8	IP ToS: Precedence and ECN	IPTOS=1
ipToSPrec_ecn	H8	IP ToS: DSCP and ECN	IPTOS=2
ipID	U16	IP ID	
ipIDDiff	I32	IP ID diff	
ipFrag	H16	IP fragment	
ipTTL	U8	IP TTL	
ipHdrChkSum	H16	IP header checksum	
ipCalChkSum	H16	IP header computed checksum	
l4HdrChkSum	H16	Layer 4 header checksum	
l4CalChkSum	H16	Layer 4 header computed checksum	
ipFlags	H16	IP flags	
ip6HHOptLen	I16	IPv6 Hop-by-Hop options length	IPV6_ACTIVATE>0
ip6HHOpts	R(H8)	IPv6 Hop-by-Hop options	IPV6_ACTIVATE>0
ip6DOptLen	I16	IPv6 Destination options length	IPV6_ACTIVATE>0
ip6D_OPTS	R(H8)	IPv6 Destination options	IPV6_ACTIVATE>0
ipOptLen	I16	IPv4 options length	IPV6_ACTIVATE=0 2
ipOpts	R(H8)	IPv4 options	IPV6_ACTIVATE=0 2
seq	U32/H32	Sequence number	SEQ_ACK_NUM=1
ack	U32/H32	Acknowledgement number	SEQ_ACK_NUM=1
seqMax	U32/H32	Sequence number max	SEQ_ACK_NUM=1
seqDiff	I32	Sequence number diff	SEQ_ACK_NUM=1
ackDiff	I32	Acknowledgement number diff	SEQ_ACK_NUM=1
seqLen	U32	Sequence length	SEQ_ACK_NUM=1
ackLen	U32	Acknowledgement length	SEQ_ACK_NUM=1
seqFlowLen	I64	Sequence flow length	SEQ_ACK_NUM=1
ackFlowLen	I64	Acknowledgement flow length	SEQ_ACK_NUM=1
tcpMLen	I64	Aggregated valid bytes transmitted so far	SEQ_ACK_NUM=1
tcpBFtgt	U32	Number of bytes in flight (not acknowledge)	SEQ_ACK_NUM=1
tcpFStat	H16	TCP aggregated protocol flags + combinations (CWR, ACK, PSH, RST, SYN, FIN, ...)	SEQ_ACK_NUM=1
tcpFlags	H16	Flags	
tcpAnomaly	H16	TCP aggregated header anomaly flags	
tcpWin	U32	TCP window size	
tcpWS	U16	TCP window scale factor	
tcpMSS	U16	TCP maximum segment size	
tcpTmS	U32	TCP time stamp	NAT_BT_EST=1
tcpTmER	U32	TCP time echo reply	NAT_BT_EST=1
tcpMPTyp	U16	MPTCP type	MPTCP=1
tcpMPF	H8	MPTCP flags	MPTCP=1
tcpMPAID	U8	MPTCP address ID	MPTCP=1
tcpMPDSSF	H8	MPTCP DSS flags	MPTCP=1
tcpOptLen	I32	TCP options length	
tcpOpts	R(H8)	TCP options	

71.5 Monitoring Output

In monitoring mode, the `tcpFlags` plugin outputs the following columns:

Column	Type	Description	Flags
<code>tcpScan</code>	U64	Number of TCP scans attempted	
<code>tcpSuccScan</code>	U64	Number of TCP scans successful	
<code>tcpSynRetries</code>	U64	Number of TCP SYN retries	
<code>tcpSeqRetries</code>	U64	Number of TCP seq retries	

71.6 Plugin Report Output

The following information is reported:

- Aggregated `ipFlags`
- Aggregated `tcpFlags`
- Aggregated `tcpAnomaly`
- Aggregated `ipToS`
- Number of TCP scans attempted, successful
- Number of TCP SYN retries, seq retries
- Number of WinSz below `WINMIN`

71.7 References

- <http://www.iana.org/assignments/ip-parameters>
- <http://www.iana.org/assignments/tcp-parameters/tcp-parameters.xml>

72 tcpStates

72.1 Description

The `tcpStates` plugin tracks the actual state of a TCP connection, by analyzing the flags set in the packet header. The plugin recognizes and reports non-compliant behavior.

72.2 Configuration Flags

None.

72.3 Flow File Output

The `tcpStates` plugin outputs the following columns:

Column	Type	Description	Flags
<code>tcpStatesAFlags</code>	H8	TCP state machine anomalies	

72.3.1 `tcpStatesAFlags`

The `tcpStatesAFlags` column is to be interpreted as follows:

<code>tcpStatesAFlags</code>	Description
0x01	Malformed connection establishment
0x02	Malformed teardown
0x04	Malformed flags during established connection
0x08	Packets detected after teardown
0x10	Packets detected after reset
0x20	—
0x40	Reset from sender
0x80	Potential evil behavior (scan)

72.3.2 Flow Timeouts

The `tcpStates` plugin also changes the timeout values of a flow according to its recognized state:

State	Description	Timeout (seconds)
New	Three way handshake is encountered	120
Established	Connection established	610
Closing	Hosts are about to close the connection	120
Closed	Connection closed	10
Reset	Connection reset encountered by one of hosts	0.1

72.3.3 Differences to the Host TCP State Machines

The plugin state machine (Figure 8) and the state machines usually implemented in hosts differ in some cases. Major differences are caused by the benevolence of the plugin. For example, if a connection has not been established in a correct way, the plugin treats the connection as established, but sets the *malformed connection establishment* flag. The reasons for this benevolence are the following:

- A flow might have been started before invocation of Tranalyzer2.
- A flow did not finish before Tranalyzer2 terminated.
- Tranalyzer2 did not detect every packet of a connection, for example due to a router misconfiguration.
- Flows from malicious programs may show suspicious behavior.
- Packets may be lost **after** being captured by Tranalyzer2 but **before** they reached the opposite host.

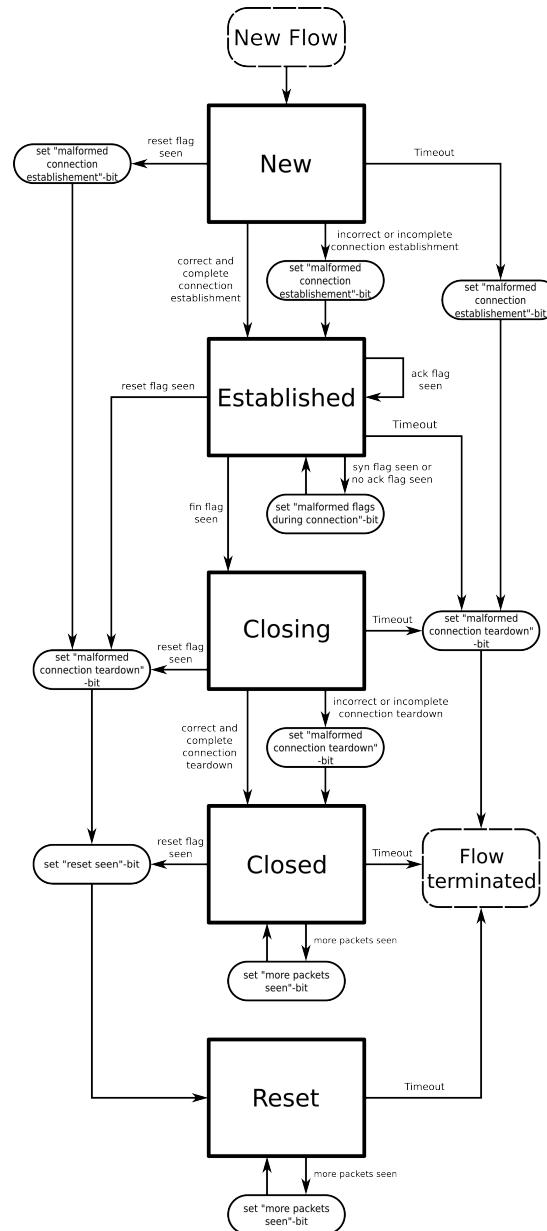
72.4 Packet File Output

In packet mode (-s option), the tcpStates plugin outputs the following columns:

Column	Type	Description	Flags
tcpStatesAFlags	H8	TCP state machine anomalies	

72.5 Plugin Report Output

The aggregated [tcpStatesAFlags](#) anomalies is reported.

**Figure 8:** State machine of the *tcpState* plugin

73 telegram

73.1 Description

The telegram plugin detects Telegram UDP and TCP traffic using heuristics and the IP labeling of basicFlow. Moreover it can extract L7 content of flows for further deobfuscation experiments. It is a rudimentary version, lots to improve.

73.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
TG_SAVE	0	1: save telegram flows
TG_DEOBFUSCATE	0	1: deobfuscate telegram flows
TG_4_9_OR_NEWER	1	1: reduce deobfuscation false positives for Telegram >= 4.9.0 0: needed when deobfuscating traffic generated by Telegram < 4.9.0
TG_DEBUG_MESSAGES	0	1: print debug messages

73.3 Flow File Output

The telegram plugin outputs the following columns:

Column	Type	Description	Flags
<code>tgStat</code>	H16	Status	
<code>tgAuthKeyId</code>	H64	Authentication key ID	TG_DEOBFUSCATE!=0

73.3.1 tgStat

The `tgStat` column is to be interpreted as follows:

tgStat	Description
2^0 (=0x0001)	Telegram detected by heuristics
2^1 (=0x0002)	Control channel
2^1 (=0x0004)	Voice
2^3 (=0x0008)	Telegram detected by IP
2^4 (=0x0010)	File save
2^5 (=0x0020)	Bot app
2^6 (=0x0040)	—
2^7 (=0x0080)	—
2^8 (=0x0100)	Write error
2^9 (=0x0200)	—
2^{10} (=0x0400)	—
2^{11} (=0x0800)	—
2^{12} (=0x1000)	Internal state machine
2^{13} (=0x2000)	Internal state machine

tgStat	Description
2^{14} (=0x4000)	Internal state machine
2^{15} (=0x8000)	Internal state machine init

73.4 Packet File Output

In packet mode (-s option), the telegram plugin outputs the following columns:

Column	Type	Description	Flags
tgStat	H16	Status	

73.5 Plugin Report Output

The following information is reported:

- Aggregated [tgStat](#)
- Number of Telegram packets

73.6 TODO

- Shift IP labeling to on flow generated in [basicFlow](#)

74 telnetDecode

74.1 Description

The telnetDecode plugin analyzes TELNET traffic and is capable to extract L7 content.

74.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
TEL_SAVE	0	Save content to TEL_F_PATH	
TEL_RMDIR	1	Empty TEL_F_PATH before starting	TEL_SAVE=1
TEL_SAVE_SPLIT	1	Save requests (A) and responses (B)	TEL_SAVE=1
TEL_SEQPOS	0	0: no file position control, 1: seq number file position control	TEL_SAVE=1
TEL_CMDOPTS	1	0: Output command/options, 1: Output command/options names	
TEL_CMD_AGGR	1	Aggregate commands	
TEL_OPT_AGGR	1	Aggregate options	
TELCMDN	25	Maximal command / flow	
TELUPLN	25	Maximal length user/password	
TELOPTN	25	Maximal options / flow	
TEL_F_PATH	"/tmp/TELFILES/"	Path for extracted content	

74.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- TEL_RMDIR
- TEL_F_PATH

74.3 Flow File Output

The telnetDecode plugin outputs the following columns:

Column	Type	Description	Flags
telStat	H8	Status	
telCmdBF	H16	Commands	TEL_BTFLD=1
telOptBF	H32	Options	TEL_BTFLD=1
telUsr	SC	Username	
telPW	SC	Password	
telTCCnt	U16	Total command count	
telTOCnt	U16	Total option count	
telCCnt	U16	Stored command count	
telCmdC	R(U8)	Command codes	TEL_CMDOPTS=0

Column	Type	Description	Flags
telCmdS	R(S)	Command names	TEL_CMDOPTS=1
telOCnt	U16	Stored options count	
telOptC	R(U8)	Option codes	TEL_CMDOPTS=0
telOptS	R(S)	Option names	TEL_CMDOPTS=1

74.3.1 telStat

The telStat column is to be interpreted as follows:

telStat	Description	Flags
2 ⁰ (=0x01)	TELNET port found	
2 ¹ (=0x02)	—	
2 ² (=0x04)	Successful username found	
2 ³ (=0x08)	Successful password found	
2 ⁴ (=0x10)	—	
2 ⁵ (=0x20)	File open error	TEL_SAVE=1
2 ⁶ (=0x40)	Command array overflow... increase TELCMDN	
2 ⁷ (=0x80)	Options array overflow... increase TELOPTN	

74.3.2 telCmdBF

The telCmdBF column is to be interpreted as follows:

telCmdBF	Description	telCmdBF	Description
2 ⁰ (=0x0001)	SE - End subNeg	2 ⁸ (=0x0100)	Erase line
2 ¹ (=0x0002)	NOP - No operation	2 ⁹ (=0x0200)	Go ahead!
2 ² (=0x0004)	Data Mark	2 ¹⁰ (=0x0400)	SB - SubNeg
2 ³ (=0x0008)	Break	2 ¹¹ (=0x0800)	WILL use
2 ⁴ (=0x0010)	Int process	2 ¹² (=0x1000)	WON'T use
2 ⁵ (=0x0020)	Abort output	2 ¹³ (=0x2000)	DO use
2 ⁶ (=0x0040)	Are You There?	2 ¹⁴ (=0x4000)	DON'T use
2 ⁷ (=0x0080)	Erase char	2 ¹⁵ (=0x8000)	IAC

74.3.3 telOptBF

The telOptBF column is to be interpreted as follows:

telOptBF	Description	telOptBF	Description
2^0 (=0x00000001)	Bin Xmit	2^{16} (=0x00010000)	Lf Use
2^1 (=0x00000002)	Echo Data	2^{17} (=0x00020000)	Ext ASCII
2^2 (=0x00000004)	Reconn	2^{18} (=0x00040000)	Logout
2^3 (=0x00000008)	Suppr GA	2^{19} (=0x00080000)	Byte Macro
2^4 (=0x00000010)	Msg Sz	2^{20} (=0x00100000)	Data Term
2^5 (=0x00000020)	Opt Stat	2^{21} (=0x00200000)	SUPDUP
2^6 (=0x00000040)	Timing Mark	2^{22} (=0x00400000)	SUPDUP Outp
2^7 (=0x00000080)	R/C XmtEcho	2^{23} (=0x00800000)	Send Locate
2^8 (=0x00000100)	Line Width	2^{24} (=0x01000000)	Term Type
2^9 (=0x00000200)	Page Length	2^{25} (=0x02000000)	End Record
2^{10} (=0x00000400)	CR Use	2^{26} (=0x04000000)	TACACS ID
2^{11} (=0x00000800)	Horiz Tabs	2^{27} (=0x08000000)	Output Mark
2^{12} (=0x00001000)	Hor Tab Use	2^{28} (=0x10000000)	Term Loc
2^{13} (=0x00002000)	FF Use	2^{29} (=0x20000000)	3270 Regime
2^{14} (=0x00004000)	Vert Tabs	2^{30} (=0x40000000)	X.3 PAD
2^{15} (=0x00008000)	Ver Tab Use	2^{31} (=0x80000000)	Window Size

74.3.4 telCmdC and telCmdS

The telCmdC and telCmdS columns are to be interpreted as follows:

telCmdC	telCmdS	Description
0xf0	SE	End of subnegotiation parameters
0xf1	NOP	No Operation
0xf2	DM	Data Mark
0xf3	BRK	Break
0xf4	IP	Interrupt Process
0xf5	AO	Abort Output
0xf6	AYT	Are You There
0xf7	EC	Erase Character
0xf8	EL	Erase Line
0xf9	GA	Go Ahead
0xfa	SB	Subnegotiation
0xfb	WILL	Will Perform
0xfc	WONT	Won't Perform
0xfd	DO	Do Perform
0xfe	DONT	Don't Perform

telCmdC	telCmdS	Description
0xff	IAC	Interpret As Command

74.3.5 telOptC and telOptS

The `telOptC` and `telOptS` columns are to be interpreted as follows:

telOptC	telOptS	Description
0xf0	SE	End of subnegotiation parameters
0xf1	NOP	No Operation
0xf2	DM	Data Mark
0xf3	BRK	Break
0xf4	IP	Interrupt Process
0xf5	AO	Abort Output
0xf6	AYT	Are You There
0xf7	EC	Erase Character
0xf8	EL	Erase Line
0xf9	GA	Go Ahead
0xfa	SB	Subnegotiation
0xfb	WILL	Will Perform
0xfc	WONT	Won't Perform
0xfd	DO	Do Perform
0xfe	DONT	Don't Perform
0xff	IAC	Interpret As Command

74.4 Packet File Output

In packet mode (-s option), the `telnetDecode` plugin outputs the following columns:

Column	Type	Description	Flags
<code>telStat</code>	H8	Status	
<code>telCmdC</code>	U8	Last command code	TEL_CMDOPTS=0
<code>telCmdS</code>	S	Last command name	TEL_CMDOPTS=1
<code>telOptC</code>	U8	Last option code	TEL_CMDOPTS=0
<code>telOptS</code>	S	Last option name	TEL_CMDOPTS=1

74.5 Plugin Report Output

The following information is reported:

- Aggregated `telStat`
- Number of Telnet packets
- Number of files extracted (TEL_SAVE=1)

74.6 TODO

- fragmentation

75 tftpDecode

75.1 Description

The tftpDecode plugin analyzes TFTP traffic. User defined compiler switches are in *tftpDecode.h*.

75.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
TFTP_SAVE	0	Save content to TFTP_F_PATH	
TFTP_RMDIR	1	Empty TFTP_F_PATH before starting	TFTP_SAVE=1
TFTP_CMD_AGGR	1	Aggregate TFTP commands/errors	
TFTP_BTFLD	1	Bitfield coding of TFTP commands	
TFTP_MXNMLN	15	Maximal name length	
TFTP_MAXCNM	2	Maximal length of command field	
TFTP_F_PATH	"/tmp/TFTPFILES/"	Path for extracted content	

75.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- TFTP_RMDIR
- TFTP_F_PATH

75.3 Flow File Output

The tftpDecode plugin outputs the following columns:

Column	Type	Description	Flags
tftpStat	H16	Status	
tftpPFlow	U64	Parent flow	
tftpOpCBF	H8	Opcode bitfield	TFTP_BITFIELD=1
tftpErrCBF	H8	Error code bitfield	TFTP_BITFIELD=1
tftpNumOpcode	U8	Number of opcodes	
tftpOpcode	RSC	Opcodes	TFTP_MAXCNM>0
tftpNumParam	U8	Number of parameters	
tftpParam	RS	Parameters	TFTP_MAXCNM>0
tftpNumError	U8	Number of errors	
tftpErrC	RU16	Error codes	TFTP_MAXCNM>0

75.3.1 tftpStat

The tftpStat column is to be interpreted as follows:

tftpStat	Description	Flags
2^0 (=0x0001)	TFTP flow found	
2^1 (=0x0002)	TFTP data read	
2^2 (=0x0004)	TFTP data write	
2^3 (=0x0008)	File open error	TFTP_SAVE=1
2^4 (=0x0010)	Error in block send sequence	
2^5 (=0x0020)	Error in block ack sequence	
2^6 (=0x0040)	Error or TFTP protocol error or not TFTP	
2^7 (=0x0080)	Array overflow...increase TFTP_MAXCNM	
2^8 (=0x0100)	String truncated...increase TFTP_MXNMLN	
2^9 (=0x0200)	—	
2^{10} (=0x0400)	—	
2^{11} (=0x0800)	Crafted packet or TFTP read/write parameter length error	
2^{12} (=0x1000)	TFTP active	
2^{13} (=0x2000)	TFTP passive	
2^{14} (=0x4000)	—	
2^{15} (=0x8000)	—	

75.3.2 tftpOpcode and tftpOpCBF

The tftpOpCBF column is to be interpreted as follows:

tftpOpCBF	tftpOpcode	Description
2^0 (=0x01)	RRQ	Read request
2^1 (=0x02)	WRQ	Write request
2^2 (=0x04)	DTA	Read or write the next block of data
2^3 (=0x08)	ACK	Acknowledgment
2^4 (=0x10)	ERR	Error message
2^5 (=0x20)	OAK	Option acknowledgment
2^6 (=0x40)	---	—
2^7 (=0x80)	---	—

75.3.3 tftpErrC and tftpErrCBF

The tftpErrCBF column is to be interpreted as follows:

tftpErrC	tftpErrCBF	Description
	0x00	No Error
0	0x01	File not found
1	0x02	Access violation
2	0x04	Disk full or allocation exceeded
3	0x08	Illegal TFTP operation

tftpErrC	tftpErrCBF	Description
4	0x10	Unknown transfer ID
5	0x20	File already exists
6	0x40	No such user
7	0x80	Terminate transfer due to option negotiation

75.4 Packet File Output

In packet mode (-s option), the tftpDecode plugin outputs the following columns:

Column	Type	Description	Flags
tftpOpcode	SC	TFTP opcode	

75.5 Plugin Report Output

The following information is reported:

- Aggregated [tftpStat](#)
- Number of TFTP packets
- Number of files extracted (TFTP_SAVE=1)

76 torDetector

76.1 Description

This plugin detects Tor flows.

76.2 Dependencies

This plugin requires the **libssl**.

Ubuntu:	sudo apt-get install libssl-dev
Arch:	sudo pacman -S openssl
openSUSE:	sudo zypper install libopenssl-devel
Red Hat/Fedora ⁵³ :	sudo dnf install openssl-devel
macOS ⁵⁴ :	brew install openssl@1.1

76.3 Configuration Flags

Name	Default	Description
TOR_DETECT_OBFUSCATION	1	Detect obfuscation protocols
TOR_DEBUG_MESSAGES	0	Activate debug output
TOR_PKTL	1	Activate packet length modulo 8 heuristic

The obfuscation detection method has a pretty high rate of false positives when only one direction of the traffic is captured. It should therefore be disabled (or low confidence should be given to the resulting output) when analyzing Tor traffic which was only captured in one direction.

76.4 Flow File Output

The torDetector plugin outputs the following columns:

Column	Type	Description
torStat	H8	Tor status

76.4.1 torStat

The `torStat` column is to be interpreted as follows:

⁵³If the `dnf` command could not be found, try with `yum` instead

⁵⁴Brew is a packet manager for macOS that can be found here: <https://brew.sh>

torStat	Description
0x01	Tor flow
0x02	Obfuscated Tor flow (TOR_DETECT_OBFUSCATION=1)
0x04	Tor address detected
0x08	Tor pktlen modulo 8 detected
0x10	Internal state: SYN detected
0x20	Internal state: obfuscation checked
0x40	—
0x80	Packet snapped or decoding failed

76.5 Packet File Output

In packet mode (-s option), the torDetector plugin outputs the following columns:

Column	Type	Description	Flags
torStat	H8	Status	

76.6 Plugin Report Output

The following information is reported:

- Aggregated `torStat`
- Number of Tor packets

76.7 Plugin Detection Method

This subsection briefly present the detection methods used by this plugin. Tor version 0.2.7.6 and 0.4.5.10 were used to test these detection methods, they might become invalid in the future.

76.7.1 TLS Certificate

On older versions of the Tor client (< 0.2.9.15) the TLS certificate can be extracted from the traffic. The following conditions are used to determine if it belongs to a Tor flow:

Size of the certificate Tor certificates are very minimalist and are always smaller than 500 bytes.

Public key algorithm Currently Tor uses RSA-1024 certificates. Proposal 220, <https://gitweb.torproject.org/torspec.git/plain/proposals/220-ecc-id-keys.txt>, defines how to support Ed25519 in addition to RSA-1024. This proposal was implemented in Tor 0.2.7.5 according to the changelog: <https://gitweb.torproject.org/tor.git/tree/ReleaseNotes?h=release-0.2.8#n96>. However tests with Tor 0.2.7.6 (client and entry node) showed that RSA-1024 was still used for the TLS link key.

Validity period

before 0.2.4.11 Tor certificate are always valid for exactly one year (60*60*24*365 seconds).

0.2.4.11 and after In March 2013, Tor changed the way it generates the validity period in certificates. Certificates are valid for a random period of time but validity periods always start at exactly midnight.

Certificate issuer The certificate issuer is only defined by its common name (no organization, country, ...). This common name has the following format: `www.RAND.com` where `RAND` is between 8 and 20 (inclusive) base32 characters. If Tor is compiled with the `DISABLE_V3_LINKPROTO_SERVERSIDE` flag, the common name ends in `.net` instead of `.com`.

Certificate subject The certificate subject is only defined by its common name (no organization, country, ...). This common name has the following format: `www.RAND.net` where `RAND` is between 8 and 20 (inclusive) base32 characters.

76.7.2 TLS Client Hello

Recent version of the Tor client ($\geq 0.2.9.15$) use TLS 1.3 which encrypts the certificate. On these versions, only the TLS handshake Client Hello and Server Hello can be used.

Cipher list The `TLS_EMPTY_RENEGOTIATION_INFO_SCSV` is always the last element in the supported cipher list sent by Tor. Recent versions of Firefox and Chrome do not send this cipher.

TLS extensions The `renegotiation_info`, the Application-Layer Protocol Negotiation (ALPN) and the Next Protocol Negotiation (NPN) extensions are never present in Tor Client Hello messages. These extensions are almost always present in modern browsers.

Server name extension The `server_name` extension contains a hostname with the following format: `www.RAND.com` where `RAND` is between 4 and 25 (inclusive) base32 characters.

76.7.3 TLS Server Hello

The following fields are checked in the Server Hello to differentiate Tor from other TLS traffic:

TLS extensions The Application-Layer Protocol Negotiation (ALPN) and the Next Protocol Negotiation (NPN) extensions are never present in Tor Server Hello messages. These extensions are sometimes present in traffic sent by web servers.

76.7.4 Obfuscation detection

Different pluggable transport protocols can be used to obfuscate the Tor traffic between a client and a bridge. For more info: <https://www.torproject.org/docs/pluggable-transports.html.en>.

This plugin tries to detect the `obfs3` and `obfs4` obfuscation methods. The goal of these obfuscation methods is to make the traffic look completely random from the first byte (including the (EC) Diffie-Hellman key exchange). This characteristic is used to detect flows with high entropy in their first packets. Indeed, most encrypted protocols (TLS or SSH for instance) start with an unencrypted handshake phase.

76.8 Other Detection Methods

This subsection describes other possible detection methods not implemented in this plugin.

76.8.1 Relay blacklist

The list of all relays can be queried from the Tor directory. This list contain the relay IP address and OR port. Tor flows can easily be identified by comparing their IP addresses and ports against this list. A CSV list can be downloaded from <https://torstatus.blutmagie.de/>.

The disadvantages of this method are:

- It will not detect Bridge relays (<https://www.torproject.org/docs/bridges>) because it is not possible to query a list of Bridges from the Tor directory. Bridges addresses can only be retrieved three at a time from <https://bridges.torproject.org/> unless the bridge was specifically configured to not publish its descriptor.
- As this list is constantly changing, we need to regularly fetch it and keep all possible versions to use the version with the date closest to the analyzed PCAP.

The advantage of this method is:

- The traffic payload is not necessary, this method only needs the flow ports and IP addresses.

76.8.2 Packet size distribution

Tor always pack data in cells of 512 bytes. This means that when a client or a relay only has a few bytes to transmit, a packet with a size a bit superior to 512 bytes (because of TLS overhead) will be sent. This results in a peak in the packet size distribution around 550 bytes (543 bytes in the tests done with Tor version 0.2.7.6) and very few packets with smaller sizes. Figures 9 and 10 show the difference in the packet size distribution between HTTPS traffic and Tor traffic.

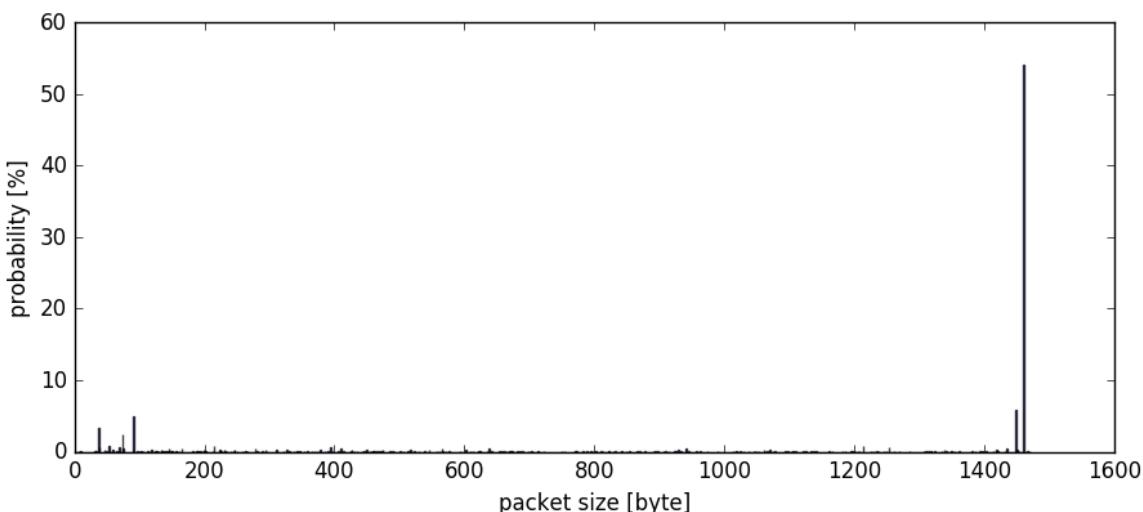


Figure 9: Packet size distribution of HTTPS traffic.

The disadvantage of this method is:

- False positives and negatives are way more frequent than with certificate analysis.

The advantage of this method is:

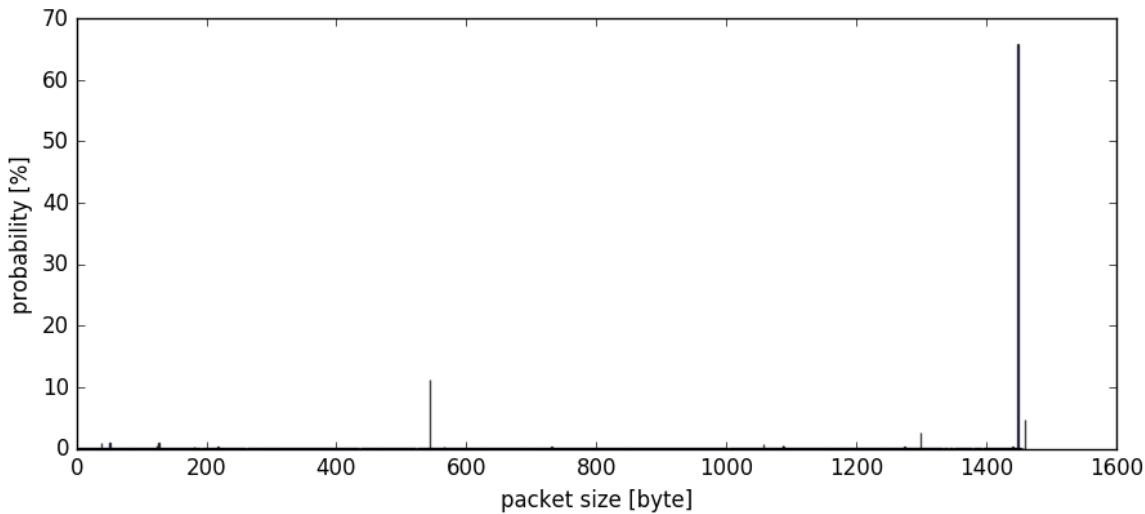


Figure 10: Packet size distribution of Tor traffic.

- The traffic payload is not necessary, this method only needs the packet size and the 5 (or 6 with VLAN) tuple necessary to aggregate packets in flows.

77 tp0f

77.1 Description

The tp0f plugin classifies IP addresses according to OS type and version. It uses initial TTL and window size and can also use the rules from p0f. In order to label non-TCP flows, the plugin can store a hash of already classified IP addresses.

77.1.1 Required Files

If TP0FRULES=1, then the file tp0fL34.txt is required.

77.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
TP0FRULES	1	0: Standard OS guessing 1: OS guessing and p0f L3/4 rules
TP0FHSH	1	0: No IP hash 1: IP hash to recognize IP already classified
TP0FRC	0	0: Only human readable 1: p0f rule and classifier numbers
TP0F_L34FILE	"tp0fL34.txt"	File containing converted L3/4 rules

In tp0flist.h:

Name	Default	Description
TCPOPTMAX	40	Maximal TCP option codes to store and process

77.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- TP0F_L34FILE

77.3 Flow File Output

The p0f plugin outputs the following columns:

Column	Type	Description	Flags
tp0fStat	H8	Status	
tp0fDis	U8	Initial TTL distance	
tp0fRN	U16	Rule number that triggered	TP0FRC=1
tp0fClass	U8	OS class of rule file	TP0FRC=1
tp0fProg	U8	Program category of rule file	TP0FRC=1
tp0fVer	U8	Version category of rule file	TP0FRC=1

Column	Type	Description	Flags
tp0fC1Name	SC	OS class name	
tp0fPrName	SC	OS/program name	
tp0fVerName	SC	OS/program version name	

77.3.1 tp0fStat

The tp0fStat column is to be interpreted as follows:

tp0fStat	Description
0x01	SYN tp0f rule fired
0x02	SYN-ACK tp0f rule fired
0x04	—
0x08	—
0x10	—
0x20	—
0x40	IP already seen by tp0f
0x80	TCP option length or content corrupt

77.4 Plugin Report Output

The number of packets which fired a tp0f rule is reported.

77.5 TODO

- Integrate TLS rules
- Integrate HTTP rules

77.6 References

- <http://www.netresec.com/?page=Blog&month=2011-11&post=Passive-OS-Fingerprinting>
- <http://lcamtuf.coredump.cx/p0f3/>

78 txtSink

78.1 Description

The txtSink plugin provides human readable text output which can be saved in a file `PREFIX_flows.txt`, where `PREFIX` is provided via the `-w` option. The generated output contains a textual representation of all plugins results. Each line in the file represents one flow. The different output statistics of the plugins are separated by a tab character to provide better post-processing with command line scripts or statistical toolsets.

78.2 Dependencies

78.2.1 External Libraries

If gzip compression is activated (`TFS_GZ_COMPRESS=1`), then **zlib** must be installed.

TFS_GZ_COMPRESS=1		
Ubuntu:	<code>sudo apt-get install zlib1g-dev</code>	
Arch:	<code>sudo pacman -S zlib</code>	
Gentoo:	<code>sudo emerge zlib</code>	
openSUSE:	<code>sudo zypper install zlib-devel</code>	
Red Hat/Fedora⁵⁵:	<code>sudo dnf install zlib-devel</code>	
macOS⁵⁶:	<code>brew install zlib</code>	

78.2.2 Core Configuration

This plugin requires the following core configuration:

- `$T2HOME/tralyzer2/src/tralyzer.h`:
 - `BLOCK_BUF=0`

78.3 Configuration Flags

The configuration flags for the txtSink plugins are separated in two files.

78.3.1 txtSink.h

Name	Default	Description
<code>TFS_SPLIT</code>	1	Split the output file (Tralyzer <code>-W</code> option)
<code>TFS_PRI_HDR</code>	1	Print a row with column names at the start of the flow file
<code>TFS_HDR_FILE</code>	1	Generate a separate header file (Section 78.4.1)
<code>TFS_PRI_HDR_FW</code>	0	Print header in every output fragment (Tralyzer <code>-W</code> option)
<code>TFS_GZ_COMPRESS</code>	0	Compress the output (gzip)
<code>TFS_FLOWS_TXT_SUFFIX</code>	<code>"_flows.txt"</code>	Suffix for the flow file

⁵⁵If the `dnf` command could not be found, try with `yum` instead

⁵⁶Brew is a packet manager for macOS that can be found here: <https://brew.sh>

Name	Default	Description
TFS_HEADER_SUFFIX	"_headers.txt"	Suffix for the header file

78.3.2 bin2txt.h

bin2txt.h controls the conversion from internal binary format to standard text output.

Name	Default	Description
IP4_FORMAT	0	IPv4 addresses representation: 0: normal, 1: normalized (padded with zeros), 2: one 32-bits hex number 3: one 32-bits unsigned number
IP6_FORMAT	0	IPv6 addresses representation: 0: compressed, 1: uncompressed, 2: one 128-bits hex number, 3: two 64-bits hex numbers
MAC_FORMAT	0	MAC addresses representation: 0: normal (edit MAC_SEP to change the separator), 1: one 64-bits hex number,
MAC_SEP	:	Separator to use in MAC addresses: 11:22:33:44:55:66
B2T_NON_IP_STR	"_"	Representation of non-IPv4/IPv6 addresses in IP columns
HEX_CAPITAL	0	Hex output: 0: lower case; 1: upper case
TFS_EXTENDED_HEADER	0	Extended header in flow file
B2T_NANOSECS	0	Time precision: 0: microsecs, 1: nanosecs
TFS_NC_TYPE	2	Types in header file: 0: none, 1: numbers, 2: C types
TFS_SAN_UTF8	1	Activates the UTF-8 sanitizer for strings
B2T_TIMESTR	0	Print unix timestamps as human readable dates
HDR_CHR	"%"	start character(s) of comments
SEP_CHR	"\t"	column separator in the flow file ";", ".", "_" and "\\" should not be used

78.3.3 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- TFS_FLOWS_TXT_SUFFIX
- TFS_HEADER_SUFFIX

78.4 Additional Output

78.4.1 Header File

The header file `PREFIX_headers.txt` describes the columns of the flow file and provides some additional information, such as plugins loaded and PCAP file or interface used, as depicted below. The default suffix used for the header file is `_headers.txt`. This suffix can be configured using `TFS_HEADER_SUFFIX`.

```

# Date: 1566316839.259591 sec (Tue 5 Aug 2023 18:00:39 CEST)
# Tranalyzer 0.9.0 (Anteater), Cobra.
# Core configuration: L2, IPv4, IPv6
# SensorID: 666
# PID: 13221
# Command line: /home/user/tralyzer2-0.9.0/tralyzer2/src/tralyzer -r file.pcap
# HW Info: hostname;sysname;release;version;machine
# SW info: libpcap version 1.9.1
#
# Plugins loaded:
#   01: protoStats, version 0.9.0
#   02: basicFlow, version 0.9.0
#   03: macRecorder, version 0.9.0
#   04: portClassifier, version 0.9.0
#   05: basicStats, version 0.9.0
#   06: tcpFlags, version 0.9.0
#   07: tcpStates, version 0.9.0
#   08: icmpDecode, version 0.9.0
#   09: connStat, version 0.9.0
#  10: txtSink, version 0.9.0
#
# Col No.    Type          Name           Description
1          C              dir            Flow direction
2          U64            flowInd        Flow index
3          H64            flowStat       Flow status and warnings
4          U64.U32        timeFirst     Date time of first packet
5          U64.U32        timeLast      Date time of last packet
6          U64.U32        duration      Flow duration
7          U8             numHdrDesc  Number of different headers descriptions
8          U16:R          numHdtrs     Number of headers (depth) in hdrDesc
9          SC:R           hdrDesc       Headers description
10         MAC:R          srcMac        Mac source
11         MAC:R          dstMac        Mac destination
12         H16            ethType      Ethernet type
13         U16:R          vlanID       VLAN IDs
14         IPX            srcIP         Source IP address
15         SC              srcIPCC      Source IP country
16         S               srcIPOrg    Source IP organization
17         U16            srcPort      Source port
18         IPX            dstIP         Destination IP address
19         SC              dstIPCC      Destination IP country
20         S               dstIPOrg    Destination IP organization
21         U16            dstPort      Destination port
22         U8             l4Proto      Layer 4 protocol
23         H8             macStat      macRecorder status
...

```

The column number can be used, e.g., with awk or [tawk](#) to query a given column. For example, to extract all ICMP flows (layer 4 protocol equals 1) from a flow file:

```
awk -F'\t' '$22 == 1' PREFIX_flows.txt
```

The second column indicates the type of the column (see table below). If the value is repetitive, the type is postfixed with :R. Repetitive values can occur any number of times (from 0 to N). Each repetition is separated by a semicolon. The '_' indicates a compound, i.e., a value containing 2 or more subvalues.

#	Name	Description	#	Name	Description	#	Name	Description
1	I8	int8	11	U128	uint128	21	LD	long double
2	I16	int16	12	U256	uint256	22	C	char
3	I32	int32	13	H8	hex8	23	S	string
4	I64	int64	14	H16	hex16	24	C	flow direction ⁵⁷
5	I128	int128	15	H32	hex32	25	TS	timestamp ⁵⁸
6	I256	int256	16	H64	hex64	26	U64.U32	duration
7	U8	uint8	17	H128	hex128	27	MAC	mac address
8	U16	uint16	18	H256	hex256	29	IP4	IPv4 address
9	U32	uint32	19	F	float	29	IP6	IPv6 address
10	U64	uint64	20	D	double	30	IPX	IPv4 or 6 address
						31	SC	string class ⁵⁹

⁵⁷A: client→server, B: server→client⁵⁸U64.U32/S (See B2T_TIMESTR in [bin2txt.h](#))⁵⁹string without quotes

79 voipDetector

79.1 Description

The idea of this plugin is to identify SIP, RTP and RTCP flows independently of each other, so that also non standard traffic can be detected. Moreover certain QoS values are extracted.

79.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
VOIP_SIP	2	0: do not decode SIP, 1: Enable SIP decoder, 2: Decode SIP and add RTP/SIP findex/ssrc flow correlation	
VOIP_SIP_PRV	0	0: No RTP/SIP flow correlation enhancement, 1: add RTP srcIP, 2: add srcIP of SIP flow	VOIP_SIP=2 VOIP_SIP=2 VOIP_SIP=2
VOIP_RTP	1	Enable RTP decoder	
VOIP_RTCP	1	Enable RTCP decoder	
VOIP_ANALEN	0	0: only ssrc check, 1: additional check report len against payload length	
VOIP_SAVE	0	Save RTP content to <code>VOIP_V_PATH</code>	
VOIP_BUFSIZE	1	Enable buffering of saved RTP content	VOIP_SAVE=1
VOIP_SILREST	1	Restore back G.711 suppressed silences	VOIP_SAVE=1
VOIP_PLDOFF	0	Offset for payload to save	VOIP_SAVE=1
VOIP_SVFDX	1	Merge ops: 1: findex, 0: SSRC	VOIP_SAVE=1
VOIP_MINPKT	1	Minimum packet length of a flow	VOIP_SAVE=1
RTPFMAX	10	Maximal SSRC files	VOIP_SAVE=1&& VOIP_SVFDX=0
SIPNMMAX	35	Maximal SIP caller name length	
SIPSTATMAX	8	Maximal SIP state requests	
SIPCLMAX	3	Maximal SIP state requests name length	
SIPRFXMAX	100	Maximal SIP IP, m=audio / video ports	
RTPBUFSIZE	4096	Size of buffer for RTP content	VOIP_SAVE=1
RTPMAXVERS	1	Maximal number of version violations	
VOIP_RMDIR	1	Empty <code>VOIP_V_PATH</code> before starting	VOIP_SAVE=1
VOIP_PERM	<code>S_IRWXU</code>	File permissions	VOIP_SAVE=1
VOIP_V_PATH	<code>"/tmp/TranVoIP"</code>	Path for extracted content	VOIP_SAVE=1
VOIP_FNAME	<code>"nudel"</code>	Default content file name prefix	VOIP_SAVE=1

79.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- `VOIP_RMDIR`
- `VOIP_V_PATH`

- VOIP_FNAME

79.3 Flow File Output

The voipDetector plugin outputs the following columns:

Column	Type	Description	Flags
voipStat	H16	Status	
voipType	R(U8)	RTP/RTCP type	
voipSSRC	R(H32)	RTP/RTCP Synchronization Source Identifier	
voipCSRC	R(H32)	RTP/RTCP Contributing Sources	
voipSRCnt	U8	RTP SID / RTCP record count	
rtpPMCnt	U32	RTP packet miss count	
rtpPMr	F	RTP packet miss ratio	
<hr/>			
sipMethods	H16	SIP methods	VOIP_SIP>0
sipStatCnt	U8	SIP stat count	VOIP_SIP>0
sipReqCnt	U8	SIP request count	VOIP_SIP>0
sipUsrAgnt	S	SIP User-Agent	VOIP_SIP>0
sipRealIP	S	SIP X-Real-IP	VOIP_SIP>0
sipFrom	R(S)	SIP Caller	VOIP_SIP>0
sipTo	R(S)	SIP Callee	VOIP_SIP>0
sipCallID	R(S)	SIP Call-ID	VOIP_SIP>0
sipContact	R(S)	SIP Contact	VOIP_SIP>0
sipStat	R(U16)	SIP stat	VOIP_SIP>0
sipReq	R(SC)	SIP request	VOIP_SIP>0
<hr/>			
sdpSessID	R(S)	SDP session ID	VOIP_SIP>0
sdpRFAdd	R(IP)	SDP RTP audio/video flow address	VOIP_SIP>0
sdpRAFPrt	R(U16)	SDP RTP audio flow port	VOIP_SIP>0
sdpRVFPrt	R(U16)	SDP RTP video flow port	VOIP_SIP>0
sdpRTPMap	R(SC)	SDP rtpmap	VOIP_SIP>0
voipFindex	R(U64)	SIP RTP findex	VOIP_SIP>1
<hr/>			
rtcpTPCnt	U32	RTCP cumulated transmitter packet count	VOIP_RTCP=1
rtcpTBCnt	U32	RTCP cumulated transmitter byte count	VOIP_RTCP=1
rtcpFracLst	U8	RTCP cumulated fraction lost	VOIP_RTCP=1
rtcpCPMCnt	U32	RTCP cumulated packet miss count	VOIP_RTCP=1
rtcpMaxIAT	U32	RTCP max inter-arrival time	VOIP_RTCP=1
<hr/>			
voipFname	S	RTP content filename	VOIP_SAVE=1

79.3.1 voipStat

The voipStat column is to be interpreted as follows:

voipStat	Description
2^0 (=0x0001)	RTP detected
2^1 (=0x0002)	RTCP detected
2^2 (=0x0004)	SIP detected
2^3 (=0x0008)	STUN detected
2^4 (=0x0010)	RTP extension header
2^5 (=0x0020)	RT(C)P padding bytes
2^6 (=0x0040)	SDP detected
2^7 (=0x0080)	RTP marker
2^8 (=0x0100)	RTP content write operation
2^9 (=0x0200)	SIP audio RTP flow announced
2^{10} (=0x0400)	SIP video RTP flow announced
2^{11} (=0x0800)	sdpRFAAdd field truncated...increase SIPRFMAX
2^{12} (=0x1000)	RTP packet loss detected
2^{13} (=0x2000)	RTP sequence number jump to past
2^{14} (=0x4000)	RTP new frame header flag
2^{15} (=0x8000)	RTP error in detection

79.3.2 sipMethods

The `sipMethods` column is to be interpreted as follows:

sipMethods	Description
2^0 (=0x0001)	Unknown method
2^1 (=0x0002)	INVITE
2^2 (=0x0004)	ACK
2^3 (=0x0008)	BYE
2^4 (=0x0010)	CANCEL
2^5 (=0x0020)	REGISTER
2^6 (=0x0040)	OPTIONS
2^7 (=0x0080)	PRACK
2^8 (=0x0100)	SUBSCRIBE
2^9 (=0x0200)	NOTIFY
2^{10} (=0x0400)	PUBLISH
2^{11} (=0x0800)	INFO
2^{12} (=0x1000)	REFER
2^{13} (=0x2000)	MESSAGE
2^{14} (=0x4000)	UPDATE
2^{15} (=0x8000)	—

79.4 Packet File Output

In packet mode (-s option), the voipDetector plugin outputs the following columns:

Column	Type	Description	Flags
voipStat	H16	Status	
voipType	R(U8)	RTP / RTCP type	
voipSeqN	U8	RTP / RTCP sequence number	
voipTs	U32	RTP / RTCP timestamp	
voipTsDiff	I32	RTP / RTCP timestamp difference	
voipSSRC	H32	RTP / RTCP ID	

79.5 Monitoring Output

In monitoring mode, the voipDetector plugin outputs the following columns:

Column	Type	Description	Flags
voipFHndl	U64	Number of file handles	VOIP_SAVE=1
sipPkts	U64	Number of SIP packets	VOIP_SIP>0
sipUnkPkts	U64	Number of SIP UNKNOWN packets	VOIP_SIP>0
sipInvPkts	U64	Number of SIP INVITE packets	VOIP_SIP>0
sipAckPkts	U64	Number of SIP ACK packets	VOIP_SIP>0
sipByePkts	U64	Number of SIP BYE packets	VOIP_SIP>0
sipCanPkts	U64	Number of SIP CANCEL packets	VOIP_SIP>0
sipRegPkts	U64	Number of SIP REGISTER packets	VOIP_SIP>0
sipOptPkts	U64	Number of SIP OPTIONS packets	VOIP_SIP>0
sipPraPkts	U64	Number of SIP PRACK packets	VOIP_SIP>0
sipSubPkts	U64	Number of SIP SUBSCRIBE packets	VOIP_SIP>0
sipNotPkts	U64	Number of SIP NOTIFY packets	VOIP_SIP>0
sipPubPkts	U64	Number of SIP PUBLISH packets	VOIP_SIP>0
sipInfPkts	U64	Number of SIP INFO packets	VOIP_SIP>0
sipRefPkts	U64	Number of SIP REFER packets	VOIP_SIP>0
sipMsgPkts	U64	Number of SIP MESSAGE packets	VOIP_SIP>0
sipUpdPkts	U64	Number of SIP UPDATE packets	VOIP_SIP>0
sdpPkts	U64	Number of SDP packets	VOIP_SIP>0
sipAPCntr	U64	Number of unique SDP audio address, port	VOIP_SIP>0
sipFdxMtch	U64	Number of SIP/RTP matches	VOIP_SIP>0
rtpPkts	U64	Number of RTP packets	VOIP_RTP=1
rtcpPkts	U64	Number of RTCP packets	VOIP_RTCP=1

79.6 Plugin Report Output

The following information is reported:

- Aggregated [voipStat](#)
- Aggregated [sipMethods](#)

- Number of SIP packets (VOIP_SIP=1)
- Number of SIP UNKNOWN packets (VOIP_SIP=1)
- Number of SIP INVITE packets (VOIP_SIP=1)
- Number of SIP ACK packets (VOIP_SIP=1)
- Number of SIP BYE packets (VOIP_SIP=1)
- Number of SIP CANCEL packets (VOIP_SIP=1)
- Number of SIP REGISTER packets (VOIP_SIP=1)
- Number of SIP OPTIONS packets (VOIP_SIP=1)
- Number of SIP PRACK packets (VOIP_SIP=1)
- Number of SIP SUBSCRIBE packets (VOIP_SIP=1)
- Number of SIP NOTIFY packets (VOIP_SIP=1)
- Number of SIP PUBLISH packets (VOIP_SIP=1)
- Number of SIP INFO packets (VOIP_SIP=1)
- Number of SIP REFER packets (VOIP_SIP=1)
- Number of SIP MESSAGE packets (VOIP_SIP=1)
- Number of SIP UPDATE packets (VOIP_SIP=1)
- Number of SDP packets (VOIP_SIP=1)
- Number of unique SDP audio address, port (VOIP_SIP=1)
- Number of unique SIP/RTP flow matches (VOIP_SIP=1)
- Number of RTP packets (VOIP_RTP=1)
- Number of RTCP packets (VOIP_RTCP=1)
- Max number of file handles (VOIP_SAVE=1)

79.7 TODO

- Skype
- Google Talk

80 vrrpDecode

80.1 Description

The vrrpDecode plugin analyzes Virtual Router Redundancy Protocol (VRRP) traffic.

80.2 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
VRRP_NUM_VRID	5	number of unique virtual router ID to store	
VRRP_NUM_IP	25	number of unique IPs to store	
VRRP_RT	1	output routing tables	
VRRP_SUFFIX	"_vrrp.txt"	Suffix for routing tables file	VRRP_RT=1

80.2.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- VRRP_SUFFIX

80.3 Flow File Output

The vrrpDecode plugin outputs the following columns:

Column	Type	Description
vrrpStat	H16	Status
vrrpVer	H8	Version
vrrpType	H8	Type
vrrpVRIDCnt	U32	Virtual router ID count
vrrpVRID	R(U8)	Virtual router ID
vrrpMinPri	U8	Minimum priority
vrrpMaxPri	U8	Maximum priority
vrrpMinAdvInt	U8	Minimum advertisement interval (seconds)
vrrpMaxAdvInt	U8	Maximum advertisement interval (seconds)
vrrpAuthType	H8	Authentication type
vrrpAuth	SC	Authentication string
vrrpIPCnt	U32	IP address count
vrrpIP	R(IP)	IP addresses

80.3.1 vrrpStat

The vrrpStat column is to be interpreted as follows:

vrrpStat	Description
0x0001	flow is VRRP
0x0002	invalid version
0x0004	invalid type
0x0008	invalid checksum
0x0010	invalid TTL (should be 255)
0x0020	invalid destination IP (should be 224.0.0.18)
0x0040	—
0x0080	—
0x0100	Virtual Router ID list truncated... increase VRRP_NUM_VRID
0x0200	IP list truncated... increase VRRP_NUM_IP
0x0400	—
0x0800	—
0x1000	—
0x2000	—
0x4000	Packet snapped
0x8000	Malformed packet... covert channel?

80.3.2 vrrpVer

The vrrpVer column is to be interpreted as follows:

vrrpVer	Description
0x04	VRRPv2
0x08	VRRPv3

80.3.3 vrrpType

The vrrpType column is to be interpreted as follows:

vrrpType	Description
0x01	Advertisement

80.3.4 vrrpAuthType

The vrrpAuthType column is to be interpreted as follows:

vrrpAuthType	Description
0x01	No authentication
0x02	Simple text password
0x04	IP Authentication Header

80.4 Monitoring Output

In monitoring mode, the vrrpDecode plugin outputs the following columns:

Column	Type	Description	Flags
vrrp2NPkts	U64	Number of VRRPv2 packets	
vrrp3NPkts	U64	Number of VRRPv3 packets	
vrrpStat	H16	Status	

80.5 Plugin Report Output

The following information is reported:

- Aggregated [vrrpStat](#)
- Number of VRRPv2 packets
- Number of VRRPv3 packets

80.6 Additional Output

Non-standard output:

- PREFIX_vrrp.txt: VRRP routing tables

The routing tables contain the following columns:

Name	Description
VirtualRtrID	Virtual router ID
Priority	Priority
SkewTime	Skew time (seconds)
MasterDownInterval	Master down interval (seconds)
AddrCount	Number of addresses
Addresses	List of addresses
Version	VRRP version
Type	Message type
AdverInt	Advertisement interval (seconds)
AuthType	Authentication type
AuthString	Authentication string
Checksum	Stored checksum
CalcChecksum	Calculated checksum
flowInd	Flow index

80.7 Post-Processing

The routing tables can be pruned by using the following command:

```
sort -u PREFIX_vrrp.txt > PREFIX_vrrp_pruned.txt
```

81 vtpDecode

81.1 Description

The vtpDecode plugin analyzes the VLAN Trunking Protocol (VTP) protocol.

81.2 Dependencies

81.2.1 Core Configuration

This plugin requires the following core configuration:

- \$T2HOME/tralyzer2/src/networkHeaders.h:
 - ETH_ACTIVATE>0

81.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description	Flags
VTP_AGGR	1	Aggregate updater identity	
VTP_SAVE	1	Extract all VLANs info in a separate file	
VTP_DEBUG	0	Print debug messages	
VTP_TS_FRMT	1	Format for timestamps: 0: string, 1: timestamp	
VTP_NUM_UPDID	16	Max number of updater identity	
VTP_STR_MAX	64	Max length for strings	
VTP_SUFFIX	"_vtp.txt"	Suffix for separate file	VTP_SAVE=1
VTP_VLANID_FRMT	1	Format for VLAN ID: 0: int, 1: hex	VTP_SAVE=1

81.3.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- VTP_SUFFIX

81.4 Flow File Output

The vtpDecode plugin outputs the following columns:

Column	Type	Description	Flags
vtpStat	H8	Status	
vtpVer	H8	Version	
vtpCodeBF	H8	Aggregated codes	
vtpVlanTypeBF	H8	Aggregated VLAN types	
vtpDomain	S	Management Domain	
vtpNumUpdId	U32	Number of Updater identity	VTP_NUM_UPDID>0
vtpUpdId	R(IP4)	Updater identity	VTP_NUM_UPDID>0

Column	Type	Description	Flags
vtpFirstUpdTS	S/TS	Timestamp of first update	VTP_TS_FRMT=0/1
vtpLastUpdTS	S/TS	Timestamp of last update	VTP_TS_FRMT=0/1

81.4.1 vtpStat

The vtpStat column is to be interpreted as follows:

vtpStat	Description
0x0001	Flow is VTP
0x0002	Different versions used
0x0004	Different Management Domains used
0x0008	-
0x0010	Invalid Management Domain Length (> 32)
0x0020	Invalid version
0x0040	Invalid code
0x0080	Invalid VLAN type
0x0100	—
0x0200	—
0x0400	—
0x0800	—
0x1000	—
0x2000	Array truncated... increase VTP_NUM_UPDID
0x4000	String truncated... increase VTP_STR_MAX
0x8000	Packet snapped, decoding failed

81.4.2 vtpCode and vtpCodeBF

The vtpCode and vtpCodeBF columns are to be interpreted as follows:

vtpCode	vtpCodeBF	Description
--	0x01	—
0x01	0x02	Summary Advertisement
0x02	0x04	Subset Advertisement
0x03	0x08	Advertisement Request
0x04	0x10	Join/Prune Message
--	0x20	—
--	0x40	—
--	0x80	Unknown VTP code

81.4.3 vtpVlanType and vtpVlanTypeBF

The vtpVlanType and vtpVlanTypeBF columns are to be interpreted as follows:

vtpVlanType	vtpVlanTypeBF	Description
--	0x01	—
0x01	0x02	Ethernet
0x02	0x04	Fiber Distributed Data Interface (FDDI)
0x03	0x08	Token Ring Concentrator Relay Function (TrCRF)
0x04	0x10	Fiber Distributed Data Interface Network Entity Title (FFID-net)
0x05	0x20	Token Ring Bridge Relay Function (TrBRF)
--	0x40	—
--	0x80	Unknown VTP VLAN type

81.5 Packet File Output

In packet mode (-s option), the vtpDecode plugin outputs the following columns:

Column	Type	Description	Flags
vtpStat	H8	Status	
vtpVer	H8	Version	
vtpCode	H8	Code	
vtpDomain	SC	Management Domain	
vtpVlanTypeBF	H8	Aggregated VLAN type	

81.6 Plugin Report Output

The following information is reported:

- Aggregated vtpStat
- Aggregated vtpCodeBF
- Aggregated vtpVlanTypeBF
- Number of VTP packets
- Number of VTPv1, VTPv2 and VTPv3 packets
- Number of VTP Summary Advertisement packets
- Number of VTP Subset Advertisement packets
- Number of VTP Advertisement Request packets
- Number of VTP Join/Prune Message packets
- Number of VTP packets with unknown type

81.7 Additional Output

Non-standard output:

- PREFIX_vtp.txt: List of VLANs extracted from Subset Advertisement messages

The PREFIX_vtp.txt file contains the following columns:

Name	Description
pktNo	Packet number
flowInd	Flow index
srcMac	MAC address which issued this advertisement
vtpVer	VTP version
vtpDomain	VTP Management Domain
vtpRevNum	VTP Configuration Revision Number
vtpVlanType	Aggregated VLAN type
vtpVlanID	ISL VLAN ID
vtpVlanName	VLAN Name
vtpVlanSAID	802.10 Index (IEEE 802.10 security association identifier for this VLAN)
vtpVlanMTU	MTU Size
vtpVlanSuspended	State of the VLAN (suspended or not)

81.8 References

- [Understanding VLAN Trunk Protocol \(VTP\)](#)

82 wavelet

82.1 Description

The wavelet plugin calculates the Daubechies wavelet transformation of the IP packet length or the inter-arrival time of packets (IAT).

82.2 Configuration Flags

The following flags, defined in *define_global.h*, can be used to control the output of the plugin:

Name	Default	Description
WAVELET_IAT	0	Values to analyze: 0: Packet length, 1: Inter-arrival times (IAT)
WAVELET_SIG	0	Print signal
WAVELET_PREC	0	Precision: 0: Float, 1: Double
WAVELET_THRES	8	Min. number of packets for analysis
WAVELET_MAX_PKT	40	Max. number of selected packets
WAVELET_LEVEL	3	Wavelet decomposition level
WAVELET_EXTMODE	ZPD	Extension Mode: NON: No extension, SYM: Symmetrization, ZPD: Zero-padding
WAVELET_TYPE	DB3	Mother Wavelet: DB1: Daubechies 1 wavelet DB2: Daubechies 2 wavelet DB3: Daubechies 3 wavelet DB4: Daubechies 4 wavelet

82.3 Flow File Output

The wavelet plugin outputs the following columns:

Name	Type	Description	Flags
waveNumPnts	U16	Number of points	
waveSig	R(F/D)	Packet length / IAT signal	WAVELET_PREC=0/1
waveNumLvl	U32	Number of wavelet levels	
waveCoefDetail	R(R(F/D))	Wavelet detail coefficients	WAVELET_PREC=0/1
waveCoefApprox	R(R(F/D))	Wavelet approximation coefficients	WAVELET_PREC=0/1

83 wechatDecode

83.1 Description

The wechatDecode plugin detects and decodes JCE encoded data found in observed HTTP traffic and writes the results as JSON into a file.

To identify relevant packets to decode, the HTTP payload is searched for the string "TMASDK_". If this string is found, the HTTP payload is processed using a JCE decoder.

Decryption and decompression of the body are handled automatically.

83.2 Dependencies

83.2.1 External Libraries

This plugin depends on the **zlib** library.

Ubuntu:	sudo apt-get install zlib1g-dev
Arch:	sudo pacman -S zlib
Gentoo:	sudo emerge zlib
openSUSE:	sudo zypper install zlib-devel
Red Hat/Fedora ⁶⁰ :	sudo dnf install zlib-devel
macOS ⁶¹ :	brew install zlib

83.3 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
WECHAT_JSON_SUFFIX	"_wechat.json"	Suffix appended to the base output file name
WECHAT_JSON_ARRAY	0	0: Output a single JSON array 1: Output a line-delimited JSON
WECHAT_INITIAL_JSON_BUFFER_SIZE	2048	Initial size of output buffer (increased dynamically)
WECHAT_MAX_HTTP_HDR_FIELD_LEN	1024	Maximum length of a HTTP header field
WECHAT_MAX_QUA_MATCH_LEN	255	Max. length of a matching group in the QUA string
WECHAT_VERBOSITY_LEVEL	0	Verbosity level: 0: Quiet mode (only warnings and errors) 1: Debug mode

83.4 Flow File Output

This plugin does not output any columns in the flow file. It writes the decoded data directly to the JSON output file.

⁶⁰If the `dnf` command could not be found, try with `yum` instead

⁶¹Brew is a packet manager for macOS that can be found here: <https://brew.sh>

83.5 Custom File Output

The wechatDecode plugin produces JSON output in the file `PREFIX_wechat.json`, where `PREFIX` is provided via the Tr-analyzer option `-w` or `-W` and the `_wechat.json` suffix can be overridden via the configuration flag `WECHAT_JSON_SUFFIX`.

83.5.1 JSON format

For each packet detected to contain JCE data, the HTTP payload is decoded, decrypted and, if necessary, decompressed. The extracted data is then written as one JSON object per packet.

Each JSON object is divided into three sections, i.e., keys: `flow`, `requestHeader` and `body`. The `flow` section contains the six-tuple and the `firstSeen` timestamp of the flow. The `requestHeader` contains any information extracted from the `reqHead` part of the JCE payload, including any optional fields. The `body` section contains the decrypted and decoded information found in the `body` part of the decoded HTTP payload.

Numerical values in the `requestHeader` extracted from the `reqHead` are printed as quoted JSON strings as there are no static guarantees on the bounds of the matched numbers. This avoids integer overflows and unexpected incorrect results.

The following currently understood body types are supported:

- ReportLog
- GetSettings
- GetConfig
- StatReport

If the configuration flag `WECHAT_JSON_ARRAY` is set to 0, the JSON objects are output as line-delimited JSON, i.e. one JSON object per line. Note that in this mode, a JSON object may not contain a newline character as it is already used as a delimiter between individual JSON objects.

If `WECHAT_JSON_ARRAY` is 1, the JSON objects are enclosed in a JSON array and separated with a comma to yield valid JSON output. The line-delimited JSON mode is more robust as it does not rely on proper termination of the application to write the closing square bracket.

84 scripts

This section describes various scripts and utilities for Tranalyzer. For a complete list of options, use the scripts `-h` option.

84.1 b64ex

Extracts all HTTP, EMAIL, FTP, TFTP, etc base 64 encoded content extracted from T2. To produce a list of files containing base64 use `grep` as indicated below:

- `grep "base64" /tmp/SMTPFILE/*`
- `./b64ex /tmp/SMTPFILES/file@wurst.ch_0_1223`

84.2 fpsGplt

Transforms the output of the `nFrstPkts` plugin signal output to gnuplot or `t2plot` format for encrypted traffic mining purposes. It generates an output file: `flowfile_nps.txt` containing the processed PL signal according to `nFrstPkts` plugin configuration.

```
$ fpsGplt -h
```

Usage:

```
fpsGplt [OPTION...] <FILE_flows.txt>
```

Optional arguments:

<code>-f findex</code>	Flow index to extract [default: all flows]
<code>-d A B</code>	Flow direction: A or B only [default: A and B]
<code>-t</code>	No time, but counts on x axis [default: time on x axis]
<code>-i</code>	Invert B flow PL
<code>-s</code>	Time sorted ascending
<code>-p s</code>	Sample sorted signal with smplIAT in [s]; $f = 1/\text{smplIAT}$
<code>-e s</code>	Time for each PL pulse edge in [s]
<code>-j</code>	Calculate the jumps in IAT and report appropriate values for MINIAT(S/U)
<code>-h, --help</code>	Show this help, then exit

If `-f` is omitted all flows will be included. If `-d` is omitted both flow directions will be processed. `-t` removes the timestamp and replaces it with an integer count. `-i` inverts the B flow signal to produce a symmetrical signal. `-p` samples the sorted signal with the IAT in seconds resp. frequency you deem necessary and `-e` defines the pulse flank in seconds. `-j` calculates the jumps in IAT to allows the user to choose an appropriate MINIAT(S/U) in `nFrstPkts` plugin.

84.3 gpq3x

Use this script to create 3D waterfall plot. Was originally designed for the `centrality` plugin:

```
cat FILE_centrality | ./gpq3x
```

The script can be configured through the command line. For a full list of options, run `./gpq3x -help`

84.4 osStat

Counts the number of hosts of each operating system (OS) in a PCAP file. In addition, a file with suffix _IP_OS.txt mapping every IP to its OS is created. This script uses p0f which requires a fingerprints file (p0f.fp), the location of which can be specified using the -f option. Version 2 looks first in the current directory, then in /etc/p0f. Version 3 looks only in the current directory.

- list all the options: osStat --help
- top 10 OS: osStat file.pcap -n 10
- bottom 5 OS: osStat file.pcap -n -5

84.5 protStat

The protStat script can be used to sort the PREFIX_protocols.txt file (generated by the [protoStats](#) plugin) or the PREFIX_nDPI.txt file (generated by the [nDPI](#) plugin) for the most or least occurring protocols (in terms of number of packets or bytes). In addition, it can also sort the PREFIX_icmpStats.txt and PREFIX_igmpStats.txt files (generated by the [icmpDecode](#) and [igmpDecode](#) plugins respectively). It can output the top or bottom *N* protocols or only those with at least a given percentage:

- list all the options: protStat --help
- for better readability, use protStat with tcol: protStat ... | tcol
- sorted list of protocols (by packets): protStat PREFIX_protocols.txt
- sorted list of protocols (by bytes): protStat PREFIX_protocols.txt -b
- top 10 protocols (by packets): protStat PREFIX_protocols.txt -n 10
- bottom 5 protocols (by bytes): protStat PREFIX_protocols.txt -n -5 -b
- protocols with packets percentage greater than 20%: protStat PREFIX_protocols.txt -p 20
- protocols with bytes percentage smaller than 5%: protStat PREFIX_protocols.txt -b -p -5
- TCP and UDP statistics only: protStat PREFIX_protocols.txt -udp -tcp

84.6 statGplt

Transforms 2/3D statistics output from [pktSIATHisto](#) plugin to gnuplot or [t2plot](#) format for encrypted traffic mining purposes.

84.7 t2_aliases

Set of aliases for Tranalyzer.

84.7.1 Description

t2_aliases defines the following aliases, functions and variables:

T2HOME

Variable pointing to the root folder of Tranalyzer, e.g., `cd $T2HOME`.

T2PLHOME

Variable pointing to the root folder of Tranalyzer plugins, e.g., `cd $T2PLHOME`. In addition, every plugin can be accessed by typing its name instead of its full path. For example to access `tcpFlags` home folder, `tcpFlags` can be used instead of `cd $T2PLHOME/tcpFlags` or `cd $T2HOME/plugins/tcpFlags`.

tran

Shortcut to access `$T2HOME`, e.g., `tran`

tranpl

Shortcut to access `$T2PLHOME`, e.g., `tranpl`

.tran

Shortcut to access `$HOME/.tranalyzer/plugins`, e.g., `.tran`

awkf

Configures `awk` to use tabs, i.e., ‘\t’ as input and output separator (prevents issue with repetitive values), e.g., `awkf '{ print $4 }' file_flows.txt`

tawk

Shortcut to run `tawk` from anywhere, e.g., `tawk`

tcol

Displays columns with minimum width, e.g., `tcol file_flows.txt`.

lsx

Displays columns with fixed width (default: 40), e.g., `lsx file_flows.txt` or `lsx 45 file_flows.txt`.

Note that ZSH already defines a `lsx` alias, therefore if using ZSH this command will **NOT** be installed. To have it installed, add the following line to your `~/.zshrc` file: `unalias lsx`

sortu

Sort rows and count the number of times a given row appears, then sort by the most occurring rows. (Alias for `sort | uniq -c | sort -rn`). Useful, e.g., to analyze the most occurring user-agents: `tawk '{ print $httpUsrAg }' FILE_flows.txt | sortu`

sortup

Same as `sortu`, but display the relative percentage instead of the absolute count. e.g., to analyze the most occurring user-agents: `tawk '{ print $httpUsrAg }' FILE_flows.txt | sortup`

t2

Shortcut to run Tranalyzer from anywhere, e.g., `t2 -r file.pcap -w out`

gt2

Shortcut to run Tranalyzer in `gdb` (Linux) or `lldb` (macOS) from anywhere, e.g., `gt2 -r file.pcap -w out`

st2

Shortcut to run Tranalyzer with sudo, e.g., st2 -i eth0 -w out

tranalyzer

Shortcut to run Tranalyzer from anywhere, e.g., tranalyzer -r file.pcap -w out

fextractor

Shortcut to run fextractor from anywhere, e.g., fextractor -r file_flows.xer 1234

fpsGplt

Shortcut to run fpsGplt from anywhere, e.g., fpsGplt file_flows.txt

protStat

Shortcut to run protStat from anywhere, e.g., protStat file_protocols.txt

statGplt

Shortcut to run statGplt from anywhere, e.g., statGplt file_flows.txt

t2b2t

Shortcut to run t2b2t from anywhere, e.g., t2b2t -r file_flows.bin -w file_flows.txt.

t2build

Function to build Tranalyzer and the plugins from anywhere, e.g., t2build tcpFlags. Use <tab> to list the available plugins and complete names. Use t2build -h for a full list of options.

t2caplist

Shortcut to run t2caplist from anywhere, e.g., t2caplist

t2conf

Shortcut to run t2conf from anywhere, e.g., t2conf --gui

t2dmon

Shortcut to run t2dmon from anywhere, e.g., t2dmon dumps/

t2doc

Shortcut to run t2doc from anywhere, e.g., t2doc tranalyzer2

t2docker

Shortcut to run t2docker from anywhere, e.g., t2docker -r file.pcap

t2dpdk

Shortcut to run t2dpdk from anywhere, e.g., t2dpdk -N 4 -i 0000:04:00.0.

t2flowstat

Shortcut to run t2flowstat from anywhere, e.g., t2flowstat file_flows.txt -c pktsSnt -s 1 -m 9000 -o

t2fm

Shortcut to run [t2fm](#) from anywhere, e.g., `t2fm -r file.pcap`

t2fuzz

Shortcut to run [t2fuzz](#) from anywhere, e.g., `t2fuzz file.pcap`

t2locate

Shortcut to run [t2locate](#) from anywhere, e.g., `t2locate`

t2mmdb

Shortcut to run `t2mmdb` (see [geoip](#) plugin documentation for more information) from anywhere, e.g., `t2mmdb`

t2netID

Shortcut to run [t2netID](#) from anywhere, e.g., `t2netID 0x138020a5`

t2plot

Shortcut to run [t2plot](#) from anywhere, e.g., `t2plot file.txt`

t2plugin

Shortcut to run [t2plugin](#) from anywhere, e.g., `t2plugin -c pluginName`.

t2rrd

Shortcut to run [t2rrd](#) from anywhere, e.g., `t2 -i eth0 | t2rrd -m` or `t2rrd V4Pkts V6Pkts`

t2stat

Shortcut to run [t2stat](#) from anywhere, e.g., `t2stat -USR2`

t2test

Shortcut to run `tests/T2Tester.py` from anywhere, e.g., `t2test tranalyzer2`

t2timeline

Shortcut to run [t2timeline](#) from anywhere, e.g., `t2timeline file.txt`

t2topcap

Shortcut to run [t2topcap](#) from anywhere, e.g., `t2topcap file.pcapng`

t2update

Shortcut to run `./setup.sh -C` from anywhere, e.g., `t2update`. This alias checks for the availability of a new version of Tranalyzer and proceed with the installation if requested.

t2viz

Shortcut to run [t2viz](#) from anywhere, e.g., `t2viz file.txt`

t2whois

Shortcut to run [t2whois](#) from anywhere, e.g., `t2whois 1.2.3.4`

84.7.2 Usage

Those aliases can be activated using either one of the following methods:

1. Append the content of this file to `~/.bash_aliases` or `~/.bashrc`
2. Append the following line to `~/.bashrc` (make sure to replace `$T2HOME` with the actual path, e.g., `$HOME/tranalyzer2-0.9.3`):

```
if [ -f "$T2HOME/scripts/t2_aliases" ]; then
    . $T2HOME/scripts/t2_aliases          # Note the leading `.'
fi
```

84.7.3 Known Bugs and Limitations

ZSH already defines a `1sx` alias, therefore if using ZSH this command will **NOT** be installed. To have it installed, add the following line to your `~/.zshrc` file: `unalias 1sx`

84.8 t2alive

In order to monitor the status of T2, the `t2alive` script sends syslog messages to server defined by the user whenever the status of T2 changes. It acquires the PID of the T2 process and transmits every REP seconds a `kill -SYS $pid`. If T2 answers with a corresponding kill command defined in `tranalyzer.h`, s.b., then status is set to alive, otherwise to dead. Only if a status change is detected a syslog message is transmitted. The following constants residing in `tranalyzer.h` govern the functionality of the script:

Name	Default	Description
SERVER	"127.0.0.1"	syslog server IP
PORT	514	syslog server port
FAC	"<25>"	facility code
STATFILE	"/tmp/t2alive.txt"	alive status file
REP	10	T2 test interval [s]

Table 471: *t2alive* script configuration

T2 on the other hand has also to be configured. To preserve simplicity the unused SYS interrupt was abused to respond to the `t2alive` request, hence the monitoring mode depending on USR1 and USR2 can be still functional. Configuration is carried out in `tranalyzer.h` according to the table below:

Name	Default	Description
REPSUP	0	1: activate alive mode
ALVPROG	"t2alive"	name of control program
REPCMDAW	"a='pgrep ALVPROG'; if [\$a]; then kill -USR1 \$a; fi"	alive and stall (no packets, looping?)
REPCMDAS	"a='pgrep ALVPROG'; if [\$a]; then kill -USR2 \$a; fi"	alive and well (working)

Table 472: T2 configuration for *t2alive* mode

`REPSUP=1` activates the alive mode. If more functionality is requested the `REPCMDA` constant facilitates the necessary changes. On some Linux distributions the pcap read callback function is not thread safe, thus signals of any kind might

lead to crashes especially when capturing live traffic. Therefore **MONINTTHR=1** in *main.h* is set by default. Note that *t2alive* should be executed in a shell as a standalone script. If executed as a cron job, the while loop and the sleep command has to be removed, as described in the script itself.

84.9 t2b2t

The program *t2b2t* can be used to transform binary Tranalyzer files generated by the *binSink* or *socketSink* plugin into text or json files. The converted files use the same format as the ones generated by the *txtSink* or *jsonSink* plugin.

The program can be found in `$T2HOME/utils/t2b2t` and can be compiled by typing `make`.

The use of the program is straightforward:

- `bin→txt`: `t2b2t -r FILE_flows.bin -w FILE_flows.txt`
- `bin→json`: `t2b2t -r FILE_flows.bin -j -w FILE_flows.json`
- `bin→compressed txt`: `t2b2t -r FILE_flows.bin -c -w FILE_flows.txt.gz`
- `bin→compressed json`: `t2b2t -r FILE_flows.bin -c -j -w FILE_flows.json.gz`

If the `-w` option is omitted, the destination is inferred from the input file, e.g., the examples above would produce the same output files with or without the `-w` option. Note that `-w -` can be used to output to stdout.

Additionally, the `-n` option can be used **not** to print the name of the columns as the first row.

Try `t2b2t -h` for more information.

84.10 t2caplist

Generates a list of PCAP files with absolute path to use with Tranalyzer `-R` option. If no argument is provided, then lists all the PCAP files in the current directory. If a folder name is given, lists all capture files in the folder. If a list of files is given, list those files. Try `t2caplist -help` for more information.

- `t2caplist > pcap_list.txt`
- `t2caplist ~/dumps/ > pcap_list.txt`
- `t2caplist ~/dumps/testnet*.pcap > pcap_list.txt`

84.11 t2conf

Use *t2conf* to configure, activate and deactivate Tranalyzer plugins:

- To change the value of a configuration flag run:

```
t2conf pluginName -D FLAG_NAME=new_value
```

- To check the value of a configuration flag run:

```
t2conf pluginName -G FLAG_NAME
```

- To list the configuration flags available run:

```
t2conf pluginName -I
```

- To reset a plugin configuration to its default values run:

```
t2conf pluginName --reset
```

- To save a plugin configuration run:

```
t2conf pluginName -g
```

- Note that the `-g` option accepts a filename:

```
t2conf pluginName -g /where/to/save/file.config
```

- To load plugin configuration run:

```
t2conf pluginName -C /path/to/file.config
```

- Note that if the default filename is used, `-C auto` can be used instead:

```
t2conf pluginName -C auto
```

- For more details about `t2conf`, run:

```
t2conf --help
```

Alternatively, use `t2conf --gui` option or the `t2plconf` script provided with all the plugins to configure individual plugins as follows:

- `cd $T2PLHOME/pluginName`
- `./t2plconf`
 - Navigate through the different options with the up and down arrows
 - Use the left and right arrows to select an action:
 - * `ok`: apply the changes
 - * `configure`: edit the selected entry (use the space bar to select a different value)
 - * `cancel`: discard the changes
 - * `edit`: open the file containing the selected option in `EDITOR` (default: `vim`)
 - Use the space bar to select a different value

A more detailed description of the script can be found in [Tranalyzer2 documentation](#).

84.11.1 Dependencies

The `t2conf` (if the `--gui` option is used) and `t2plconf` scripts require `dialog` (version 1.1-20120703 minimum) and the `vim` editor. The easiest way to install them is to use the `install.sh` script provided (Section 84.11.2). Note that the editor can be changed by exporting the environment variable `EDITOR` as follows: `export EDITOR=/path/to/editor`, e.g., `export EDITOR=/usr/bin/nano` or by setting the `EDITOR` variable at line 9 of the `t2conf` script and at line 90 of the `t2plconf` script.

84.11.2 Installation

The easiest way to install `t2conf` and its dependencies is to use `tranalyzer2 setup.sh` script. Alternatively, the provided `install.sh` script can be used to only install `t2conf`. Run `./install.sh --help` to see what can be installed.

Alternatively, use `t2_aliases` or add the following alias to `~/.bash_aliases`:

```
alias t2conf="$T2HOME/scripts/t2conf/t2conf"
```

Where `$T2HOME` is the root folder containing the source code of Tranalyzer2 and its plugins, i.e., where `README.md` is located.

84.11.3 Usage

For a complete list of options use the `-h` option, i.e., `t2conf -h`, or the man page (`man t2conf`).

84.11.4 Patch

`t2conf` can be used to patch Tranalyzer and the plugins (useful to save settings such as hash table size, IPv6, ...).

The format of the patch file is as follows:

- Empty lines and lines starting with '%' or '#' are ignored
- Filenames are relative to `$T2HOME`
- A line is composed of three or four tabs (not spaces) separated columns:
 - NAME <tab> newvalue <tab> oldvalue <tab> file
 - NAME <tab> newvalue <tab> file
- `--patch` uses `newvalue`
- `--rpatch` uses `oldvalue`⁶²

As an example, let us take the value `T2PSKEL_IP` defined in `t2PSkel/src/t2PSkel.h`:

```
#define T2PSKEL_IP    1 // whether or not to output IP (var2)
```

A patch to set this value to 0 would look as follows (where the spaces between the columns are tabs, i.e., '\t'):

- `T2PSKEL_IP 0 1 t2PSkel/src/t2PSkel.h`
- `T2PSKEL_IP 0 t2PSkel/src/t2PSkel.h`

84.12 t2dmon

Monitors a folder for new files and creates symbolic links with incrementing indexes. This can be used with the `-D` option when the filenames have either multiple indexes, e.g., date and count, or when the filenames do not possess an index.

84.12.1 Dependencies

This script requires `inotify-tools`:

⁶²This option is not valid if the patch has only three columns.

Arch: sudo pacman -S inotify-tools

Fedora: sudo yum install inotify-tools

Gentoo: sudo emerge inotify-tools

Ubuntu: sudo apt-get install inotify-tools

84.12.2 Usage

t2dmon works as a daemon and as such, should either be run in the background (the ampersand `&' in step 1 below) or on a different terminal.

1. t2dmon dumps/ -o nudel.pcap &
2. tranalyzer -D dumps/nudel.pcap0 -w out
3. Finally, copy/move the pcap files into the dumps/ folder.

84.13 t2doc

Access Tranalyzer documentation from anywhere, e.g., t2doc tcpFlags. Use `<tab>` to list the available plugins and complete names.

84.14 t2docker

Create and manage Tranalyzer Docker containers.

- Create a Docker container: `t2docker -B t2-latest.tar.gz`
- Download the latest version of T2 and create a Docker container: `t2docker -B latest`
- List existing Tranalyzer Docker containers: `t2docker -ls`
- Save a Docker image: `t2docker -S image-name-or-id`
- Load a Docker image: `t2docker -L t2docker-image.tar[.gz]`
- Get a Shell in a Docker image: `t2docker -X image-name-or-id`
- Run Tranalyzer inside a Docker container: `t2docker -r file.pcap`
- Run another T2 command inside a Docker container: `t2docker tawk -V flowStat`

Note that when running T2 inside a Docker container, the pcap file is copied to a temporary folder (automatically removed or shredded (`--shred` option) after the script has ended). This extra step make sure only the required files are accessible by the container.

84.15 t2dpdk

Run N instances of T2 in DPDK multi-process mode. T2 must be compiled with DPDK_MP=1:

```
$ t2conf tranalyzer2 -D DPDK_MP=1
$ t2build -R -r -f
$ t2dpdk -N 4 -i 0000:04:00.0
```

84.16 t2flowstat

Calculates statistical distributions of selected columns/flows from a flow file.

84.17 t2fm

Generates a PDF report out of:

- a flow file (-F option): t2fm -F file_flows.txt
- a live interface (-i option): t2fm -i eth0
- a PCAP file (-r option): t2fm -r file.pcap
- a list of PCAP files (-R option): t2fm -R pcap_list.txt

84.17.1 Required Plugins

- | | | |
|-------------|--------------|-----------|
| • basicFlow | • basicStats | • txtSink |
|-------------|--------------|-----------|

84.17.2 Optional Plugins

- | | | | |
|-------------|---------------|------------------|-------------|
| • arpDecode | • geoip | • nDPI | • pwX |
| • dnsDecode | • httpSniffer | • portClassifier | • sshDecode |

84.18 t2fuzz

Randomly corrupt a PCAP file and run T2 against it: t2fuzz file.pcap.

Try t2fuzz --help for more information.

84.19 t2locate

Query location database to acquire information about a city near float point terrestrial coordinates. Build the DB first with the script: \$T2HOME/scripts/t2locate/update_db, duration ca 30 min.

84.20 t2netID

Decode hexadecimal network IDs. An ID contain information about the country, the organization and whether or not the address is a Tor address. An ID can be decoded as follows: t2netID 0x138020a5.

Try t2netID --help for more information.

84.21 t2plot

2D/3D plot for Tranalyzer using gnuplot. First row of the input file must be the column names (may start with a ‘%’). The input file must contain one, two or more columns separated by tabs (\t). Columns to plot can be selected with -o option. Try t2plot --help for more information.

Dependencies: The t2plot script requires **gnuplot**.

Arch: sudo pacman -S gnuplot

Ubuntu: sudo apt-get install gnuplot-qt

macOS: brew install gnuplot --with-qt

Examples:

- tawk '{ print ip2num(shost()), ip2num(dhost()) }' f_flows.txt | t2plot -pt
- tawk '{ print ip2num(\$srcIP), \$timeFirst, \$connSip }' f_flows.txt | t2plot
- tawk '{ print \$pktsSnt }' f_flows.txt | t2plot -H 10 -sx "0:40"
- t2plot -o pktsSnt f_flows.txt
- t2plot -D -o srcIPCC:l7BytesSnt f_flows.txt
- tawk '{ print proto(), \$l7BytesSnt }' f_flows.txt | t2plot -D
- t2plot file_with_one_two_or_three_columns.txt
- t2plot -o "26:28" file_with_many_columns.txt
- t2plot -o "l7BytesSnt:l7BytesRcvd" file_with_many_columns.txt
- Try t2plot -e for more examples.

84.22 t2plugin

Use this script to create a new plugin or list existing plugins. For a more comprehensive description of how to write a plugin, refer to Appendix A (Creating a custom plugin) of [\\$T2HOME/doc/documentation.pdf](#).

84.23 t2rrd

Store Tranalyzer monitoring output into a RRD database (-m option) or use the RRD database generated with -m to monitor and plot various values, e.g., number of flows.

84.24 t2stat

Sends USR1 signal to Tranalyzer to produce intermediary report. The signal sent can be changed with the -SIGNAME option, e.g., t2stat -USR2 or t2stat -INT. If Tranalyzer was started as root, the -s option can be used to run the command with sudo. The -p option can be used to print the PID of running Tranalyzer instances and the -l option provides additional information about the running instances (command and running time). The -i option can be used to cycle through all the running instances and will prompt for confirmation before sending the signal to a specific process. If a numeric argument N is provided, sends the signal every N seconds, e.g., t2stat 10 to report every 10s. Use t2stat --help for more information.

84.25 t2timeline

Timeline plot of flows: t2timeline FILE_flows.txt

- To use relative time, i.e., starting at 0, use the `-r` option.
- The vertical space between A and B flows can be adapted with the `-v` option, e.g., `-v 50`.
- When hovering over a flow, the following information is displayed:
`flowInd_flowStat_srcIP:srcPort_dstIP:dstPort_l4Proto_vlanID`.
- Additional information can be displayed with the `-e` option, e.g., `-e macS,macD,duration`
- Use `t2timeline --help` for more information.

An example graph is depicted in Figure 11.

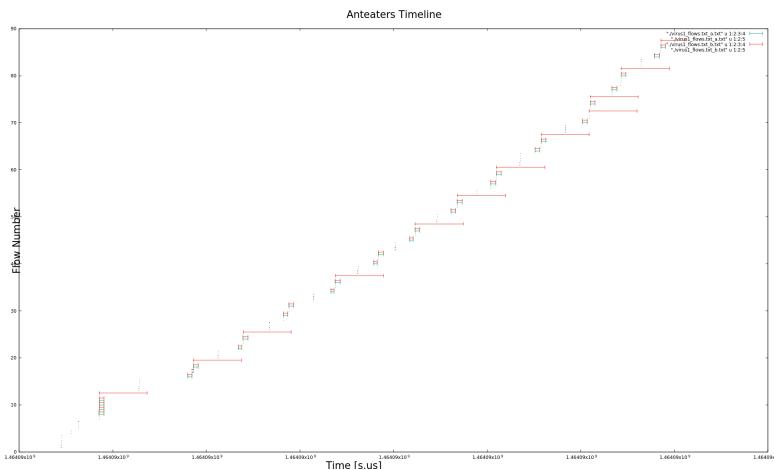


Figure 11: T2 timeline flow plot

84.26 t2topcap

Convert pcapng files to pcap: `t2topcap file.pcapng` Try `t2topcap --help` for more information.

84.27 t2utils.sh

Collection of bash functions and variables (readonly).

84.27.1 Usage

To access the functions and variables provided by this file, source it in your script as follows:

```
source "$(dirname "${0}")/t2utils.sh"
```

Note that if your script is not in the `scripts/` folder, you will need to adapt the path above to `t2utils.sh` accordingly.

[ZSH] If writing a script for ZSH, add the following line **BEFORE** sourcing the script:

```
unsetopt function_argzero
```

84.27.2 Colors

Alternatives to printbold, printerr, printinf, printok and printwrn:

Variable	Description	Example
<code> \${BLUE}</code>	Set the color to blue	<code>printf "\${BLUE}%s\${NOCOLOR}\n" "msg"</code>
<code> \${GREEN}</code>	Set the color to green	<code>printf "\${GREEN}%s\${NOCOLOR}\n" "msg"</code>
<code> \${ORANGE}</code>	Set the color to orange	<code>printf "\${ORANGE}%s\${NOCOLOR}\n" "msg"</code>
<code> \${RED}</code>	Set the color to red	<code>printf "\${RED}%s\${NOCOLOR}\n" "msg"</code>
<code> \${BLUE_BOLD}</code>	Set the color to blue bold	<code>printf "\${BLUE_BOLD}msg\${NOCOLOR}\n"</code>
<code> \${GREEN_BOLD}</code>	Set the color to green bold	<code>printf "\${GREEN_BOLD}msg\${NOCOLOR}\n"</code>
<code> \${ORANGE_BOLD}</code>	Set the color to orange bold	<code>printf "\${ORANGE_BOLD}msg\${NOCOLOR}\n"</code>
<code> \${RED_BOLD}</code>	Set the color to red bold	<code>printf "\${RED_BOLD}msg\${NOCOLOR}\n"</code>
<code> \${BLUE_ITALIC}</code>	Set the color to blue italic	<code>printf "\${BLUE_BOLD}\${msg}\${NOCOLOR}\n"</code>
<code> \${GREEN_ITALIC}</code>	Set the color to green italic	<code>printf "\${GREEN_BOLD}\${msg}\${NOCOLOR}\n"</code>
<code> \${ORANGE_ITALIC}</code>	Set the color to orange italic	<code>printf "\${ORANGE_BOLD}\${msg}\${NOCOLOR}\n"</code>
<code> \${RED_ITALIC}</code>	Set the color to red italic	<code>printf "\${RED_BOLD}\${msg}\${NOCOLOR}\n"</code>
<code> \${BLUE_UNDERLINE}</code>	Set the color to blue underline	<code>echo -e "\${BLUE_ITALIC}msg\${NOCOLOR}"</code>
<code> \${GREEN_UNDERLINE}</code>	Set the color to green underline	<code>echo -e "\${GREEN_ITALIC}msg\${NOCOLOR}"</code>
<code> \${ORANGE_UNDERLINE}</code>	Set the color to orange underline	<code>echo -e "\${ORANGE_ITALIC}msg\${NOCOLOR}"</code>
<code> \${RED_UNDERLINE}</code>	Set the color to red underline	<code>echo -e "\${RED_ITALIC}msg\${NOCOLOR}"</code>
<code> \${BOLD}</code>	Set the font to bold	<code>echo -e "\${BOLD}\${msg}\${NOCOLOR}"</code>
<code> \${ITALIC}</code>	Set the font to italic	<code>echo -e "\${ITALIC}\${msg}\${NOCOLOR}"</code>
<code> \${UNDERLINE}</code>	Set the font to underline	<code>echo -e "\${UNDERLINE}\${msg}\${NOCOLOR}"</code>
<code> \${STRIKETHROUGH}</code>	Set the font to strikethrough	<code>echo -e "\${STRIKETHROUGH}\${msg}\${NOCOLOR}"</code>
<code> \${NOCOLOR}</code>	Reset the color	<code>echo -e "\${BOLD}\${msg}\${NOCOLOR}"</code>

84.27.3 Folders

Variable	Description	Example
<code> \${SHOME}</code>	Points to the folder where the script resides	For <code>macRecorder/utils/mconv</code> , <code> \${SHOME}</code> is <code> \${T2PLHOME}/macRecorder/utils</code>
<code> \${T2HOME}</code>	Points to the root folder of Tranalyzer	<code>cd "\${T2HOME}/tranalyzer2"</code>
<code> \${T2PLHOME}</code>	Points to the root folder of Tranalyzer plugins	<code>cd "\${T2PLHOME}/t2PSkel"</code>

84.27.4 Scripts and Programs

Functions and variables pointing to programs.

Functions

Name	Program	Example
AWK	gawk	AWK '{ print }' file
AWKF	gawk -F '\t' -v OFS='\t'	AWKF '{ print }' file
T2	Most recent t2analyzer executable in \${T2HOME}/t2analyzer2/	T2 -r file.pcap
T2B2T	Most recent t2b2t executable in \${T2HOME}/utils/t2b2t/	T2B2T -r file.bin -w file.txt
T2WHOIS	Most recent t2whois executable in \${T2HOME}/utils/t2whois/	T2WHOIS 1.2.3.4

Variables

Name	Program	Example
<code> \${AWK_EXEC}</code>	<code>gawk</code>	<code>"\${AWK_EXEC}" 'NR > 0 { print \$2 }' file</code>
<code> \${OPEN}</code>	<code>xdg-open (Linux), open (macOS)</code>	<code>"\${OPEN}" file.pdf</code>
<code> \${PYTHON}</code>	<code>python3, python or python2</code>	<code>"\${PYTHON}" file.py</code>
<code> \${READLINK}</code>	<code>readlink (Linux) / greadlink (macOS)</code>	<code>"\${READLINK}" file</code>
<code> \${SED}</code>	<code>sed (Linux) / gsed (macOS)</code>	<code>"\${SED}" `s/ _/g' << "\$str"</code>
<code> \${T2BUILD}</code>	<code>"\${T2HOME}/autogen.sh"</code>	<code>"\${T2BUILD}" tranalyzer2</code>
<code> \${T2CONF}</code>	<code>"\${T2HOME}/scripts/t2conf/t2conf"</code>	<code>"\${T2CONF}" basicFlow -D BFO_MAC=1</code>
<code> \${T2PLOT}</code>	<code>"\${T2HOME}/scripts/t2plot"</code>	<code>"\${T2PLOT}" -o srcIPCC file</code>
<code> \${TAWK}</code>	<code>"\${T2HOME}/scripts/tawk/tawk"</code>	<code>"\${TAWK}" '{ print tuple4() } file'</code>

84.27.5 Functions

Function	Description
<code>printbold "msg"</code>	print a message (bold) with a newline
<code>printer "msg"</code>	print an error message (red) with a newline
<code>printfinf "msg"</code>	print an info message (blue) with a newline
<code>printok "msg"</code>	print an ok message (green) with a newline
<code>printwrn "msg"</code>	print a warning message (orange) with a newline
<code>printfbold "msg"</code>	print a message (bold) without a newline
<code>printferr "msg"</code>	print an error message (red) without a newline
<code>printfinf "msg"</code>	print an info message (blue) without a newline
<code>printfok "msg"</code>	print an ok message (green) without a newline
<code>printfwrn "msg"</code>	print a warning message (orange) without a newline
<code>fatal "msg"</code>	print an error message (red) with a newline and exit with status 1
<code>check_dependency "bin" "pkg"</code>	check whether a dependency exists (Linux/macOS)
<code>check_dependency_linux "bin" "pkg"</code>	check whether a dependency exists (Linux)
<code>check_dependency_macos "bin" "pkg"</code>	check whether a dependency exists (macOS)
<code>test_min_version "ver" "req"</code>	return true if version number ver is greater than or equal to req
<code>has_define "file" "name"</code>	return 0 if the macro name exists in file, 1 otherwise
<code>get_define "name" "file"</code>	return the value of the macro name in file
<code>set_define "name" "value" "file"</code>	set the value of the macro name in file to value
<code>replace_suffix "name" "old" "new"</code>	replace the old suffix in name by new
<code>join_by "sep" "values..."</code>	join values... with a separator sep
<code>ask_default_no "msg" ["answer"]</code>	ask the question msg (y/N)?, read the answer and assume no if answer is not [yY] or [yY] [eE] [sS]. Return 0 if answer is

Function	Description
ask_default_yes "msg" ["answer"]	yes. answer can be set to yes or no to force the answer ask the question msg (Y/n) ?, read the answer and assume yes if answer is not [nN] or [nN] [oO]. Return 0 if answer is yes. answer can be set to yes or no to force the answer
find_most_recent_dir "dir" "dirname" find_most_recent_file "dir" "filename"	recursively find the most recent dirname in a directory dir recursively find the most recent filename in a directory dir
t2_build_exec "/path/to/exec" [force]	ask whether to build the given executable if it does not exist (set force to 1 to rebuild exec regardless)
get_t2_exec	search for t2analyzer executable in \${T2HOME}/t2analyzer2 (return an empty string, if it could not be found)
get_t2b2t_exec	search for t2b2t executable in \${T2HOME}/utils/t2b2t (return an empty string, if it could not be found)
get_t2whois_exec	search for t2whois executable in \${T2HOME}/utils/t2whois (return an empty string, if it could not be found)
abort_if_t2_exec_not_found	search for t2analyzer executable in \${T2HOME}/t2analyzer2. Abort with status 1 and an error and info message if it could not be found.
abort_if_t2b2t_exec_not_found	search for t2b2t executable in \${T2HOME}/utils/t2b2t. Abort with status 1 and an error and info message if it could not be found.
abort_if_t2whois_exec_not_found	search for t2whois executable in \${T2HOME}/utils/t2whois. Abort with status 1 and an error and info message if it could not be found.
t2_wget "url" ["outfile"]	download the data at url (set outfile (optional) to change the output path)
t2_wget_n "url" ["outfile"]	same as t2_wget, but turn on timestamping (see wget -N option)
get_nproc	return the number of processing units available
validate_float "float" validate_int "int" validate_num "num" validate_ip "string" validate_pcap "file"	return 0 if float is a valid floating point value, 1 otherwise return 0 if int is a valid integer, 1 otherwise return 0 if num is a valid positive integer, 1 otherwise return 0 if string is a valid IPv4 address, 1 otherwise return 0 if file is a valid PCAP file, 1 otherwise
validate_next_arg "curr" "next"	check whether next exists and is not an option (curr is used for error reporting)
validate_next_arg_exists "curr" "next"	check whether next exists
validate_next_dir "curr" "next"	check whether next exists and is a directory
validate_next_file "curr" "next"	check whether next exists and is a regular file

Function	Description
validate_next_file_or_dir "curr" "next"	check whether next exists and is a regular file or directory
validate_next_pcap "curr" "next"	check whether next exists and is a PCAP file
validate_next_num "curr" "next"	check whether next exists and is a positive integer
validate_next_int "curr" "next"	check whether next exists and is an integer
validate_next_float "curr" "next"	check whether next exists and is a float
arg_is_option "arg"	check whether arg exists and is an option (starts with -)
abort_missing_arg "option"	print an error about a missing argument and exit with status 1
abort_option_unknown "option"	print an error about an unknown option and exit with status 1
abort_required_file	print an error about a missing required file and exit with status 1
abort_required_dir	print an error about a missing required directory and exit with status 1
abort_required_file_or_dir	print an error about a missing required file or directory and exit with status 1
abort_with_help	print a message explaining how to get help and exit with status 1

84.28 t2viz

Generates a graphviz script which can be loaded into xdot or dotty: `t2viz FILE_flows.txt`.

Accepts T2 flow or packet files with header description.

Try `t2viz --help` for more information.

84.29 t2voipconv

Convert raw audio files extracted with [voipDetector](#) to wav: `t2voipconv /tmp/TranVoIP`.

Try `t2voipconv --help` for more information.

84.30 t2whois

Query Tranalyzer subnet databases to get geolocation information about IP addresses:

- Get geo info for one IP address: `t2whois 1.2.3.4`
- Get geo info for multiple IPs: `t2whois 1.2.3.4 5.6.7.8`
- Get geo info for one or multiple IPs listed in a file (one IP per line): `t2whois -r file.txt`
- Get geo info for one or multiple IPs entered to stdin via a prompt: `t2whois` or `t2whois -r -`
- Get geo info for one or multiple IPs entered to stdin via a pipe: `cat file.txt | t2whois -q -l`
- Format the output (one line per IP, no header): `t2whois -l -H -r file.txt`
- Format the output (one line per IP, comma separated): `t2whois -l -s "," -r file.txt`

- Query information about the databases `t2whois -V`
- Try `t2whois -h` for more information

84.31 topNStat

Generates sorted lists of all the columns (names or numbers) provided. A list of examples can be displayed using the `-e` option.

85 PDF Report Generation from PCAP using t2fm

85.1 Introduction

This tutorial presents `t2fm`, a script which generates a PDF report out of a PCAP file. Information provided in the report includes top source and destination addresses and ports, protocols and applications, DNS and HTTP activity and potential warnings, such as executable downloads or SSH connections.

85.2 Prerequisites

For this tutorial, it is assumed the user has a basic knowledge of Tranalyzer and that the file `t2_aliases` has been sourced in `~/.bashrc` or `~/.bash_aliases` as follows⁶³ (make sure to replace `$T2HOME` with the actual path, e.g., `$HOME/tranalyzer2-0.9.3`):

```
# $HOME/.bashrc

if [ -f "$T2HOME/scripts/t2_aliases" ]; then
    . "$T2HOME/scripts/t2_aliases"           # Note the leading '.'
fi
```

85.2.1 Required plugins

The following plugins must be loaded for `t2fm` to produce a useful report:

- `basicFlow`
- `basicStats`
- `txtSink`

85.2.2 Optional plugins

The following plugins are optional:

- `arpDecode`
- `dnsDecode`
- `geoip`
- `pwX`
- `sshDecode`
- `sslDecode`
- `httpSniffer` , configured as follows⁶⁴:
 - `HTTP_SAVE_IMAGE=1`
 - `HTTP_SAVE_VIDEO=1`
 - `HTTP_SAVE_AUDIO=1`
 - `HTTP_SAVE_MSG=1`
 - `HTTP_SAVE_TEXT=1`
 - `HTTP_SAVE_APPL=1`
- `nDPI` , configured as follows:
 - `NDPI_OUTPUT_STR=1`
- `portClassifier` , configured as follows:
 - `PBC_NUM=1`
 - `PBC_STR=1`

If one of those plugin is not loaded, messages like `N/A: dnsDecode plugin required` will be displayed in the PDF where the information could not be accessed.

⁶³Refer to the file `README.md` or to the documentation for more details

⁶⁴This is only required to report information about EXE downloaded

85.2.3 Packages

The following packages are required to build the PDF:

- texlive-latex-extra
- texlive-fonts-recommended

85.3 PCAP to PDF in One Command

For simplicity, this tutorial assumes the user wants a complete report, i.e., requires all of the optional plugins. The `-b` option builds and configures Tranalyzer and the plugins, the `-A` option opens the generated report.

1. Run `t2fm` directly on the PCAP file (the report will be named `file.pdf`):

```
t2fm -b -A -r file.pcap
```

85.4 Step-by-Step Instructions (PCAP to PDF)

Alternatively if you prefer to configure Tranalyzer, build the plugins and open the generated report yourself:

1. Make sure all the plugins are configured as described in Section 85.2
2. Build Tranalyzer and the plugins ⁶⁵:


```
t2build tranalyzer2 basicFlow basicStats txtSink arpDecode dnsDecode geoip \
httpSniffer nDPI portClassifier pwX sshDecode sslDecode
```

(Note that those first two steps can be omitted if `t2fm -b` option is used)
3. Run `t2fm` directly on the PCAP file (the report will be named `file.pdf`):


```
t2fm -r file.pcap
```
4. Open the generated PDF report `file.pdf` (Note that this step can be omitted if `t2fm -A` option is used):


```
evince file.pdf
```

85.5 Step-by-Step Instructions (flow file to PDF)

Alternatively, if you prefer to run Tranalyzer yourself or already have access to a flow file, replace step 3 of Section 85.4 with the following steps:

1. Run Tranalyzer on a pcap file as follows:


```
t2 -r file.pcap -w out
```
2. The previous command should have created the following files:


```
out_headers.txt
out_flows.txt
```
3. Run the `t2fm` script on the flow file generated previously:


```
t2fm -F out_flows.txt
```

⁶⁵Hint: use the tab completion to avoid typing the full name of all the plugins: `t2build tr<tab> ... ht<tab> ...`

85.6 Step-by-Step Instructions (ClickHouse / MongoDB / PostgreSQL to PDF)

If the `clickhouseSink`, `mongoSink` or `psqlSink` plugins were loaded, `t2fm` can use the created databases to generate the report (faster).

1. Follow point 1 and 2 from Section 85.4⁶⁶
2. Build the `clickhouseSink`, `mongoSink` or `psqlSink` plugin:
 - **ClickHouse:** `t2build clickhouseSink`
 - **MongoDB:** `t2build mongoSink`
 - **PostgreSQL:** `t2build psqlSink`
3. Run `Tralyzer` on a pcap file as follows:
`t2 -r file.pcap -w out`
4. Run the `t2fm` script on the database generated previously:
 - **ClickHouse:** `t2fm -C tralyzer`
 - **MongoDB:** `t2fm -m tralyzer`
 - **PostgreSQL:** `t2fm -p tralyzer`

When generating a report from a database a time range to query can be specified with the `-T` option. The complete format is as follows: `YYYY-MM-DD HH:MM:SS.USEC([+-]OFFSET|Z)`, e.g., `2022-08-23 12:34:56.912345+0100`. Note that only the required fields must be specified, e.g., `2022-08-23` is equivalent to `2022-08-23 00:00:00.000000`. For example, to generate a report from the 1st of July to the 11. of August 2022 at 14:59 from a PostgreSQL database, run the following command: `t2fm -p tralyzer -T "2022-07-01" "2022-08-11 14:59"`

85.7 NetFlow Records to PDF in One Command

For simplicity, this tutorial assumes the user wants a complete report. The `-A` option opens the generated report.

1. Run `t2fm` directly on the NetFlow records (file or directory) (the report will be named `2024.pdf`):

```
t2fm -A -N netflow/2024
```

85.8 Conclusion

This tutorial has presented how `t2fm` can be used to create a PDF report summarizing the traffic contained in a PCAP file or NetFlow records. Although not discussed in this tutorial, it is also possible to use `t2fm` on a live interface (`-i` option) or on a list of PCAP files (`-R` option). For more details, refer to `t2fm` man page or use `t2fm --help`.

⁶⁶`HTTP_SAVE_*` do not need to be set as EXE downloads detection is currently not implemented in the DB backends

86 tawk

86.1 Description

This document describes tawk and its functionalities. tawk works just like awk, but provides access to the columns via their names. In addition, it provides access to helper functions, such as `host()` or `port()`. Custom functions can be added in the folder named `t2custom` where they will be automatically loaded.

86.2 Dependencies

gawk version 4.1 is required.

Ubuntu:	sudo apt-get install gawk
Arch:	sudo pacman -S gawk
Gentoo:	sudo emerge gawk
openSUSE:	sudo zypper install gawk
Red Hat/Fedora ⁶⁷ :	sudo dnf install gawk
macOS ⁶⁸ :	brew install gawk

86.3 Installation

The recommended way to install tawk is to install `t2_aliases` as documented in `README.md`:

- Append the following line to `~/.bashrc` (make sure to replace `$T2HOME` with the actual path, e.g., `$HOME/tralyzer2-0.9.3`):

```
if [ -f "$T2HOME/scripts/t2_aliases" ]; then
    . $T2HOME/scripts/t2_aliases          # Note the leading `.'
fi
```

86.3.1 Man Pages

The man pages for tawk and `t2nfdump` can be installed by running: `./install.sh man`. Once installed, they can be consulted by running `man tawk` and `man t2nfdump` respectively.

86.4 Usage

- To list the column numbers and names: `tawk -l file_flows.txt`
- To list the column numbers and names as 3 columns: `tawk -l=3 file_flows.txt`
- To list the available functions: `tawk -g file_flows.txt`
- To list the available functions as 3 columns: `tawk -g=3 file_flows.txt`
- To save the original filename and filter used: `tawk -c 'FILTER' file_flows.txt > file.txt`

⁶⁷If the `dnf` command could not be found, try with `yum` instead

⁶⁸Brew is a packet manager for macOS that can be found here: <https://brew.sh>

- To extract all ICMP flows and the header: `tawk 'hdr() || $14Proto == 1' file_flows.txt > icmp.txt`
- To extract all ICMP flows without the header: `tawk -H 'icmp()' file_flows.txt > icmp.txt`
- To extract the flow with index 1234: `tawk '$flowInd == 1234' file_flows.txt`
- To extract all DNS flows and the header: `tawk 'hdr() || strtonum($dnsStat)' file_flows.txt`
- To consult the documentation for the function ‘func’: `tawk -d func`
- To consult the documentation for the functions ‘min’ and ‘max’: `tawk -d min,max`
- To consult the documentation for all the available functions: `tawk -d all`
- To consult the documentation for the variable ‘var’: `tawk -V var`
- To consult the documentation for the variable ‘var’ with value 0x8a: `tawk -V var=0x8a`
- To decode all variables from [Tranalyzer2](#) log file: `tawk -L out_log.txt`
- To decode all variables from [Tranalyzer2](#) log file (stdout): `t2 -r file.pcap | tawk -L`
- To convert the output to JSON: `tawk 'json($flowStat "\t" tuple5())' file_flows.txt`
- To convert the output to JSON: `tawk 'aggr(tuple2())' file_flows.txt | tawk 'json()'`
- To create a PCAP with all packets from flow 5: `tawk -x flow5.pcap '$flowInd == 5' file_flows.txt`
- To create a PCAP with packets 4-9: `tawk -P -x pkts-4_to_9.pcap 'packet("4-9")' file_packets.txt`
- To see all ICMP packets in Wireshark: `tawk -k 'icmp()' file_flows.txt`
- To see packet 4, 10 and 42 in Wireshark: `tawk -P -k 'packet("4;10;42")' file_packets.txt`

For a complete list of options, use the `-h` option.

Note that an option not recognized by `tawk` is internally passed to `awk/gawk`. One of the most useful is the `-v` option to set the value of a variable:

- Changing the output field separator:
`tawk -v OFS=',' '{ print $col1, $col2 }' file.txt`
- Passing a variable to `tawk`:
`tawk -v myvar=myvalue '{ print $col1, myvar }' file.txt`

For a complete list of options, run `awk -h`.

86.5 *-s and -N Options*

The `-s` option can be used to specify the starting character(s) of the row containing the column names (default: ‘%’). If several rows start with the specified character(s), then the last one is used as column names. To change this behavior, the line number can be specified as well with the help of the `-N` option. For example, if rows 1 to 5 start with ‘#’ and row 3 contains the column names, specify the separator as follows: `tawk -s '#' -N 3` If the row with column names does not start with a special character, use `-s ''`.

86.6 Related Utilities

86.6.1 awkf

Configure awk to use tabs, i.e., '\t' as input and output separator (prevent issue with repetitive values), e.g.,
`awkf '{ print $4 }' file_flows.txt`

86.6.2 lsx

Display columns with fixed width (default: 40), e.g., `lsx file_flows.txt` or `lsx 45 file_flows.txt`

86.6.3 sortu

Sort rows and count the number of times a given row appears, then sort by the most occurring rows. (Alias for `sort | uniq -c | sort -rn`). Useful, e.g., to analyze the most occurring user-agents: `tawk '{ print $httpUsrAg }' FILE_flows.txt | sortu`

sortup

Same as `sortu`, but display the relative percentage instead of the absolute count. e.g., to analyze the most occurring user-agents: `tawk '{ print $httpUsrAg }' FILE_flows.txt | sortup`

86.6.4 tcol

Display columns with minimum width, e.g., `tcol file_flows.txt`.

86.7 Functions

Collection of functions for `tawk`:

- Parameters between brackets are optional,
- IPs can be given as string ("1.2.3.4"), hexadecimal (0xffffffff) or int (4294967295),
- Network masks can be given as string ("255.255.255.0"), hexadecimal (0xffffffff00) or CIDR notation (24),
- Networks can be given as string, hexadecimal or int, e.g., "1.2.3.4/24" or "0x01020304/255.255.255.0",
- String functions can be made case insensitive by adding the suffix `i`, e.g., `streq → streqi`,
- Some examples are provided below,
- More details and examples can be found for every function by running `tawk -d funcname`.

Function	Description
<code>hdr ()</code>	Use this function in your tests to keep the header (column names).
<code>tuple2 ()</code>	Return the 2 tuple (source IP and destination IP).
<code>tuple3 ()</code>	Return the 3 tuple (source IP, destination IP and port).
<code>tuple4 ()</code>	Return the 4 tuple (source IP and port, destination IP and port).
<code>tuple5 ()</code>	Return the 5 tuple (source IP and port, destination IP and port, protocol).

Function	Description
tuple6()	Return the 6 tuple (source IP and port, dest. IP and port, proto, VLANID).
host([ip net])	Return true if the source or destination IP is equal to ip or belongs to net. If ip is omitted, return the source and destination IP.
shost([ip net])	Return true if the source IP is equal to ip or belongs to net. If ip is omitted, return the source IP.
dhost([ip net])	Return true if the destination IP is equal to ip or belongs to net. If ip is omitted, return the destination IP.
net([ip net])	Alias for host([ip net]).
snet([ip net])	Alias for shost([ip net]).
dnet([ip net])	Alias for dhost([ip net]).
loopback(ip)	Return true if ip is a loopback address.
mcast(ip)	Return true if ip is a multicast address.
privip(ip)	Return true if ip is a private IP.
port([p])	Return true if the source or destination port appears in p (comma or semicolon separated). Ranges may also be specified using a dash, e.g., port("1-3"). If p is omitted, return the source and destination port.
dport([p])	Return true if the destination port appears in p (comma or semicolon separated). Ranges may also be specified using a dash, e.g., dport("1-3"). If p is omitted, return the destination port.
sport([p])	Return true if the source port appears in p (comma or semicolon separated). Ranges may also be specified using a dash, e.g., sport("1-3"). If p is omitted, return the source port.
ip()	Return true if the flow contains IPv4 or IPv6 traffic.
ipv4()	Return true if the flow contains IPv4 traffic.
ipv6()	Return true if the flow contains IPv6 traffic.
proto([p])	Return true if the protocol number appears in p (comma or semicolon separated). Ranges may also be specified using a dash, e.g., proto("1-3"). If p is omitted, return the protocol number.
proto2str([p])	Return the string representation of the protocol number p. If p is omitted, return the string representation of the protocol.
icmp([p])	Return true if the protocol is equal to 1 (ICMP).
igmp([p])	Return true if the protocol is equal to 2 (IGMP).
tcp([p])	Return true if the protocol is equal to 6 (TCP).
udp([p])	Return true if the protocol is equal to 17 (UDP).
rsvp([p])	Return true if the protocol is equal to 46 (RSVP).
gre([p])	Return true if the protocol is equal to 47 (GRE).
esp([p])	Return true if the protocol is equal to 50 (ESP).
ah([p])	Return true if the protocol is equal to 51 (AH).
icmp6([p])	Return true if the protocol is equal to 58 (ICMPv6).

Function	Description
sctp([p])	Return true if the protocol is equal to 132 (SCTP).
dhcp()	Return true if the flow contains DHCP traffic.
dns()	Return true if the flow contains DNS traffic.
http()	Return true if the flow contains HTTP traffic.
tcpflags([val])	If val is specified, return true if the specified flags are set. If val is omitted, return a string representation of the TCP flags.
ip2num(ip)	Convert an IP address to a number.
ip2hex(ip)	Convert an IPv4 address to hex.
ip2str(ip)	Convert an IPv4 address to string.
ip6str(ip)	Convert an IPv6 address to string.
ip6compress(ip)	Compress an IPv6 address.
ip6expand(ip[,trim])	Expand an IPv6 address. If trim is different from 0, remove leading zeros.
ip2mask(ip)	Convert an IP address to a network mask (int).
mask2ip(m)	Convert a network mask (int) to an IPv4 address (int).
mask2ipstr(m)	Convert a network mask (int) to an IPv4 address (string).
mask2ip6(m)	Convert a network mask (int) to an IPv6 address (int).
mask2ip6str(m)	Convert a network mask (int) to an IPv6 address (string).
ipinnet(ip,net[,mask])	Test whether an IP address belongs to a given network.
ipinrange(ip,low,high)	Test whether an IP address lies between two addresses.
localtime(t)	Convert UNIX timestamp to string (localtime).
utc(t)	Convert UNIX timestamp to string (UTC).
timestamp(t)	Convert date to UNIX timestamp.
t2split(val,sep [,num[,osep]])	Split values according to sep. If num is omitted or 0, val is split into osep separated columns. If num > 0, return the num repetition. If num < 0, return the num repetition from the end, e.g., -1 for last element. Multiple num can be specified, e.g., "1;-1;2". Output separator osep, defaults to OFS. Split compound values. Alias for t2split(val, "_", num, osep). Split repetitive values. Alias for t2split(val, ";", num, osep).
splitc(val[,num[,osep]])	
splitr(val[,num[,osep]])	
valcontains(val,sep,item)	Return true if one item of val split by sep is equal to item.
cvalcontains(val,item)	Alias for valcontains(val, "_", item).
rvalcontains(val,item)	Alias for valcontains(val, ";", item).
strisempty(val)	Return true if val is an empty string.
streq(val1,val2)	Return true if val1 is equal to val2.

Function	Description
<code>strneq(val1, val2)</code>	Return true if <code>val1</code> and <code>val2</code> are not equal.
<code>hasprefix(val, pre)</code>	Return true if <code>val</code> begins with the prefix <code>pre</code> .
<code>hassuffix(val, suf)</code>	Return true if <code>val</code> finished with the suffix <code>suf</code> .
<code>contains(val, txt)</code>	Return true if <code>val</code> contains the substring <code>txt</code> .
<code>not(q)</code>	Return the logical negation of a query <code>q</code> . This function must be used to keep the header when negating a query.
<code>bfeq(val1, val2)</code>	Return true if the hexadecimal numbers <code>val1</code> and <code>val2</code> are equal.
<code>bitsallset(val, mask)</code>	Return true if all the bits set in <code>mask</code> are also set in <code>val</code> .
<code>bitsanyset(val, mask)</code>	Return true if one of the bits set in <code>mask</code> is also set in <code>val</code> .
<code>isset(v)</code>	Return true if <code>v</code> is set, i.e., not empty, false otherwise.
<code>isfloat(v)</code>	Return true if <code>v</code> is a floating point number.
<code>isint(v)</code>	Return true if <code>v</code> is an integer.
<code>isnum(v)</code>	Return true if <code>v</code> is a number (signed, unsigned or floating point).
<code>isuint(v)</code>	Return true if <code>v</code> is an unsigned integer.
<code>isip(v)</code>	Return true if <code>v</code> is an IPv4 address in hexadecimal, numerical or dotted decimal notation.
<code>isip6(v)</code>	Return true if <code>v</code> is an IPv6 address.
<code>isiphex(v)</code>	Return true if <code>v</code> is an IPv4 address in hexadecimal notation.
<code>isipnum(v)</code>	Return true if <code>v</code> is an IPv4 address in numerical (int) notation.
<code>isipstr(v)</code>	Return true if <code>v</code> is an IPv4 address in dotted decimal notation.
<code>join(a, s)</code>	Convert an array to string, separating each value with <code>s</code> .
<code>quote(s)</code>	Add leading and trailing quotes to a string <code>s</code> and escape all quotes in <code>s</code> .
<code>unquote(s)</code>	Remove leading and trailing quotes from a string <code>s</code> and unescape all escaped quotes in <code>s</code> .
<code>chomp(s)</code>	Remove leading and trailing spaces from a string <code>s</code> .
<code>strip(s)</code>	Remove leading and trailing spaces from a string <code>s</code> .
<code>lstrip(s)</code>	Remove leading spaces from a string <code>s</code> .
<code>rstrip(s)</code>	Remove trailing spaces from a string <code>s</code> .
<code>ientropy([num[, sc[, rev[, imin]]]])</code>	Compute the Shannon (information) entropy of each column. Set <code>imin</code> to filter out columns with low entropy ($\leq imin$). The result can be accessed with <code>get_entropy(c)</code> or printed with <code>print_entropy([c])</code> .
<code>mean(c)</code>	Compute the mean value of a column <code>c</code> . The result can be accessed with <code>get_mean(c)</code> or printed with <code>print_mean([c])</code> .
<code>min(c)</code>	Keep track of the min value of a column <code>c</code> . The result can be accessed with <code>get_min(c)</code> or printed with <code>print_min([c])</code> .
<code>max(c)</code>	Keep track of the max value of a column <code>c</code> . The result can be accessed with <code>get_max(c)</code> or

Function	Description
	printed with <code>print_max([c])</code> .
<code>abs(v)</code>	Return the absolute value of <code>v</code> .
<code>log2(n)</code>	Return the binary logarithm (log base 2) of <code>a</code> .
<code>max2(a,b)</code>	Return the maximum value between <code>a</code> and <code>b</code> .
<code>max3(a,b,c)</code>	Return the maximum value between <code>a</code> , <code>b</code> and <code>c</code> .
<code>min2(a,b)</code>	Return the minimum value between <code>a</code> and <code>b</code> .
<code>min3(a,b,c)</code>	Return the minimum value between <code>a</code> , <code>b</code> and <code>c</code> .
<code>aggr(fields[,val[,num]])</code>	Perform aggregation of <code>fields</code> and store the sum of <code>val</code> . <code>fields</code> and <code>val</code> can be tab separated lists of fields, e.g., <code>\$srcIP\t\$dstIP</code> . Results are sorted according to the first value of <code>val</code> . If <code>val</code> is omitted, the empty string or equal to "flows" or "packets" (case insensitive), count the number of records (flows or packets). If <code>num</code> is omitted or 0, return the full list. If <code>num > 0</code> return the top <code>num</code> results. If <code>num < 0</code> return the bottom <code>num</code> results.
<code>aggrep(fields[,val[,num[,ign_e[,sep]]]])</code>	Perform aggregation of the repetitive <code>fields</code> and store the sum of <code>val</code> . <code>val</code> can be a tab separated lists of fields, e.g., <code>\$17BytesSnt\t\$pktsSnt</code> . Results are sorted according to the first value of <code>val</code> . If <code>val</code> is omitted, the empty string or equal to "flows" or "packets" (case insensitive), count the number of records (flows or packets). If <code>num</code> is omitted or 0, return the full list. If <code>num > 0</code> return the top <code>num</code> results. If <code>num < 0</code> return the bottom <code>num</code> results. If <code>ign_e</code> is omitted or 0, consider all values, otherwise ignore empty values. <code>sep</code> can be used to change the separator character (default: ";").
<code>t2rsort(col[,num[,type]])</code>	Sort the file in reverse order according to <code>col</code> . (Multiple column numbers can be specified by using ";" as separator, e.g., 1 ";" 2) If <code>num</code> is omitted or 0, return the full list. If <code>num > 0</code> return the top <code>num</code> results. If <code>num < 0</code> return the bottom <code>num</code> results. <code>type</code> can be used to specify the type of data to sort: "ip", "num" or "str" (default is based on the first matching record).
<code>t2sort(col[,num[,type[,rev]]])</code>	Sort the file according to <code>col</code> . (Multiple column numbers can be specified by using ";" as separator, e.g., 1 ";" 2) If <code>num</code> is omitted or 0, return the full list. If <code>num > 0</code> return the top <code>num</code> results. If <code>num < 0</code> return the bottom <code>num</code> results. <code>type</code> can be used to specify the type of data to sort:

Function	Description
	"ip", "num" or "str" (default is based on the first matching record). If rev > 0, sort in reverse order (alternatively, use the t2rsort() function).
t2whois(ip[,o_opt])	Wrapper to call t2whois from tawk. ip must be a valid IPv4/6 address. o_opt is passed verbatim to t2whois -o option (run t2whois -L for more details).
wildcard(expr)	Print all columns whose name matches the regular expression expr. If expr is preceded by an exclamation mark, return all columns whose name does NOT match expr.
hrnum(num[,mode[,suffix]])	Convert the number num to human readable form.
hrtime(secs[,mode[,unit]])	Convert the timestamp (seconds) secs to human readable form.
json([s])	Convert the string s to JSON. The first record is used as column names. If s is omitted, convert the entire row.
texscape(s)	Escape the string s to make it LaTeX compatible.
bitshift(n,t[,d[,b]])	Shift a byte or of a list of bytes n to the left or right by a given number of bits t. To shift to the left, set d to 0 (default), to shift to the right set d ≠ 0. Set b to 16 to force interpretation as hexadecimal, e.g., interpret 45 as 69 (0x45) instead of 45.
nibble_swap(n[,b])	Swap the nibbles of a byte or of a list of bytes n. Set b to 16 to force interpretation as hexadecimal, e.g., interpret 45 as 69 (0x45) instead of 45.
tobits(u,[b])	Convert the unsigned integer u to its binary representation. Set b to 16 to force interpretation as hexadecimal, e.g., interpret 45 as 69 (0x45) instead of 45.
base64(s)	Encode a string s as base64.
base64d(s)	Decode a base64 encoded string s.
urldecode(url)	Decode the encoded URL url.
printbold(s, n)	Print the string s in bold with an added newline. If n is set, the trailing newline is omitted.
printerr(s, n)	Print the string s in red to stderr. If n is set, the trailing newline is omitted.
printfn(s, n)	Print the string s in blue. If n is set, the trailing newline is omitted.
printok(s, n)	Print the string s in green. If n is set, the trailing newline is omitted.
printwrn(s, n)	Print the string s in orange. If n is set, the trailing newline is omitted.
diff(file[,mode])	Compare two files (file and the input), and print the name and number of the columns which differ. The mode parameter can be used to control the

Function	Description
	format of the output.
ffsplit([s[,k[,h]]])	Split the input file into smaller more manageable files. The files to create can be specified as argument to the function (one comma separated string). If no argument is specified, create one file per column whose name ends with Stat, e.g., dnsStat, and one for pwxType (pw) and covertChannels (cc). If $k > 0$, then only print relevant fields and those controlled by h , a comma separated list of fields to keep in each file, e.g., "srcIP,dstIP".
flow([f])	Return all flows whose index appears in f (comma or semicolon separated). Ranges may also be specified using a dash, e.g., flow("1-3"). If f is omitted, return the flow index.
packet([p])	Return all packets whose number appears in p (comma or semicolon separated). Ranges may also be specified using a dash, e.g., packet ("1-3"). If p is omitted, return the packet number.
follow_stream(f[,of[,d[,pf[,r[,nc]]]])	Return the payload of the flow with index f . of can be used to change the output format [default: 0]: 0: Payload only, 1: prefix each payload with packet/flow info, 2: JSON, 3: Reconstruct (pipe the output to xxd -p -r to reproduce the binary file). d can be used to only extract a specific direction ("A" or "B") [default: "" (A and B)]. pf can be used to change the payload format [default: 0]: 0: ASCII, 1: Hexdump, 2: Raw/Binary, 3: Base64. r can be used to prevent the analysis of TCP sequence numbers (no TCP reassembly and reordering). nc can be used to print the data without colors.
shark(q)	Query flow files according to Wireshark's syntax.

86.8 Examples

Collection of examples using tawk functions:

Function	Description
covertChans([val[,num]])	Return information about hosts possibly involved in a covert channels. If val is omitted or equal to "flows", count the number of flows. Otherwise, sum up the values of val .

Function	Description
	If num is omitted or 0, return the full list, If num > 0 return the top num results, If num < 0 return the bottom num results.
dnsZT()	Return all flows where a DNS zone transfer was performed.
exeDL([n])	Return the top N EXE downloads.
httpHostsURL([f])	Return all HTTP hosts and a list of the files hosted (sorted alphabetically). If f > 0, print the number of times a URL was requested.
nonstdports()	Return all flows running protocols over non-standard ports.
passivedns()	Extract all DNS server replies from a flow file. The following information is reported for each reply: FirstSeen, LastSeen, Type (A or AAAA), TTL, Query, Answer, Organization, Country, AS number
passwords([val[,num]])	Return information about hosts sending authentication in cleartext. If val is omitted or equal to "flows", count the number of flows. Otherwise, sum up the values of val. If num is omitted or 0, returns the full list. If num > 0 return the top num results. If num < 0 return the bottom num results.
postQryStr([n])	Return the top N POST requests with query strings.
ssh()	Return the SSH connections.
topDnsA([n])	Return the top N DNS answers.
topDnsIp4([n])	Return the top N DNS answers IPv4 addresses.
topDnsIp6([n])	Return the top N DNS answers IPv6 addresses.
topDnsQ([n])	Return the top N DNS queries.
topHttpMimesST([n])	Return the top HTTP content-type (type/subtype).
topHttpMimesT([n])	Return the top HTTP content-type (type only).
topSLD([n])	Return the top N second-level domains queried (google.com, yahoo.com, ...).
topTLD([n])	Return the top N top-level domains (TLD) queried (.com, .net, ...).

86.9 t2nfdump

Collection of functions for tawk allowing access to specific fields using a syntax similar as nfdump.

Function	Description
ts()	Start Time — first seen
te()	End Time — last seen
td()	Duration
pr()	Protocol
sa()	Source Address
da()	Destination Address
sap()	Source Address:Port
dap()	Destination Address:Port
sp()	Source Port
dp()	Destination Port
pkt()	Packets — default input
ipkt()	Input Packets
opkt()	Output Packets
byt()	Bytes — default input
ibyt()	Input Bytes
obyt()	Output Bytes
flg()	TCP Flags
mpls1()	MPLS label 1
mpls2()	MPLS label 2
mpls3()	MPLS label 3
mpls4()	MPLS label 4
mpls5()	MPLS label 5
mpls6()	MPLS label 6
mpls7()	MPLS label 7
mpls8()	MPLS label 8
mpls9()	MPLS label 9
mpls10()	MPLS label 10
mpls()	MPLS labels 1–10
bps()	Bits per second
pps()	Packets per second
bpp()	Bytes per packet
oline()	nfdump line output format (-o line)
olong()	nfdump long output format (-o long)
oextended()	nfdump extended output format (-o extended)

86.10 t2custom

Copy your own functions in this folder. Refer to Section 86.11 for more details on how to write a tawk function. To have your functions automatically loaded, include them in the file `t2custom/t2custom.load`.

86.11 Writing a tawk Function

- Ideally one function per file (where the filename is the name of the function)
- Private functions are prefixed with an underscore

- Always declare local variables 8 spaces after the function arguments
- Local variables are prefixed with an underscore
- Use uppercase letters and two leading and two trailing underscores for global variables
- Include all referenced functions
- Files should be structured as follows:

```
#!/usr/bin/env awk
#
# Function description
#
# Parameters:
#   - arg1: description
#   - arg2: description (optional)
#
# Dependencies:
#   - plugin1
#   - plugin2 (optional)
#
# Examples:
#   - tawk 'funcname()' file.txt
#   - tawk '{ print funcname() }' file.txt

@include "hdr"
@include "_validate_col"

function funcname(arg1, arg2, [8 spaces] _locvar1, _locvar2) {
    _locvar1 = _validate_col("colname1;altcolname1", _my_colname1)
    _validate_col("colname2")

    if (hdr()) {
        if (__PRIHDR__) print "header"
    } else {
        print "something", $_locvar1, $colname2
    }
}
```

86.12 Using tawk Within Scripts

To use tawk from within a script:

1. Create a TAWK variable pointing to the script: TAWK="\$T2HOME/scripts/tawk/tawk"
2. Call tawk as follows: \$TAWK 'dport(80)' file.txt

86.13 Using tawk With Non-Tralyzer Files

tawk can also be used with files which were not produced by Tralyzer.

- The input field separator can be specified with the `-F` option, e.g., `tawk -F ',' 'program' file.csv`
- The row listing the column names, can start with any character specified with the `-s` option, e.g., `tawk -s '#' 'program' file.txt`
- All the column names must not be equal to a function or builtin name
- Valid column names must start with a letter (a-z, A-Z) and can be followed by any number of alphanumeric characters or underscores
- If the column names are different from those used by Tralyzer, refer to Section 86.13.1.

86.13.1 Mapping External Column Names to Tralyzer Column Names

If the column names are different from those used by Tralyzer, a mapping between the different names can be made in the file `my_vars`. The format of the file is as follows:

```
BEGIN {
    _my_srcIP = non_t2_name_for_srcIP
    _my_dstIP = non_t2_name_for_dstIP
    ...
}
```

Once edited, run tawk with the `-i $T2HOME/scripts/tawk/my_vars` option and the external column names will be automatically used by tawk functions, such as `tuple2()`. For more details, refer to the `my_vars` file.

86.13.2 Using tawk with Bro/Zeek Files

To use tawk with Bro/Zeek log files, use one of `--bro` or `--zeek` option:

```
tawk --bro '{ program }' file.log tawk --zeek '{ program }' file.log
```

86.14 Awk Cheat Sheet

- Tralyzer flow files default field separator is '\t':
 - Always use `awk -F '\t'` (or `awkf/tawk`) when working with flow files.
- Load libraries, e.g., tawk functions, with `-i: awk -i file.awk 'program' file.txt`
- Always use `strtonum` with hex numbers (bitfields)
- Awk indices start at 1
- Using tawk is recommended.

86.14.1 Useful Variables

- \$0: entire line
- \$1, \$2, ..., \$NF: column 1, 2, ...
- FS: field separator
- OFS: output field separator
- ORS: output record separator
- NF: number of fields (columns)
- NR: record (line) number
- FNR: record (line) number relative to the current file
- FILENAME: name of current file
- To use external variables, use the -v option, e.g., awk -v name="value" '{ print name }' file.txt.

86.14.2 Awk Program Structure

```
awk -F'\t' -i min -v OFS='\t' -v h=$(hostname) '
BEGIN { a = 0; b = 0; }          # Called once at the beginning
/^A/ { a++ }                   # Called for every row starting with char A
/^B/ { b++ }                   # Called for every row starting with char B
{ c++ }                        # Called for every row
END   { print h, min(a, b), c } # Called once at the end
' file.txt
```

86.15 Awk Templates

- Print the whole line:

- tawk '{ print }' file.txt
- tawk '{ print \$0 }' file.txt
- tawk 'FILTER' file.txt
- tawk 'FILTER { print }' file.txt
- tawk 'FILTER { print \$0 }' file.txt

- Print selected columns only:

- tawk '{ print \$srcIP, \$dstIP }' file.txt
- tawk '{ print \$1, \$2 }' file.txt
- tawk '{ print \$4 "\t" \$6 }' file.txt

```
- tawk '{
    for (i = 6; i < NF; i++) {
        printf "%s\t", $i
    }
    printf "%s\n", $NF
}' file.txt
```

- Keep the column names:

```
- tawk 'hdr() || FILTER' file.txt
- awkf 'NR == 1 || FILTER' file.txt
- awkf '/^%/ || FILTER' file.txt
- awkf '/^%[:space:]*[:alpha:][[:alnum:]_]*$/ || FILTER' file.txt
```

- Skip the column names:

```
- tawk '!hdr() && FILTER' file.txt
- awkf 'NR > 1 && FILTER' file.txt
- awkf '!/^%/ && FILTER' file.txt
- awkf '!/^%[:space:]*[:alpha:][[:alnum:]_]*$/ && FILTER' file.txt
```

- Bitfields and hexadecimal numbers:

```
- tawk 'bfeq($3,0)' file.txt
- awkf 'strtonum($3) == 0' file.txt
- tawk 'bitsanyset($3,1)' file.txt
- tawk 'bitsallset($3,0x81)' file.txt
- awkf 'and(strtonum($3), 0x1)' file.txt
```

- Split compound values:

```
- tawk '{ print splitc($16, 1) }' file.txt # first element
- tawk '{ print splitc($16, -1) }' file.txt # last element
- awkf '{ split($16, A, "_"); print A[1] }' file.txt
- awkf '{ n = split($16, A, "_"); print A[n] }' file.txt # last element
- tawk '{ print splitc($16) }' file.txt
- awkf '{ split($16, A, "_"); for (i=1;i<=length(A);i++) print A[i] }' file.txt
```

- Split repetitive values:

```
- tawk '{ print splitr($16, 3) }' file.txt # third repetition
- tawk '{ print splitr($16, -2) }' file.txt # second to last repetition
- awkf '{ split($16, A, ";"); print A[3] }' file.txt
- awkf '{ n = split($16, A, ";"); print A[n] }' file.txt # last repetition
- tawk '{ print splitr($16) }' file.txt
```

- awkf '{ split(\$16, A, ";"); for (i=1;i<=length(A);i++) print A[i] }' file.txt
- Filter out empty strings:
 - tawk '!strisempty(\$4)' file.txt
 - awkf '! (length(\$4) == 0 || \$4 == "\\"\\")' file.txt
- Compare strings (case sensitive):
 - tawk 'streq(\$3,\$4)' file.txt
 - awkf '\$3 == \$4' file.txt
 - awkf '\$3 == \"text\"' file.txt
- Compare strings (case insensitive):
 - tawk 'streqi(\$3,\$4)' file.txt
 - awkf 'tolower(\$3) == tolower(\$4)' file.txt
- Use regular expressions on specific columns:
 - awkf '\$8 ~ /¹92.168.1.[0-9]{1,3}\$/' file.txt # print matching rows
 - awkf '\$8 !~ /¹92.168.1.[0-9]{1,3}\$/' file.txt # print non-matching rows
- Use column names in awk:
 - tawk '{ print \$srcIP, \$dstIP }' file.txt
 - awkf '
 NR == 1 {
 for (i = 1; i <= NF; i++) {
 if (\$i == "srcIP") srcIP = i
 else if (\$i == "dstIP") dstIP = i
 }
 if (srcIP == 0 || dstIP == 0) {
 print "No column with name srcIP and/or dstIP"
 exit
 }
 }
 NR > 1 {
 print \$srcIP, \$dstIP
 }
 ' file.txt
 - awkf '
 NR == 1 {
 for (i = 1; i <= NF; i++) {
 col[\$i] = i
 }
 }
 NR > 1 {
 print \$col["srcIP"], \$col["dstIP"];
 }
 ' file.txt

86.16 Examples

1. Pivoting (variant 1):

- (a) First extract an attribute of interest, e.g., an unresolved IP address in the `Host:` field of the HTTP header:

```
tawk 'aggr($httpHosts)' FILE_flows.txt | tawk '{ print unquote($1); exit }'
```

- (b) Then, put the result of the last command in the `badguy` variable and use it to extract flows involving this IP:

```
tawk -v badguy="$(!!)" 'host(badguy)' FILE_flows.txt
```

2. Pivoting (variant 2):

- (a) First extract an attribute of interest, e.g., an unresolved IP address in the `Host:` field of the HTTP header, and store it into a `badip` variable:

```
badip=$(tawk 'aggr($httpHosts)' FILE_flows.txt | tawk '{ print unquote($1);exit }')
```

- (b) Then, use the `badip` variable to extract flows involving this IP:

```
tawk -v badguy="$badip" 'host(badguy)' FILE_flows.txt
```

3. Aggregate the number of bytes sent between source and destination addresses (independent of the protocol and port) and output the top 10 results:

```
tawk 'aggr($srcIP "\t" $dstIP, $l7BytesSnt, 10)' FILE_flows.txt
```

```
tawk 'aggr(tuple2(), $l7BytesSnt "\t" "Flows", 10)' FILE_flows.txt
```

4. Sort the flow file according to the duration (longest flows first) and output the top 5 results:

```
tawk 't2sort(duration, 5)' FILE_flows.txt
```

5. Extract all TCP flows while keeping the header (column names):

```
tawk 'hdr() || tcp()' FILE_flows.txt
```

6. Extract all flows whose destination port is between 6000 and 6008 (included):

```
tawk 'dport("6000-6008")' FILE_flows.txt
```

7. Extract all flows whose destination port is 53, 80 or 8080:

```
tawk 'dport("53;80;8080")' FILE_flows.txt
```

8. Extract all flows whose source IP is in subnet 192.168.1.0/24 (using `host` or `net`):

```
tawk 'shost("192.168.1.0/24")' FILE_flows.txt
```

```
tawk 'snet("192.168.1.0/24")' FILE_flows.txt
```

9. Extract all flows whose source IP is in subnet 192.168.1.0/24 (using `ipinrange`):

```
tawk 'ipinrange($srcIP, "192.168.1.0", "192.168.1.255")' FILE_flows.txt
```

10. Extract all flows whose source IP is in subnet 192.168.1.0/24 (using ipinnet):

```
tawk 'ipinnet($srcIP, "192.168.1.0", "255.255.255.0")' FILE_flows.txt
```

11. Extract all flows whose source IP is in subnet 192.168.1.0/24 (using ipinnet and a hex mask):

```
tawk 'ipinnet($srcIP, "192.168.1.0", 0xffffffff00)' FILE_flows.txt
```

12. Extract all flows whose source IP is in subnet 192.168.1.0/24 (using ipinnet and the CIDR notation):

```
tawk 'ipinnet($srcIP, "192.168.1.0/24")' FILE_flows.txt
```

13. Extract all flows whose source IP is in subnet 192.168.1.0/24 (using ipinnet and a CIDR mask):

```
tawk 'ipinnet($srcIP, "192.168.1.0", 24)' FILE_flows.txt
```

For more examples, refer to tawk -d option, e.g., tawk -d aggr, where every function is documented and comes with a set of examples. The complete documentation can be consulted by running tawk -d all.

86.17 FAQ

86.17.1 Can I use tawk with non Tranalyzer files?

Yes, refer to Section 86.13.

86.17.2 Can I use tawk functions with non Tranalyzer column names?

Yes, edit the my_vars file and load it using -i \$T2HOME/scripts/tawk/my_vars option. Refer to Section 86.13.1 for more details.

86.17.3 Can I use tawk with files without column names?

Yes, but you won't be able to use the functions which require a specific column, e.g., host().

86.17.4 The row listing the column names start with a '#' instead of a '%'...Can I still use tawk?

Yes, use the -s option to specify the first character, e.g., tawk -s '#' 'program'

86.17.5 Can I process Bro/Zeek log files with tawk?

Yes, use the --zeek option.

86.17.6 Can I process a CSV (Comma Separated Value) file with tawk?

The simplest way to process CSV files is to use the --csv option. This sets the input and output separators to a comma and considers the first row to be the column names.

```
tawk --csv 'program' file.csv
```

Alternatively, the input field separator can be changed with the -F option and the output separator with -O ',' or -v OFS=','. Note that tawk expects the column names to be the last row starting with a '%'. This can be changed with the -s and -N options (Section 86.5).

```
tawk -F ',' -v OFS=',' -s "" -N 1 'program' file.csv
```

86.17.7 Can I produce a CSV (Comma Separated Value) file from tawk?

The output field separator (OFS) can be changed with the `-O 'fs'` or `-v OFS='fs'` option. To produce a CSV file, run tawk as follows: `tawk -O ',' 'program' file.txt` or `tawk -v OFS=',' 'program' file.txt`

86.17.8 Can I write my tawk programs in a file instead of the command line?

Yes, copy the program (without the single quotes) in a file, e.g., `prog.txt` and run it as follows:
`tawk -f prog.txt file.txt`

86.17.9 Can I still use column names if I pipe data into tawk?

Yes, you can specify a file containing the column names with the `-I` option as follows:
`cat file.txt | tawk -I colnames.txt 'program'`

86.17.10 Can I use tawk if the row with the column names does not start with a special character?

Yes, you can specify the empty character with `-s ""`. Refer to Section [86.5](#) for more details.

86.17.11 I get a list of syntax errors from gawk... What is the problem?

The name of the columns is used to create variable names. If it contains forbidden characters, then an error similar to the following is reported.

```
gawk: /tmp/fileBndhdf:3: col-name = 3
gawk: /tmp/fileBndhdf:3:           ^ syntax error
```

Although tawk will try to replace forbidden characters with underscore, the best practice is to use only alphanumeric characters (A-Z, a-z, 0-9) and underscore as column names. Note that a column name **MUST NOT** start with a number.

86.17.12 I get a function name previously defined error from gawk... What is the problem?

The name of the columns is used to create variable names. If a column is named after a tawk function or a builtin, then an error similar to the following is reported.

```
gawk: In file included from ah:21,
gawk:                     from /home/user/tranalyzer2/scripts/tawk/funcs/funcs.load:8,
gawk: proto:36: error: function name 'proto' previously defined
```

In this case, you have two options. Either rename the column(s) in your file, e.g., `proto → 14Proto` or use `tawk -t` option. With the `-t` option, Tawk tries to validate the column names by ensuring that no column names is equal to a function name and that all column names used in the program exist. Note that this verification process can be slow.

86.17.13 Tawk cannot find the column names... What is the problem?

First, make sure the comment char (`-s` option) is correctly set for your file (the default is `'%'`). Second, make sure the column names do not contain forbidden characters, i.e., use only alphanumeric and underscore and do not start with a number. If the row with column names is not the last one to start with the separator character, then specify the line number with the `-N` option as follows: `tawk -N 3'` or `tawk -s '#' -N 2`. Refer to Section [86.5](#) for more details.

86.17.14 Wireshark refuses to open PCAP files generated with tawk -k option...

If Wireshark displays the message Couldn't run /usr/bin/dumpcap in child process: Permission Denied., then this means that your user does not belong to the wireshark group. To fix this issue, simply run the following command sudo gpasswd -a YOUR_USERNAME wireshark (you will then need to log off and on again).

86.17.15 Tawk reports errors similar to free(): double free detected in tcache 2

Tawk uses gawk -M option to handle IPv6 addresses. For some reasons, this option is regularly affected by bugs... If you do not need IPv6 support, you can simply comment out line 653 in tawk:

```
OPTS=(  
    #-M -v PREC=256          # <-- Add the leading sharp ('#') here  
    -v __PRIHDR__=$PRIHDR  
    -v __UNAME__="$(_uname)"  
)
```

86.17.16 Tawk -k reports errors similar to Couldn't run dumpcap in child process: Permission denied

On some Linux distributions, capturing packets is only allowed as root or as a member of the wireshark group.

Run the following command to add yourself to the wireshark group.

```
$ sudo usermod -a -G wireshark $USER
```

Then, log out and log in again (or reboot) and the problem should be fixed!

More information on the [Wireshark wiki](#).

A Creating a Custom Plugin

A plugin is a shared library file comprising of special functionality. Tranalyzer2 dynamically loads these shared libraries at runtime from the `~/.tranalyzer/plugins` directory in the user's home folder. Therefore Tranalyzer2 is available for users if being installed in the `/usr/local/bin` directory while the plugins are user dependent. To develop a plugin it is strongly recommended that the user utilizes our special “t2plugin“ script. This script uses the plugin skeleton “t2PSkel” to create a new custom named plugin. It is available in the `scripts/` folder. The script copies only the required files. Therefore it is recommended to upload the newly created folder to a SVN/GIT repository before running `./autogen.sh` (alternatively, `./autogen.sh -c` can be used to clean up automatically generated files that should not be committed). The skeleton contains a header and a source file comprising of all mandatory and optional functions as well as a small HOWTO file and a script to build and move a shared library to the plugins folder.

A.1 Plugin Name

Plugin names should be kept short, start with a lowercase letter and only contain characters in the following ranges: a–z, A–Z, 0–9, _. In addition, each “word” should start with an uppercase letter, e.g., `pluginName`.

A.2 Plugin Number

The plugin number (or order) influences when a plugin is to be loaded (useful if a plugin depends on another one). This number should consist of three digits and be unique. The plugin orders used in your Tranalyzer installation can be listed with `$T2HOME/doc/list_plugin_numbers.sh` or `t2plugin -l`. As a rule of thumb, numbers greater than 900 should be kept for sink (output) plugins and numbers smaller than 10 for global plugins.

plugin range	description
000 – 099	Global
100 – 199	Basic L2/3/4 plugins
200 – 299	Service and routing
300 – 699	L7 protocols
700 – 799	Math and statistics
800 – 899	Classifier and AI
900 – 999	Output (sink)

A.3 Plugin Creation

To create a new plugin named `pluginName` with plugin order 123, run the following command from Tranalyzer's root folder:

```
./scripts/t2plugin -c pluginName -n 123
```

If no plugin number is provided, then the script will choose a random one that is not used by any other plugin.

A C++ plugin can be created by passing the additional ‘-cpp’ option:

```
./scripts/t2plugin -c pluginName -n 123 -cpp
```

A.3.1 autogen.sh

The `autogen.sh` script provides the `EXTRAFILES` variable, which is used to list extra files, such as lists of subnets, protocols, services, databases or blacklists, that the plugin needs in order to run. The files listed in this variable are automatically copied into the Tranalyzer plugin folder.

```
EXTRAFILES=(dependency1 dependency2)
```

The `CFLAGS` variable in `autogen.sh` can be used if a plugin requires specific libraries, compilation or linking flags, e.g., `CFLAGS="-lzip"`. In such a case, the `DEPS` variable can be used to list the dependencies, e.g., `DEPS="libzip"`.

A.4 Compilation

The plugin can then be compiled by typing `./autogen.sh`. For a complete list of options, run `./autogen.sh -h`

A.5 Plugin Structure

All plugins have the same global structures, namely, a comment describing the license of the plugin, e.g., GPLv2+, some includes, followed by the declaration of variables and functions. This section discusses the Tranalyzer callbacks which follows the elements already mentioned. Note that all the callbacks are optional, but a plugin **MUST** call one of the initialization macros.

First, a plugin **MUST have one the following declarations:**

- `T2_PLUGIN_INIT(name, version, t2_v_major, t2_v_minor)`
- `T2_PLUGIN_INIT_WITH_DEPS(name, version, t2_v_major, t2_v_minor, deps)`

For example, to initialize `myPlugin`:

```
T2_PLUGIN_INIT_WITH_DEPS("myPlugin", "0.9.3", 0, 9, "tcpFlags,basicStats")
```

The available callbacks are:

- `void t2Init()`
- `binary_value_t *t2PrintHeader()`
- `void t2OnNewFlow(packet_t *packet, unsigned long flowIndex)`
- `void t2OnLayer2(packet_t *packet, unsigned long flowIndex)`
- `void t2OnLayer4(packet_t *packet, unsigned long flowIndex)`
- `void t2OnFlowTerminate(unsigned long flowIndex, outputBuffer_t *buf)`
- `void t2PluginReport(FILE *stream)`
- `void t2Finalize()`
- `void t2BufferToSink(outputBuffer_t *buffer, binary_value_t *bv)` [Sink (output) plugins only]

The following callbacks offer more advanced capabilities:

- `void t2BusCallback(uint32_t status)` [Not implemented]
- `void t2Monitoring(FILE *stream, uint8_t state)`
- `void t2SaveState(FILE *stream)`
- `void t2RestoreState(char *str)`

A.5.1 `void t2Init()`

This function is called before processing any packet.

A.5.2 `binary_value_t *t2PrintHeader()`

This function is used to describe the columns output by the plugin. Refer to Section A.8 and the `BV_APPEND` macros.

A.5.3 `void t2OnNewFlow(packet_t *packet, unsigned long flowIndex)`

This function is called every time a new flow is created.

A.5.4 `void t2OnLayer2(packet_t *packet, unsigned long flowIndex)`

This function is called for every packet with a layer 2. If `flowIndex` is `HASHTABLE_ENTRY_NOT_FOUND`, this means the packet also has a layer 4 and thus a call to `t2OnLayer4()` will follow.

A.5.5 `void t2OnLayer4(packet_t *packet, unsigned long flowIndex)`

This function is called for every packet with a layer 4.

A.5.6 `void t2OnFlowTerminate(unsigned long flowIndex, outputBuffer_t *buf)`

This function is called once a flow is terminated. Output all the statistics for the flow here. Refer to Section A.8 and the `OUTBUF_APPEND` macros.

A.5.7 `void t2BusCallback(uint32_t status)`

Currently not implemented.

A.5.8 `void t2Monitoring(FILE *stream, uint8_t state)`

This function is used to report information regarding the plugin at regular interval or when a `USR1` signal is received. `state` can be one of the following:

- `T2_MON_PRI_HDR`: a header (value names) must be printed
- `T2_MON_PRI_VAL`: the actual data must be printed
- `T2_MON_PRI_REPORT`: a report (similar to the plugin report) must be printed

A.5.9 void t2PluginReport(FILE *stream)

This function is used to report information regarding the plugin. This will appear in the final report.

A.5.10 void t2Finalize()

This function is called once all the packets have been processed. Cleanup all used memory here.

A.5.11 void t2SaveState(FILE *stream)

This function is used to save the state of the plugin. Tranalyzer can then restore the state in a future execution.

A.5.12 void t2RestoreState(char *str)

This function is used to restore the state of the plugin. str represents the line written in t2SaveState().

A.5.13 void t2BufferToSink(outputBuffer_t *buffer, binary_value_t *bv)

This callback is only required for sink (output) plugins.

A.6 Error, warning, and informational messages

Tranalyzer2 provides several macros to report errors, warnings, information or simple messages:

T2_PLOG()	print a normal message (standard terminal colors)	pluginName: message
T2_PINF()	print an information message (blue)	[INF] pluginName: message
T2_POK()	print an okay message (green)	[OK] pluginName: message
T2_PWRN()	print a warning message (yellow)	[WRN] pluginName: message
T2_PERR()	print an error message (red)	[ERR] pluginName: message

Note that T2_PERR always prints to stderr, while the other macros print to stdout or PREFIX_log.txt if Tranalyzer -l option was used.

Their usage is straightforward:

```
T2_PLOG("pluginName", "message %d", 42);
```

Note that a trailing newline is automatically added.

A.7 Naming Conventions

In order to avoid conflicts with other plugins, prefix all your macros and publicly available functions and variables with the plugin name or a few letters uniquely identifying the plugin.

A.8 Generating Output

The following macros can be used to declare and append new columns to the output buffer. The `BV_APPEND_*` macros are used to declare a new column with a given name, description desc and type. The `OUTBUF_APPEND_*` macros are used to append a value val of the given type to the buffer buf.

BV Macro	Type	Corresponding OUBUF Macro
Unsigned values		
<code>BV_APPEND_U8</code> (bv, name, desc)	<code>bt_uint_8</code>	<code>OUTBUF_APPEND_U8</code> (buf, val)
<code>BV_APPEND_U16</code> (bv, name, desc)	<code>bt_uint_16</code>	<code>OUTBUF_APPEND_U16</code> (buf, val)
<code>BV_APPEND_U32</code> (bv, name, desc)	<code>bt_uint_32</code>	<code>OUTBUF_APPEND_U32</code> (buf, val)
<code>BV_APPEND_U64</code> (bv, name, desc)	<code>bt_uint_64</code>	<code>OUTBUF_APPEND_U64</code> (buf, val)
<code>BV_APPEND_H8</code> (bv, name, desc)	<code>bt_hex_8</code>	<code>OUTBUF_APPEND_H8</code> (buf, val)
<code>BV_APPEND_H16</code> (bv, name, desc)	<code>bt_hex_16</code>	<code>OUTBUF_APPEND_H16</code> (buf, val)
<code>BV_APPEND_H32</code> (bv, name, desc)	<code>bt_hex_32</code>	<code>OUTBUF_APPEND_H32</code> (buf, val)
<code>BV_APPEND_H64</code> (bv, name, desc)	<code>bt_hex_64</code>	<code>OUTBUF_APPEND_H64</code> (buf, val)
Signed values		
<code>BV_APPEND_I8</code> (bv, name, desc)	<code>bt_int_8</code>	<code>OUTBUF_APPEND_I8</code> (buf, val)
<code>BV_APPEND_I16</code> (bv, name, desc)	<code>bt_int_16</code>	<code>OUTBUF_APPEND_I16</code> (buf, val)
<code>BV_APPEND_I32</code> (bv, name, desc)	<code>bt_int_32</code>	<code>OUTBUF_APPEND_I32</code> (buf, val)
<code>BV_APPEND_I64</code> (bv, name, desc)	<code>bt_int_64</code>	<code>OUTBUF_APPEND_I64</code> (buf, val)
Floating points values		
<code>BV_APPEND_FLT</code> (bv, name, desc)	<code>bt_float</code>	<code>OUTBUF_APPEND_FLT</code> (buf, val)
<code>BV_APPEND_DBL</code> (bv, name, desc)	<code>bt_double</code>	<code>OUTBUF_APPEND_DBL</code> (buf, val)
String values		
<code>BV_APPEND_STR</code> (bv, name, desc)	<code>bt_string</code>	<code>OUTBUF_APPEND_STR</code> (buf, val)
<code>BV_APPEND_STRC</code> (bv, name, desc)	<code>bt_string_class</code>	<code>OUTBUF_APPEND_STR</code> (buf, val)
Time values (timestamp and duration)⁶⁹		
<code>BV_APPEND_TIMESTAMP</code> (bv, name, desc)	<code>bt_timestamp</code>	<code>OUTBUF_APPEND_TIME</code> (buf, sec, usec)
<code>BV_APPEND_DURATION</code> (bv, name, desc)	<code>bt_duration</code>	<code>OUTBUF_APPEND_TIME</code> (buf, sec, usec)
IP values (network order)		
<code>BV_APPEND_IP4</code> (bv, name, desc)	<code>bt_ip4_addr</code>	<code>OUTBUF_APPEND_IP4</code> (buf, val)
<code>BV_APPEND_IP6</code> (bv, name, desc)	<code>bt_ip6_addr</code>	<code>OUTBUF_APPEND_IP6</code> (buf, val)

⁶⁹Time values use an `uint64` for the seconds and an `uint32` for the micro-seconds

BV_APPEND_IPX(bv, name, desc)	bt_ipx_addr	OUTBUF_APPEND_IPX(buf, version, val) ⁷⁰
-------------------------------	-------------	--

If more flexibility is required the following macros can be used:

- BV_APPEND(bv, name, desc, num_val, type1, type2, ...)
- OUTBUF_APPEND(buf, val, size)

A.8.1 Repetitive Values

A repetitive value consists of a `uint32` representing the number of repetitions, followed by the actual repetitions.

All the `BV_APPEND` macros introduced in the previous section can be suffixed with `_R` to represent a repetitive value:

`BV_APPEND_U8(bv, name, desc)` (non-repetitive) \Rightarrow `BV_APPEND_U8_R(bv, name, desc)` (repetitive).

In addition, the following `OUTBUF` macros are available for repetitive values:

OUTBUF Macro	Description	Type
<code>OUTBUF_APPEND_OPTSTR(buf, val)</code>	If <code>val</code> is NULL or empty, appends 0 (<code>uint32</code>) else appends 1 (<code>uint32</code>) and the string	<code>bt_string</code> , <code>bt_string_class</code>
<code>OUTBUF_APPEND_NUMREP(buf, reps)</code>	Appends the number of repetitions (<code>uint32</code>) ⁷¹	

A.8.2 Column Names

Column names should be kept short and only contain characters in the following ranges: `_`, `a-z`, `A-Z`, `0-9`. In addition, each “word” should start with an uppercase letter, e.g., `myCol2`. The `'_'` character should be used to name compound values, e.g., `field1_field2`. A good practice is to prefix each column name with the short name of the plugin, e.g., `ftpDecode` \rightarrow `ftpStat`, `ftpCNum`.

A.8.3 More Complex Output

Refer to Section A.8.

A.9 Accessible structures

Due to practical reasons all plugins are able to access every structure of the main program and the other plugins. This is indeed a security risk, but since Tranalyzer2 is a tool for practitioners and scientists in access limited environments the maximum possible freedom of the programmer is more important for us.

⁷⁰Appends the IP version (`uint8`), followed by the IP. If `version` is 6, then calls `OUTBUF_APPEND_IP6(buf, val.I Pv6.s6_addr[0])` else calls `OUTBUF_APPEND_IP4(buf, val.I Pv4.s_addr)`

⁷¹The correct number of values **MUST** then be appended.

A.10 Important structures

A predominant structure in the main program is the flow table `flow` where the six tuple for the flow lookup timing information is stored as well as a pointer to a possible opposite flow. A plugin can access this structure by including the `packetCapture.h` header. For more information please refer to the header file.

Another important structure is the main output buffer `mainOutputBuffer`. This structure holds all standard output of activated plugins whenever a flow is terminated. The main output buffer is accessible if the plugin includes the header file `main.h`.

A.11 Generating output (advanced)

As mentioned in Section A.8 there are two ways to generate output. The first is the case where a plugin just writes its arbitrary output into its own file, the second is writing flow-based information to a standard output file. We are now discussing the later case.

The standard output file generated by the `txtSink` plugin consists of a header, a delimiter and values. The header is generated using header information provided by each plugin, that writes output into the standard output file. During the initialization phase of the sniffing process, the core calls the `t2PrintHeader()` functions of these plugins. These functions return a single structure or a list of structures of type `binary_value_t`. Each structure represents a statistic. To provide a mechanism for hierarchical ordering, the statistic itself may contain one ore more values and one or more substructures.

The structure contains the following fields:

Name	Type	Description
<code>num_values</code>	<code>uint32_t</code>	Amount of values in the statistic
<code>subval</code>	<code>binary_subvalue_t*</code>	Type definition of the values
<code>name</code>	<code>char[128]</code>	Name of the statistic
<code>desc</code>	<code>char[1024]</code>	Description of the statistic
<code>is_repeating</code>	<code>uint32_t</code>	One, if the statistic is repeating, zero otherwise
<code>next</code>	<code>binary_value_t*</code>	Used if the plugin provides more than one statistics

The substructure `binary_subvalue_t` is used to describe the values of the statistic. For each value, one substructure is required. For example, if `num_values` is two, two substructures have to be allocated. The substructures must be implemented as a continuous array consisting of the following fields:

Name	Type	Description
<code>value_type</code>	<code>uint32_t</code>	Type of the value
<code>num_values</code>	<code>uint32_t</code>	Amount of values in the statistic
<code>subval</code>	<code>binary_subvalue_t*</code>	Definition of the values
<code>is_repeating</code>	<code>uint32_t</code>	one, statistic is repeating, zero otherwise

Compared to the `binary_value_t` representation two strings are omitted in the statistic's short and long description and the `*next` pointer but it contains a new field, the value type. Possible values for this new field are described in the enumeration `binary_types` defined in the header file `binaryValue.h`. If the field contains a value greater than zero the fields `num_values` and `subval` are ignored. They are needed if a `subval` contains itself subvalues. To indicate additional

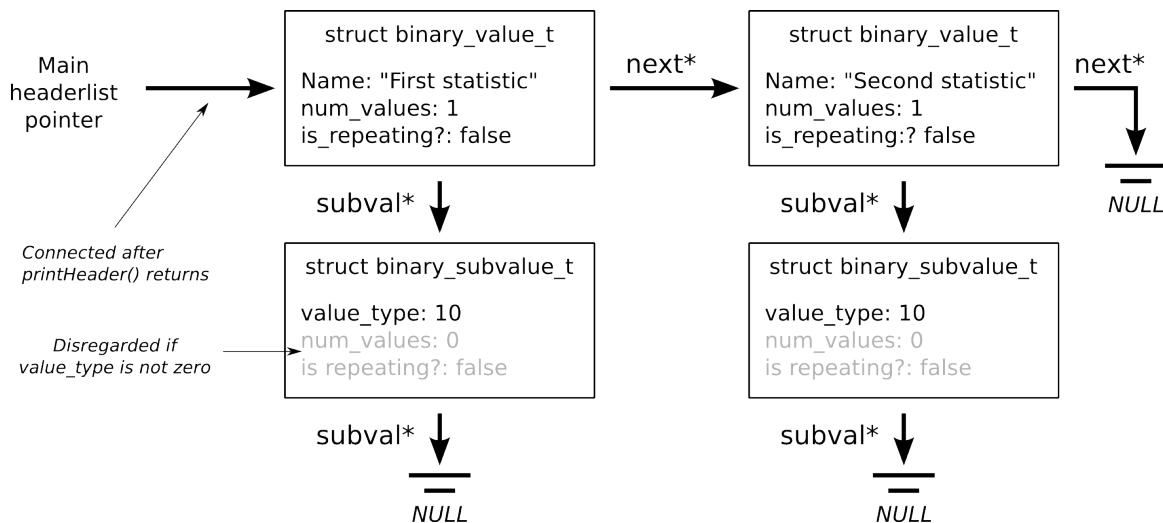
subvalues, the field `value_type` need to be set to zero. The mechanism is the same as for the `binary_value_t`.

The field `is_repeating` should be used if the number of values inside a statistic is variable; e.g. a statistic of a vector with variable length.

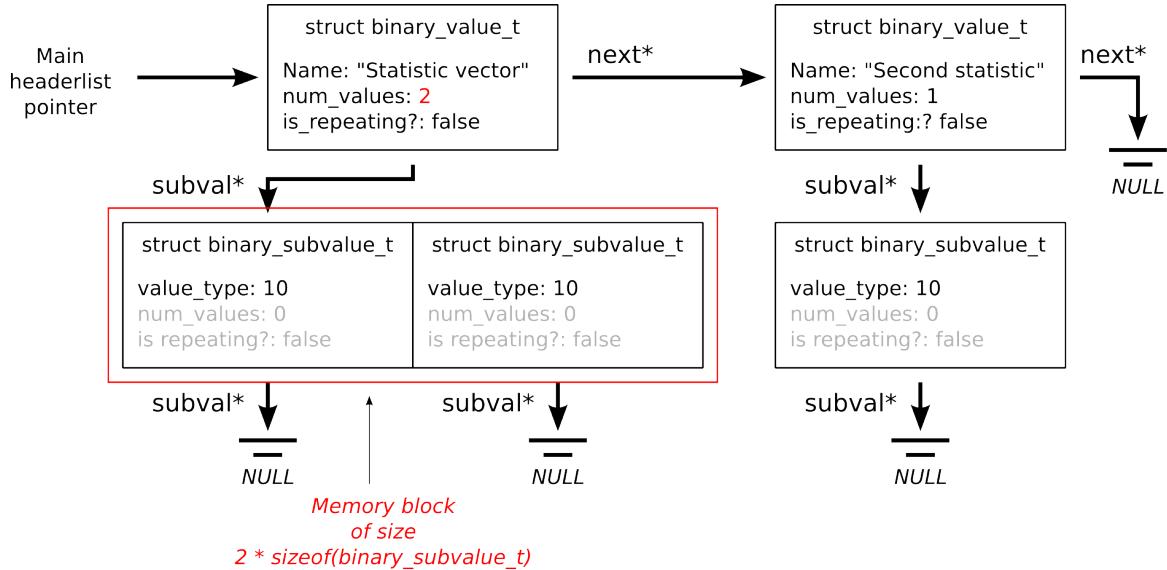
A.11.1 Examples

The following examples illustrate the usage of the said two structures:

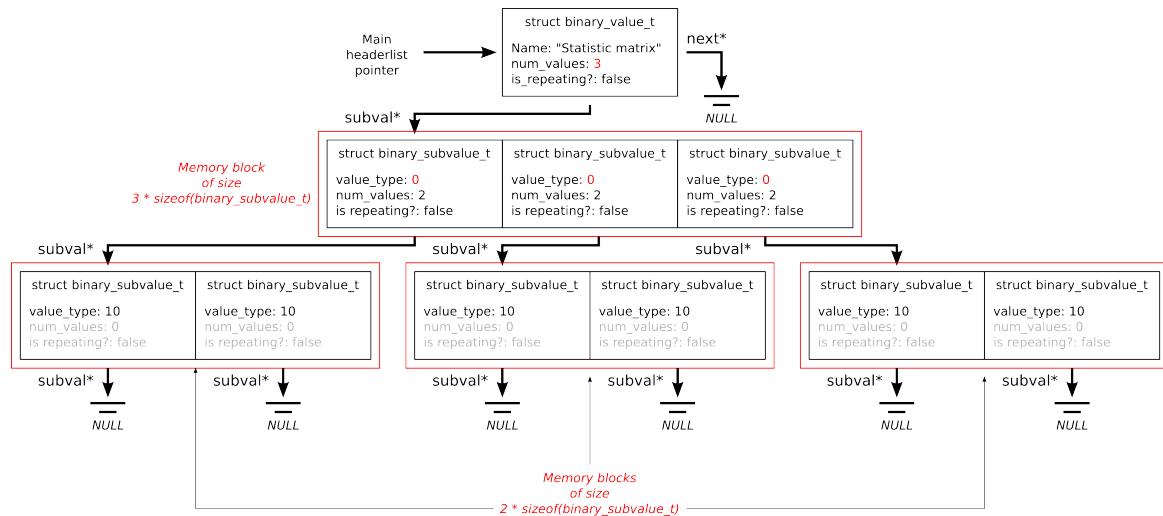
Example 1: Two Statistics each containing a single value If a plugin's output is consisting of two statistics each having a single value it needs to pass a list containing two structures of type `binary_value_t`. Both structures contain a substructure with the type of the single values. The following diagram shows the relationships between all four structures:



Example 2: A statistic composed of two values Now the output of the plugin is again two statistics, but the first statistic consists of two values; e.g. to describe a position on a grid. Therefore `num_values` is two and `subval*` points to a memory field of size two-times `struct binary_subvalue_t`. The subvalues themselves contain again the type of the statistic's values. Note: These values do not need to be identical.



Example 3: A statistic containing a complete matrix With the ability to define subvalues in subvalues it is possible to store multidimensional structures such as matrices. The following example illustrates the definition of a matrix of size three times two:



A.11.2 Helper functions

In order to avoid filling the structures by hand a small API is located in the header file `binaryValue.h` doing all the nitty-gritty work for the programmer. The most important functions are described below.

`binary_value_t* bv_append_bv(binary_value_t* dest, binary_value_t* new)`

Append a `binary_value_t` struct at the end of a list of `binary_value_t` structures.

Return a pointer to the start of the list.

Arguments:

Type	Name	Description
binary_value_t*	dest	pointer to the start of the list
binary_value_t*	new	pointer to the new binary_value_t structure

```
binary_value_t* bv_new_bv (const char *name, const char *desc, uint32_t is_repeating,
                           uint32_t num_values...)
```

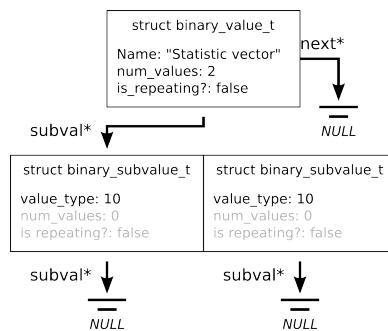
Generate a new structure of type `binary_value_t` and returns a pointer to it.

Arguments:

Type	Name	Description
char*	name	name for the statistic
char*	desc	description for the statistic
uint32_t	is_repeating	one, if the statistic is repeating, zero otherwise
uint32_t	num_values	number of values for the statistic
int	...	types of the statistical values, repeated num_values-times

The function creates a `binary_value_t` structure and sets the values. In addition, it creates an array field with `num_values` `binary_subvalue_t` structures and fills the value types provided in the variable argument list.

Example: The call `bv_new_bv("stat_vec", "Statistic vector", 2, 0, bt_uint_64, bt_uint_64)` creates the following structures:



```
binary_value_t* bv_add_sv_to_bv (binary_value_t* dest, uint32_t pos,
                                 uint32_t is_repeating, uint32_t num_values, ...)
```

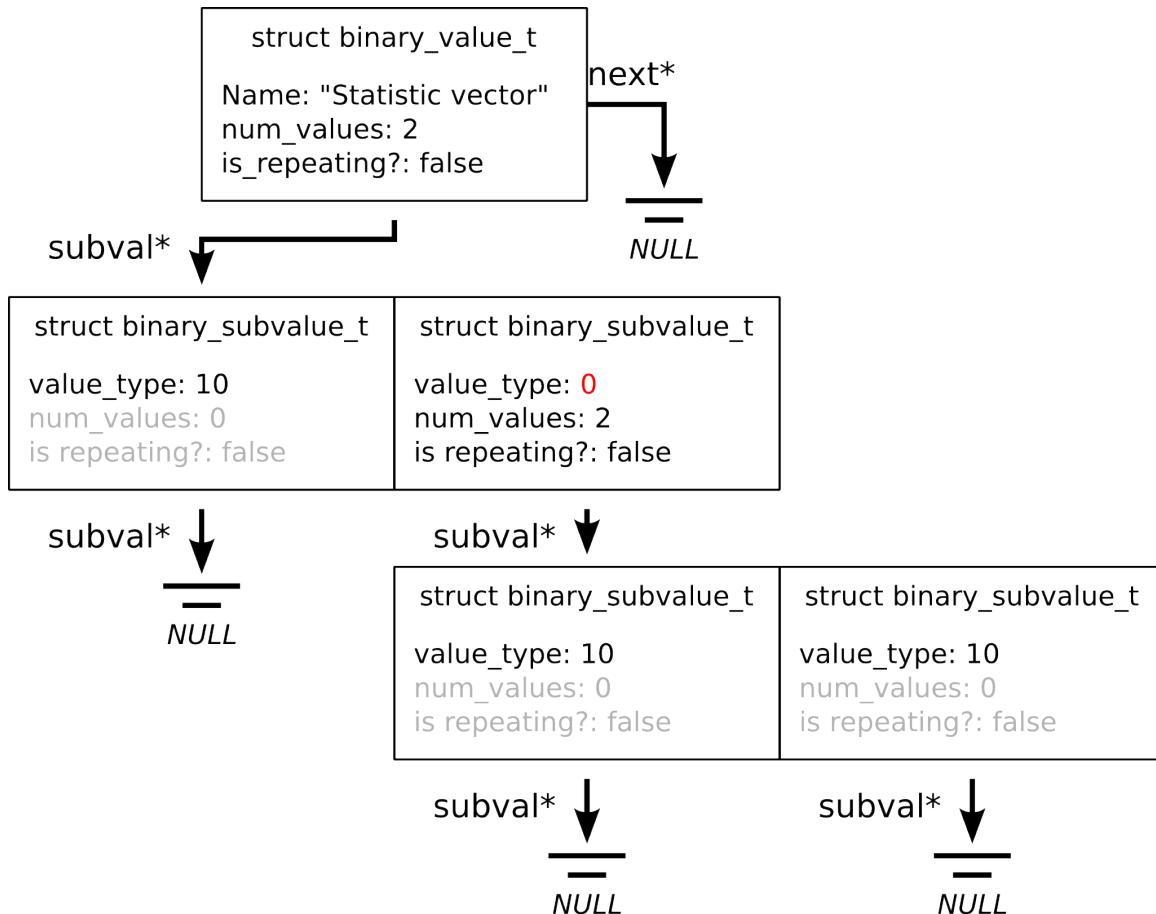
Replace a subvalue in a `binary_value_t` structure with a new substructure that contains additional substructures and returns a pointer to the parent `binary` value.

Arguments:

Type	Name	Description
<code>binary_value_t*</code>	<code>dest</code>	pointer to the parent <code>binary</code> value
<code>uint32_t</code>	<code>pos</code>	position of the substructure to be replaced, starting at 0
<code>uint32_t</code>	<code>is_repeating</code>	one, if the subvalue is repeating, zero otherwise
<code>uint32_t</code>	<code>num_values</code>	number of values in the subvalue
<code>int</code>	<code>...</code>	types of the statistical values, repeated <code>num_values</code> -times

This function is only valid if `dest` is already a complete statistic containing all necessary structures.

Example: Let `dest` be a pointer to the `binary_value_t` structure from the example above. A call to the function `bv_add_sv_to_bv(dest, 1, 0, 2, bt_uint_64, bt_uint_64)` replaces the second substructure with a new substructure containing two more substructures:



```
binary_value_t* bv_add_sv_to_sv (binary_subvalue_t* dest, uint32_t pos,
                                uint32_t is_repeating, uint32_t num_values, ...)
```

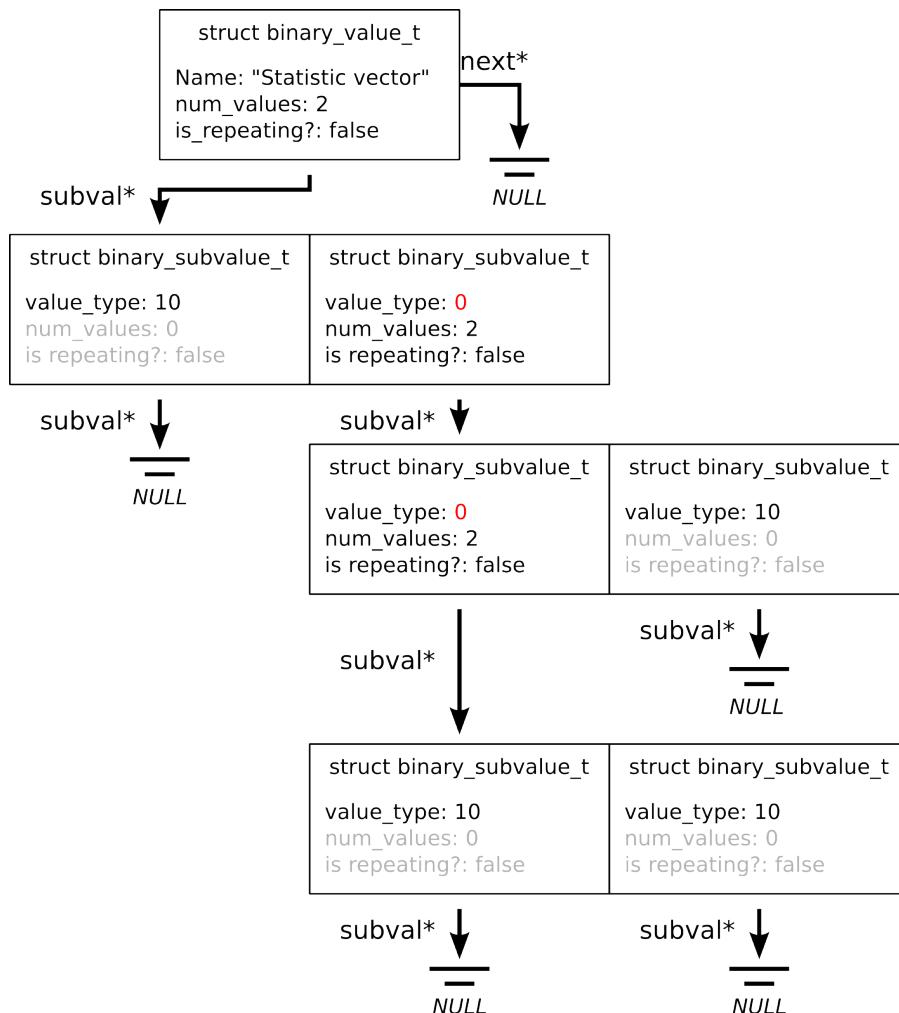
Replace a subvalue in a `binary_subvalue_t` structure with a new substructure that contains additional substructures and returns a pointer to the parent binary subvalue.

Arguments:

Type	Name	Description
<code>binary_subvalue_t*</code>	<code>dest</code>	Pointer to the parent binary subvalue
<code>uint32_t</code>	<code>pos</code>	Position of the substructure to be replaced, starting at 0
<code>uint32_t</code>	<code>is_repeating</code>	one, if the subvalue is repeating, zero otherwise
<code>uint32_t</code>	<code>num_values</code>	Number of values in the subvalue
<code>int</code>	<code>...</code>	Types of the statistical values, repeated <code>num_values</code> -times

For all hierarchical deeper located structures than above the function described above is required.

Example: Let `dest` be a pointer to the subvalue structure being replaced in the example above. A call to the function `bv_add_sv_to_sv(dest, 0, 0, 2, bt_uint_64, bt_uint_64)` replaces `dest`'s first the substructure with a new substructure containing two more substructures:



A.11.3 Writing into the standard output

Standard output is generated using a buffer structure. Upon the event `t2OnFlowTerminate` (see [A.15.6](#)) Plugins write all output into this buffer. It is strongly recommended using the function `outputBuffer_append(outputBuffer_t* buffer, char* output, size_t size_of_output)`.

Arguments:

Type	Name	Description
outputBuffer_t*	buffer	pointer to the standard output buffer structure, for standard output, this is <code>main_output_buffer</code>
char*	output	pointer to the output, currently of type char
size_t	size_of_output	the length of field <code>output</code> in single bytes

The output buffer is send to the *output sinks* after all plugins have stored their information.

Example: If a plugin wants to write two statistics each with a single value of type `uint64_t` it first has to commit its `binary_value_t` structure(s) (see section above). During the call of its `t2OnFlowTerminate()` function the plugin writes both statistical values using the append function:

```
outputBuffer_append(main_output_buffer, (char*) &value1, 4);
outputBuffer_append(main_output_buffer, (char*) &value2, 4);
```

Where `value1` and `value2` are two pointers to the statistical values.

A.12 Writing repeated output

If a statistic could be repeated (field `is_repeating` is one) the plugin has first to store the number of values as `uint32_t` value into the buffer. Afterwards, it appends the values.

Example: A plugin's output is a vector of variable length, the values are of type `uint16_t`. For the current flow, that is terminated in the function `t2OnFlowTerminate()`, there are three values to write. The plugin first writes a field of type `uint32_t` with value three into the buffer, using the append function:

```
outputbuffer_append(main_output_buffer, (char*) &numOfValues, sizeof(uint32_t));
```

Afterwards, it writes the tree values.

A.13 Important notes

- IP addresses (`bt_ip4_addr`, `bt_ip6_addr` and `bt_ipx_addr`) or MAC addresses (`bt_mac_addr`) are stored in network order.
- Strings are of variable length and need to be stored with a trailing zero byte ('\0').

A.14 Administrative functions

Every plugin has to provide five administrative functions. The first four are mandatory while the last one is optional. For convenience, the following two macros can be used instead:

- `T2_PLUGIN_INIT(name, version, t2_v_major, t2_v_minor)`
- `T2_PLUGIN_INIT_WITH_DEPS(name, version, t2_v_major, t2_v_minor, deps)`

For example, to initialize `myPlugin`:

```
T2_PLUGIN_INIT_WITH_DEPS("myPlugin", "0.9.3", 0, 9, "tcpFlags,basicStats")
```

Function	Description
<code>const char *t2PluginName()</code>	Name of the plugin, e.g., "myPlugin".
<code>const char *t2PluginVersion()</code>	Version of the plugin, e.g., "0.9.0"
<code>unsigned int t2SupportedT2Major()</code>	Minimum major version of the core supported by the plugin, e.g., 0
<code>unsigned int t2SupportedT2Minor()</code>	Minimum minor version of the core supported by the plugin, e.g., 9
<code>const char *t2Dependencies()</code>	Comma separated list of plugin names required by the plugin, e.g.,

Function	Description
	"tcpFlags, tcpStates"

The existence of these functions is checked during the plugin initialization phase one and two, as highlighted in Figure 12.

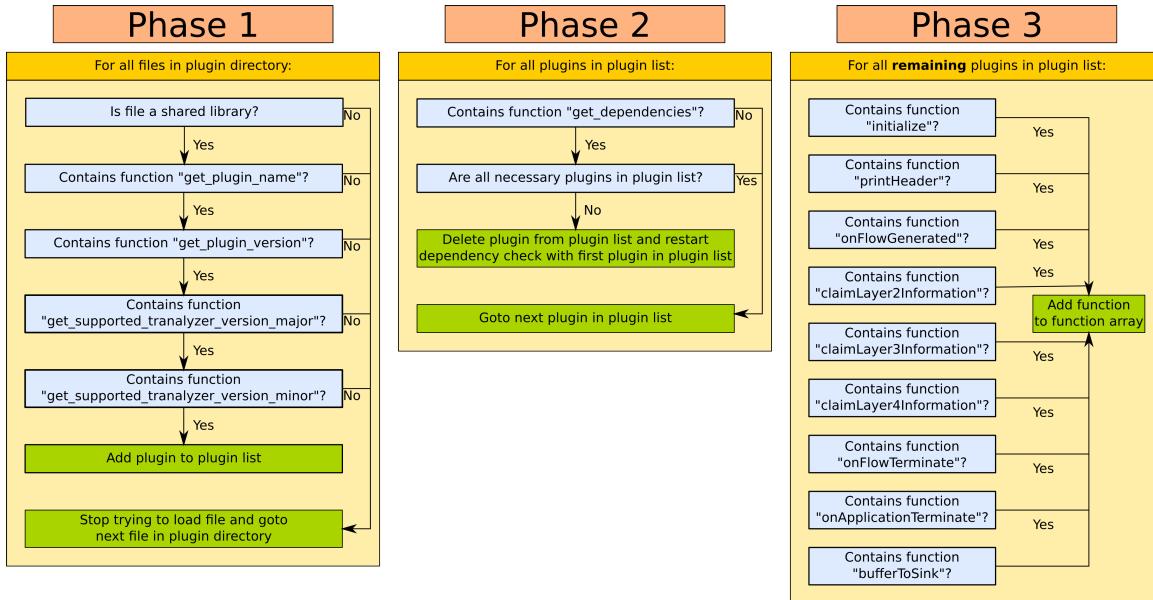


Figure 12: Processing of the plugin loading mechanism

A.15 Processing functions

During flow analysis Tralyzer2 generates several *events* based on the status of the program, the inspected OSI layer of the current packet or the status of the current flow. These events consist of specific function calls provided by the plugins. The implementation of the event functions is dependent on the required action of a plugin to be carried out upon a certain event.

A.15.1 void t2Init()

The `t2Init` event is generated before the program activates the packet capturing phase. After Tralyzer2 has initialized its internal structures it grants the same phase to the plugins. Therefore temporary values should be allocated during that event by using a C `malloc`.

A.15.2 binary_value_t *t2PrintHeader()

This event is also generated during the initialization phase. With this event the plugin providing data to the standard output file signals the core what type of output they want to write (see A.8). The function returns a pointer to the generated `binary_value_t` structure or to the start pointer of a list of generated `binary_value_t` structures.

A.15.3 void t2OnNewFlow(packet_t *packet, unsigned long flowIndex)

This event is generated every time Tranalyzer2 recognizes a new flow not present in the flow table. The first parameter is the currently processed packet, the second denotes the new generated flow index. As long as the flow is not terminated the flow index is valid. After flow termination the flow number is reintegrated into a list for later reuse.

A.15.4 void t2OnLayer2(packet_t *packet, unsigned long flowIndex)

This event is generated for every new packet comprising of a valid and supported layer two header, e.g. Ethernet as default. This is the first event generated after libpcap dispatches a packet and before a lookup in the flow table happened. At this very point in time no tests are conducted for higher layer headers. If a plugin tries to access higher layer structures it has to test itself if they are present or not. Otherwise, at non-presence of higher layers an unchecked access can result in a NULL pointer access and therefore in a possible segmentation fault! We recommend using the subsequent two events to access higher layers.

A.15.5 void t2OnLayer4(packet_t *packet, unsigned long flowIndex)

This event is generated for every new packet containing a valid and supported layer four header. The current supported layer four headers are TCP, UDP and ICMP. This event is called after Tranalyzer2 performs a lookup in its flow table and eventually generates an t2OnNewFlow event. Implementation of other protocols such as IPsec or OSPF are planned.

A.15.6 void t2OnFlowTerminate(unsigned long flowIndex, outputBuffer_t *buf)

This event is generated every time Tranalyzer2 removes a flow from its active status either due to timeout or protocol normal or abnormal termination. Only during this event, the plugins write output to the standard output.

A.15.7 void t2Finalize()

This event is generated shortly before the program is terminated. At this time no more packets or flows are processed. This event enables the plugins to do memory housekeeping, stream buffer flushing or printing of final statistics.

A.15.8 void t2BufferToSink(outputBuffer_t *buffer, binary_value_t *bv)

The Tranalyzer core generates this event immediately after the t2OnFlowTerminate event with the main output buffer as parameter. A plugin listening to this event is able to write this buffer to a data sink. For example the [binSink](#) plugin pushes the output into the `PREFIX_flows.bin` file.

A.16 Timeout handlers

A flow is terminated after a certain timeout being defined by so called *timeout handlers*. The default timeout value for a flow is 182 seconds. The plugins are able to access and change this value. For example, the [tcpStates](#) plugin changes the value according to different connection states of a TCP flow.

A.16.1 Registering a new timeout handler

To register a new timeout handler, a plugin has to call the `timeout_handler_add(float timeout_in_sec)` function. The argument is the new timeout value in seconds. Now the plugin is authorized by the core to change the timeout of a flow to the registered timeout value. Without registering a timeout handler the test is unreliable.

A.16.2 Programming convention and hints

- A call to `timeout_handler_add` should only happen during the initialization function of the plugin.
- Registering the same timeout value twice is no factor.
- Registering timeout values in fractions of seconds is possible, see `tcpStates` plugin.

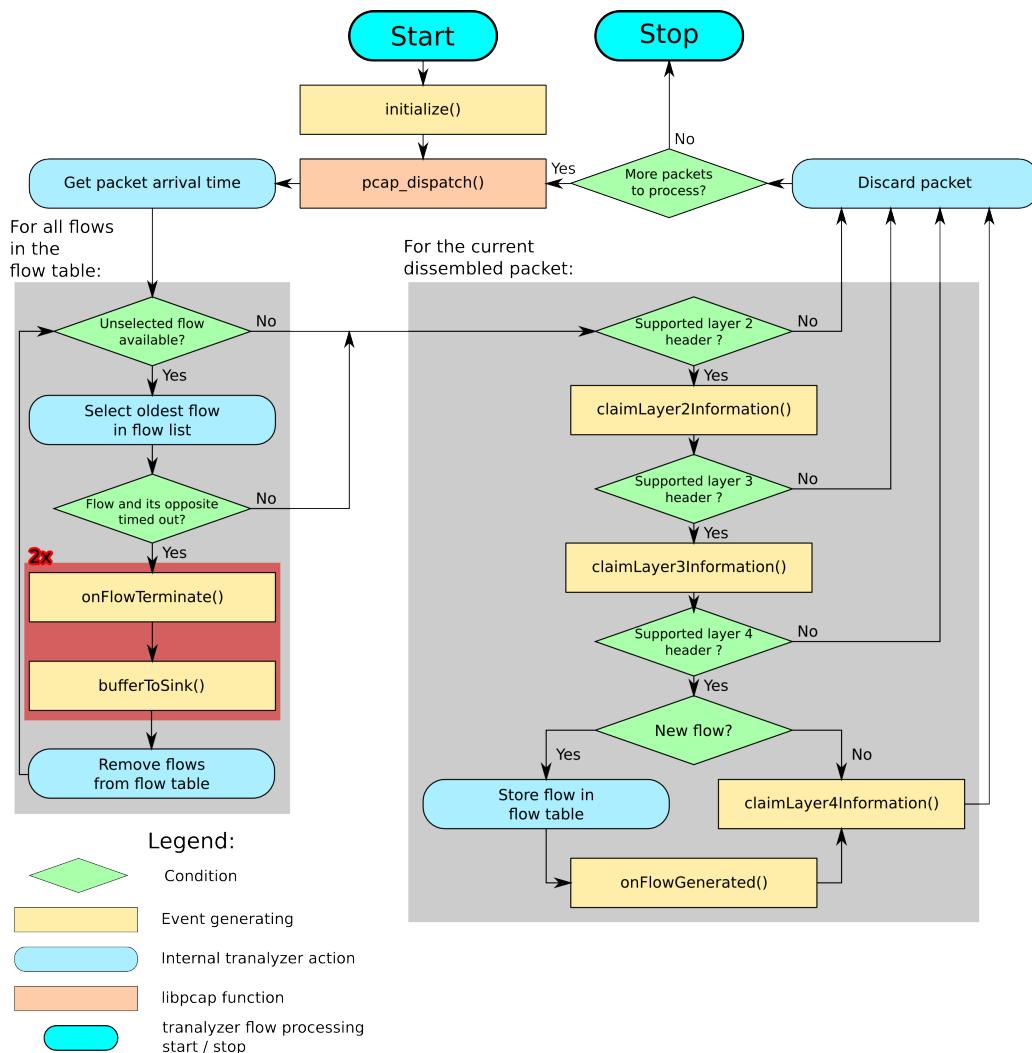


Figure 13: Tranalyzer packet processing and event generation.

B Importing Tranalyzer Flows in Splunk

B.1 Prerequisites

- tranalyzer2-0.9.3 is installed with standard/default plugins,
- Splunk 6.5.x is installed, Splunk account exists,
- At least one network interface (Ethernet or WLAN) has network traffic.

B.2 Select Network Interface

Determine the network interface name by entering the following command:

```
ifconfig
```

at the terminal command line. In the output look for the interface name which has the IP address where the network traffic should be collected from:

```
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST>
mtu 1500 inet 10.20.6.79 netmask 0xfffffc00 broadcast 10.20.7.255
```

B.3 Configure Tranalyzer jsonSink Plugin

Go to *tranalyzer2-0.9.3/plugins/jsonSink/src/jsonSink.h* and set the configuration parameters as needed:

```
#define SOCKET_ON          1 // Whether to output to a socket (1) or file (0)
#define SOCKET_ADDR "127.0.0.1" // address of the socket
#define SOCKET_PORT         5000 // port of the socket
```

Set *SOCKET_ON* to 1 to configure the output to a socket. Set the IP address of the destination server which should receive the data stream. If the localhost will be the destination, leave the default setting "127.0.0.1". Set the socket server port of the destination.

B.4 Recompile the jsonSink Plugin

Enter the following command:

```
tranalyzer2-0.9.3/plugins/jsonSink/autogen.sh
```

Make sure that the plugin is compiled successfully. In this case the following message will be shown at the command line:

```
Plugin jsonSink copied into USER_DIRECTORY/.tranalyzer/plugins
```

B.5 Start Tranalyzer2

Start generating flow records by launching Tranalyzer2 with the interface name determined on the previous step and setting a file name as the command line arguments by entering the command:

```
tranalyzer -i en0 -w test1 &
```

Note that the file name is optional for JSON stream import, if file name is not indicated the records will be shown in the standard output (besides being streamed over the configured TCP socket).

B.5.1 Check File Output

Check that the flow records are written to the file by entering the command:

```
tail -f test1_flows.txt
```

Flow records should be shown in the terminal.

B.5.2 Collect Traffic

Let Tranalyzer2 run and collect network traffic.

B.6 Start Splunk

Start Splunk by entering the following command:

```
splunk start
```

in the directory where Splunk is installed. Wait for the confirmation message that Splunk is up and running:

```
The Splunk web interface is at http://splunk_hostname:8000
```

B.7 Login to Splunk, Import and Search Data

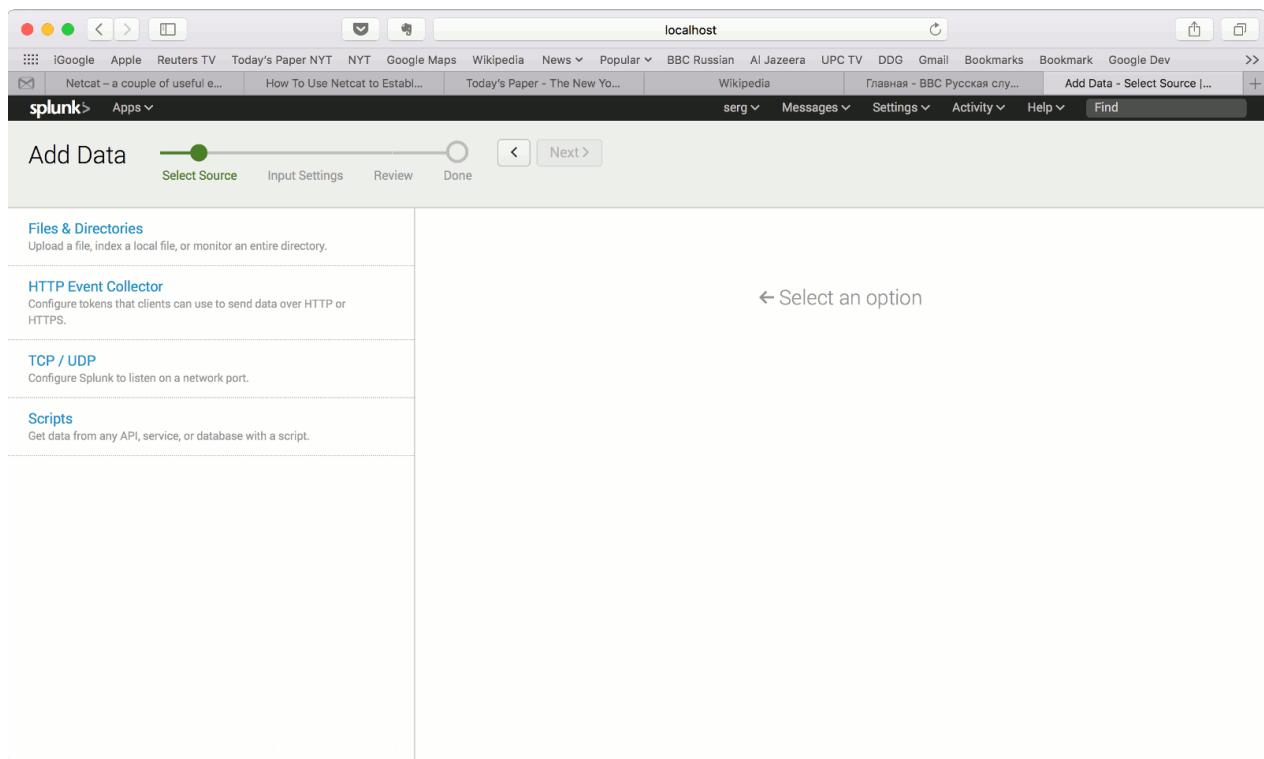


Figure 14: Select “Add Data”.

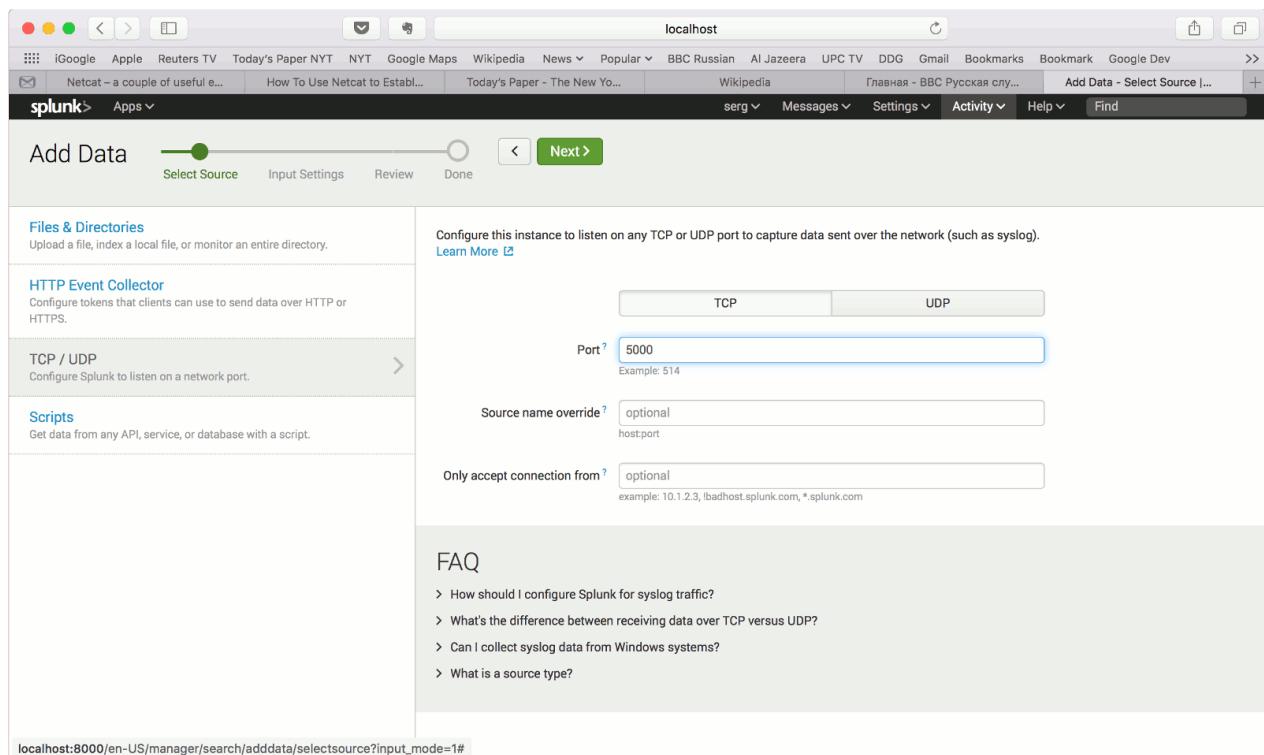


Figure 15: Select “TCP/UDP” and set protocol to “TCP” and set the correct port number (same as in the Tranalyzer2 plugin configuration file, in this example — 5000).

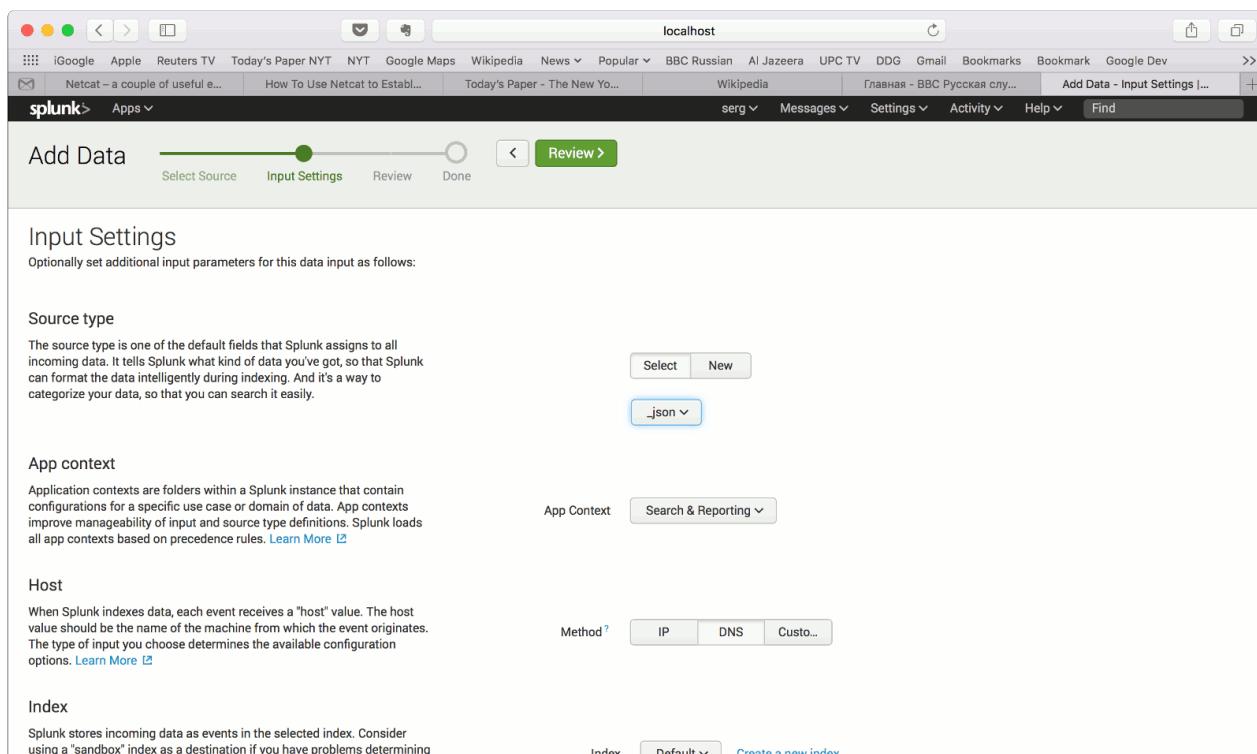


Figure 16: Select “_json” as Source Type and proceed to “Review”.

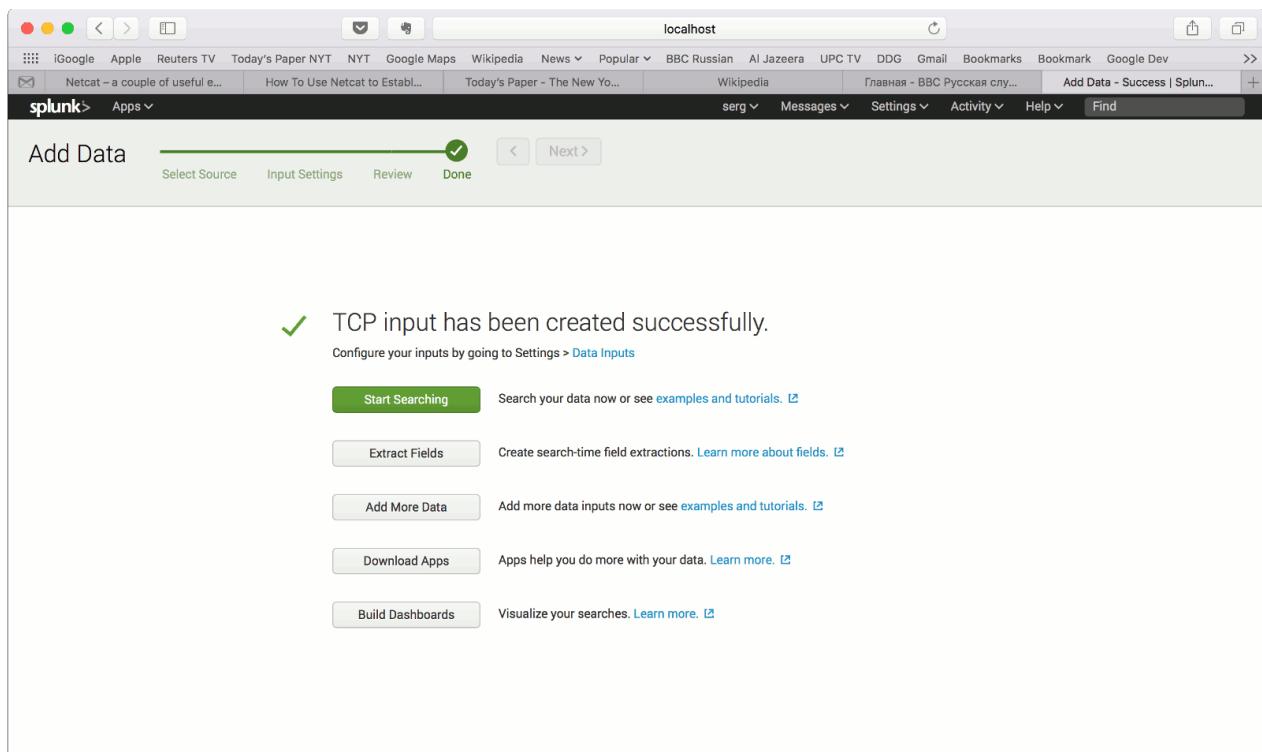


Figure 17: Select “Start Searching” to make sure that the data is being received by Splunk.

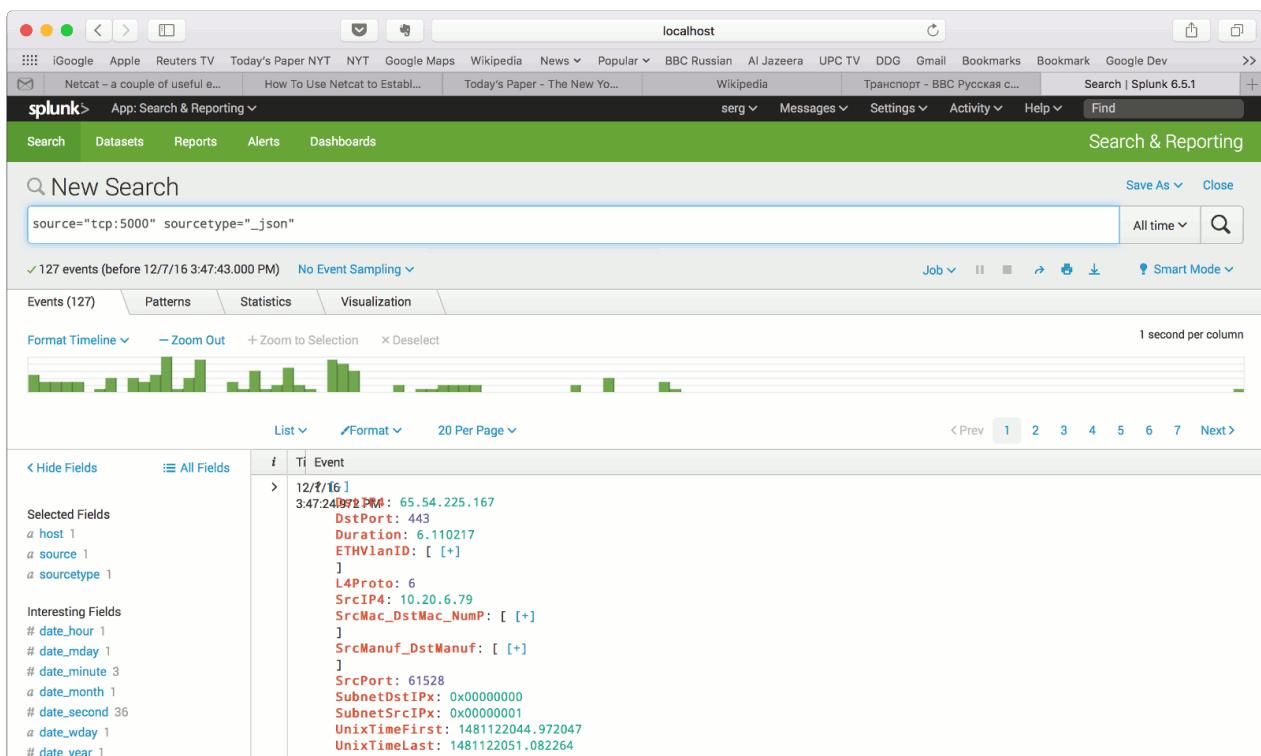


Figure 18: Note that the data is being received, but the Tranalyzer2 specific data record field are not shown yet.

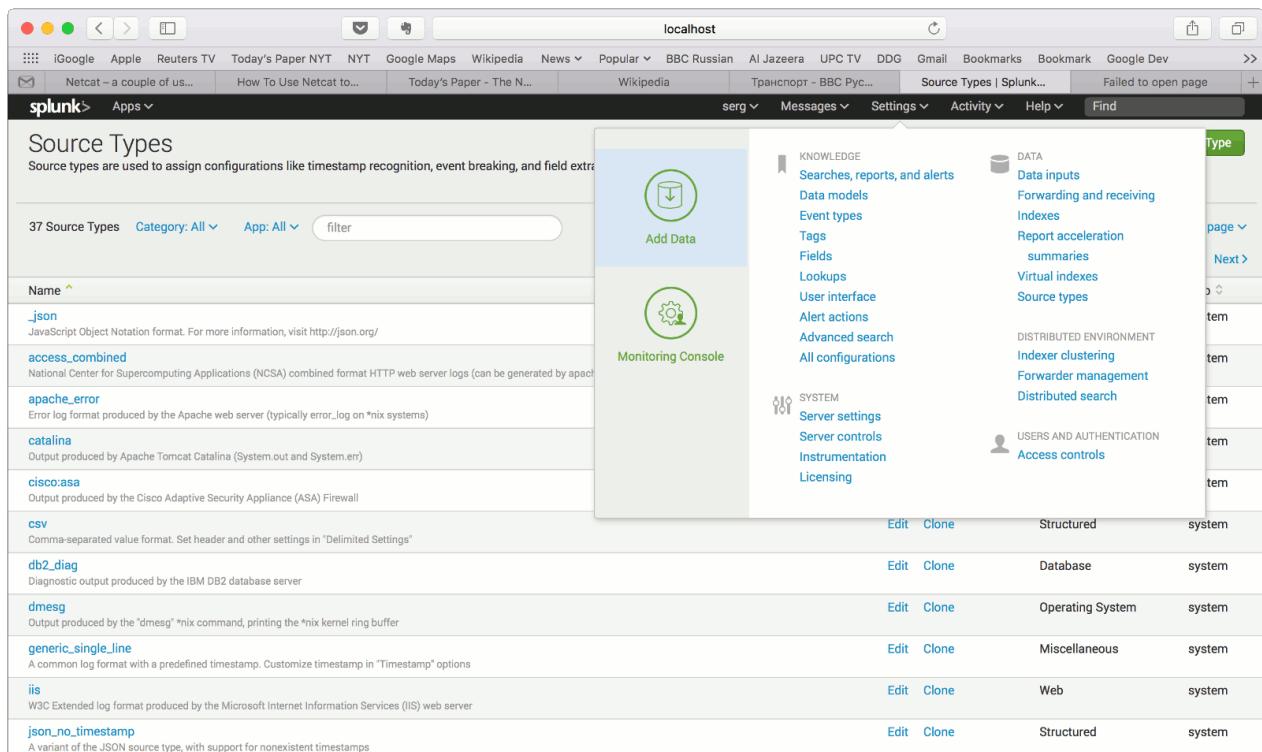


Figure 19: Go to “Settings”->“DATA”->“Source Types” and click on “`_json`” data source type to edit it.

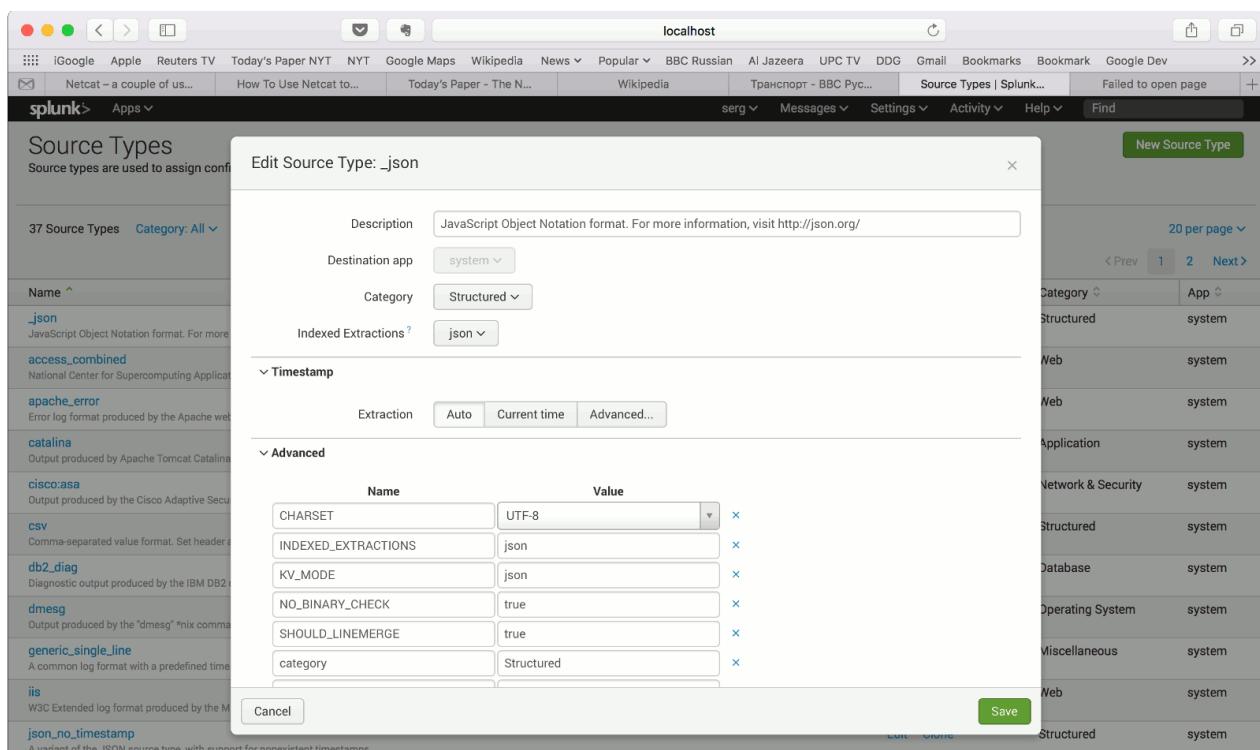


Figure 20: Change option “KV_MODE” from “none” to “json” and save the changes.

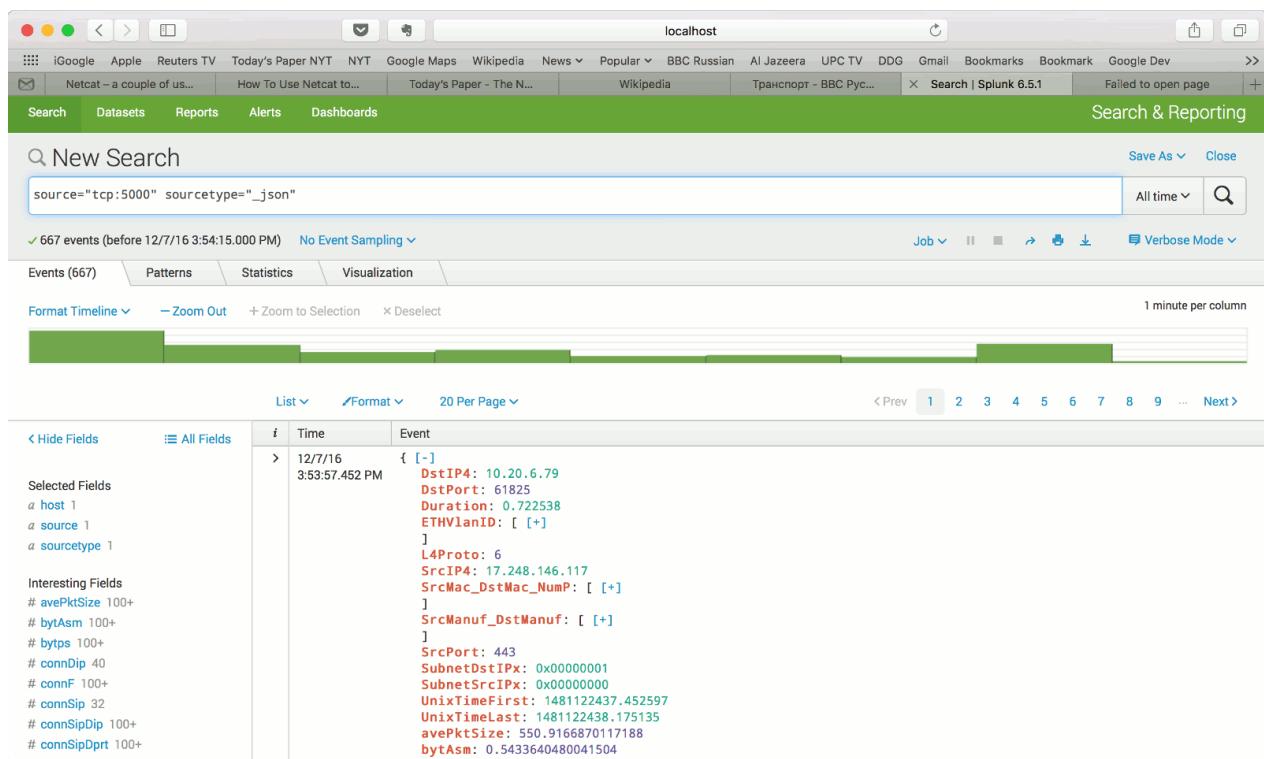


Figure 21: Return to the Search window and make sure that the Tranalyzer2 specific fields are recognized by Splunk.

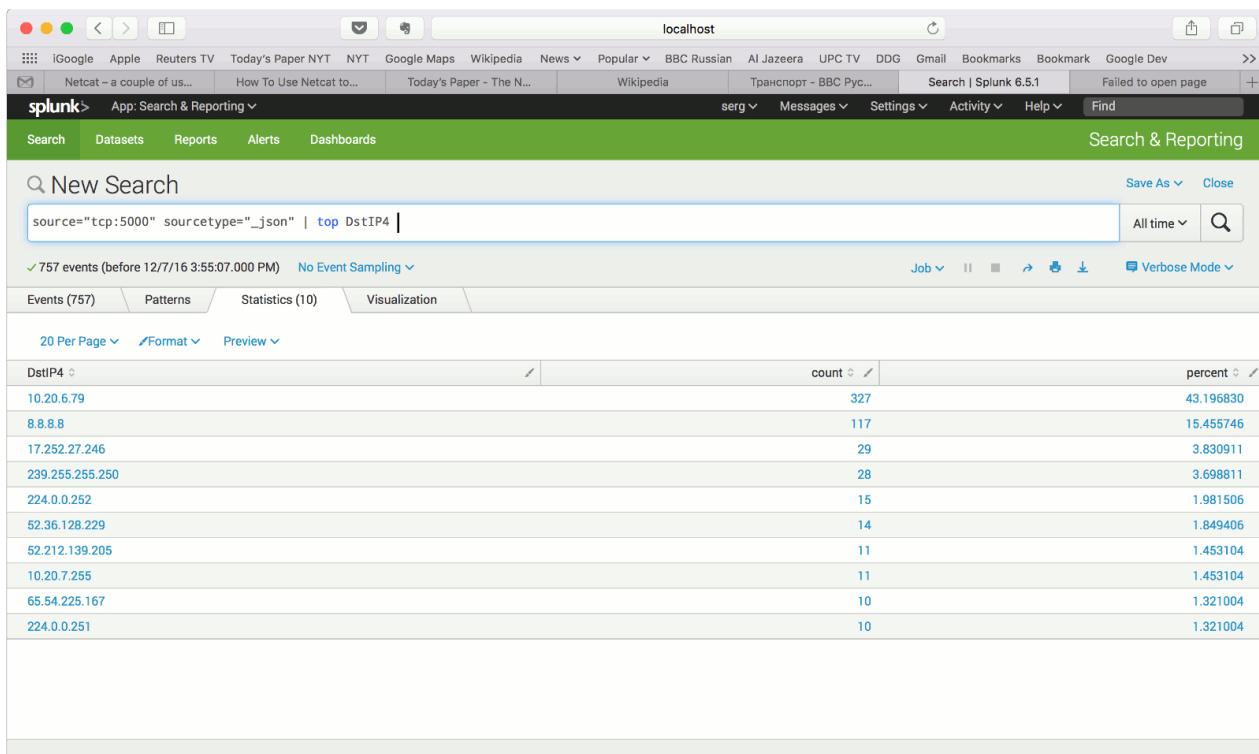


Figure 22: Query data, e.g. show top destination IP addresses by number of the records.

C Advanced Performance Enhancements with PF_RING

Under certain circumstances, e.g., large quantities of small packets, the kernel might drop packets. This happens due to the normal kernel dispatching which is known to be inefficient for packet capture operations. The capturing process can be devised more efficiently by changing the kernel as in `packet_mmap`, but then a patched libpcap is required which is not available yet.⁷² Another option is `pf_ring`. Its kernel module passes the incoming packets in a different way to the user process.⁷³

Requirements

- Kernel version prior to 3.10. ⁷⁴
- All packages needed for building a kernel module, names are distribution-dependent
- A network interface which supports NAPI polling by its driver.
- optional: A network card which supports Direct Network Interface Card (NIC) access (DNA).⁷⁵

Quick setup

Download PF_RING from a stable tar ball or development source at <http://www.ntop.org/get-started/download/>. In order to build the code the following commands have to be executed in a bash window:

```
cd PF_RING/kernel  
make && sudo make install  
modprobe pf_ring
```

Figure 23: building kernel module

Tranalyzer2 requires at least libpfring and libpcap-ring which can be installed the following way:

```
cd PF_RING/userland  
cd lib  
make && sudo make install  
cd ..  
cd libpcap  
make && sudo make install
```

Figure 24: basic userland

You may like to install other tools such as tcpdump. Just install it the same way as described above.

NOTE: The `pf_ring.ko` is loaded having the `transparent_mode=0` by default which enables NAPI polling. If you use a card with special driver support for DNA you may want to compile the driver and load `pf_ring.ko` in a different mode.⁷⁶

⁷²See https://www.kernel.org/doc/Documentation/networking/packet_mmap.txt for more information

⁷³See http://www.ntop.org/products/pf_ring/

⁷⁴Presently when composing this document there is no patch for the depreciation of `create_proc_read_entry()` function. See: <https://lkml.org/lkml/2013/4/11/215>

⁷⁵documentation: http://www.ntop.org/products/pf_ring/DNA/

⁷⁶See: `man modprobe.d`

Load on boot

Since this seems to be difficult for many users the load procedure is described in the following.

Depending on your distribution or to be more specific, the init system your distribution uses at boot time may be somewhere different. In systemd ⁷⁷ create a file with a ‘.conf’ ending at `/etc/modules-load.d/` which contains just the text `pf_ring`, the module name without the ‘.ko’ ending.⁷⁸

Ubuntu uses `/etc/modules` as a single file where you can add a line with the module name.⁷⁹

```
systemd
echo pf_ring > /etc/modules-load.d/pfring.conf
OR
ubuntu
echo pf_ring >> /etc/modules
```

Figure 25: on-boot kernel module load examples

New kernel

Once in a while there is indeed a new kernel available. If you want to use pf_ring afterwards do not forget to recompile the kernel module, or set up dkms.

⁷⁷More info: <http://www.freedesktop.org/wiki/Software/systemd/>

⁷⁸For more info: `man modules-load.d`

⁷⁹See: `man modules`

D Status

This section summarizes the available plugins. For each plugin, a brief description is provided, along with the development status (pre-alpha, alpha, beta, release-candidate, release or deprecated).

D.1 Global Plugins

Plugin Name	Number	Description	Status
protoStats	001	Overall statistics about protocols	release

D.2 Basic Plugins

Plugin Name	Number	Description	Status
basicFlow	100	Overall flow information	release
basicStats	120	Basic statistics	release
connStat	500	Connection statistics	release
macRecorder	110	MAC addresses and manufacturers	release
portClassifier	111	Classification based on port numbers	release

D.3 Layer 2 Protocol Plugins

Plugin Name	Number	Description	Status
arpDecode	179	Address Resolution Protocol (ARP)	beta
cdpDecode	207	Cisco Discovery Protocol (CDP)	beta
lldpDecode	206	Link Layer Discovery Protocol (LLDP)	beta
stpDecode	203	Spanning Tree Protocol (STP)	beta
vtcpDecode	208	VLAN Trunking Protocol (VTP)	beta

D.4 Layer 3/4 Protocol Plugins

Plugin Name	Number	Description	Status
icmpDecode	140	Internet Control Message Protocol (ICMP)	release
igmpDecode	204	Internet Group Management Protocol (IGMP)	alpha
ospfDecode	202	Open Shortest Path First (OSPF)	release
sctpDecode	135	Stream Control Transmission Protocol (SCTP)	beta
tcpFlags	130	IP and TCP flags	release
tcpStates	132	TCP connection tracker	release
vrrpDecode	220	Virtual Router Redundancy Protocol (VRRP)	beta

D.5 Layer 7 Protocol Plugins

Plugin Name	Number	Description	Status
bgpDecode	201	Border Gateway Protocol (BGP)	beta
dhcpDecode	250	Dynamic Host Configuration Protocol (DHCP)	release
dnsDecode	251	Domain Name System (DNS)	beta
ftpDecode	301	File Transfer Protocol (FTP)	release
gquicDecode	312	Google Quick UDP Internet Connections (GQUIC)	beta
gtpDecode	302	GPRS Tunneling Protocol (GTP)	alpha
httpSniffer	310	HyperText Transfer Protocol (HTTP)	release
ircDecode	401	Internet Relay Chat (IRC)	beta
ldapDecode	230	Lightweight Directory Access Protocol (LDAP)	alpha
mndpDecode	252	MikroTik Neighbor Discovery Protocol (MNDP)	beta
modbus	450	Modbus	beta
mqttDecode	452	MQ Telemetry Transport Protocol (MQTT)	alpha
ntlmsspDecode	342	NT LAN Manager (NTLM) Security Support Provider (NTLMSSP)	beta
ntpDecode	205	Network Time Protocol (NTP)	release
popDecode	304	Post Office Protocol (POP)	release
quicDecode	313	Quick UDP Internet Connections (QUIC) (IETF)	beta
radiusDecode	255	Remote Authentication Dial-In User Service (RADIUS)	beta
smbDecode	335	Server Message Block (SMB)	beta
smtpDecode	303	Simple Mail Transfer Protocol (SMTP)	release
snmpDecode	386	Simple Network Management Protocol (SNMP)	beta
sshDecode	309	Secure Shell (SSH)	beta
sslDecode	315	SSL/TLS (Secure Socket Layer/Transport Layer Security), OpenVPN	release
stunDecode	257	STUN, TURN, ICE and NAT-PMP	beta
syslogDecode	260	Syslog	release
telnetDecode	305	Telnet	release
tftpDecode	300	Trivial File Transfer Protocol (TFTP)	release

D.6 Application Plugins

Plugin Name	Number	Description	Status
bitForensic	715	Search packets for specific bits patterns	beta
covertChannels	600	Covert channel detection	release
gsmDecode	555	Global System for Mobile Communications (GSM)	release
pwX	602	Password extractor	release
regexHyperscan	604	Hyperscan regular expressions	beta
regex_pcRE	603	Perl Compatible Regular Expressions (PCRE)	release
regex_re2	606	RE2 regular expressions	release
telegram	620	Telegram	release
torDetector	609	Tor Detector	release
voipDetector	410	Voice over IP (VoIP)	release
wechatDecode	311	WeChat	beta

D.7 Math Plugins

Plugin Name	Number	Description	Status
centrality	800	Centrality	beta
descriptiveStats	702	Descriptive statistics	release
dfft	721	Discrete Fast Fourier Transform	beta
entropy	710	Entropy	beta
nFrstPkts	700	Statistics over the first N packets	release
pktSIATHisto	701	Histograms of packet size and inter-arrival times	release
wavelet	720	Wavelet	beta

D.8 Classifier Plugins

Plugin Name	Number	Description	Status
bayesClassifier	866	Classification using Naive Bayes	beta
fnameLabel	899	Classification based on filename	beta
geoiip	116	Classification based on IP address location	release
nDPI	112	Classification based on content analysis	release
p0f	779	OS classification based on content analysis (SSL)	release
tp0f	117	OS classification based on layer 3/4 (IP/TCP) analysis	release

D.9 Output Plugins

Plugin Name	Number	Description	Status
binSink	900	Binary output into a flow file	release
clickhouseSink	927	Output into a ClickHouse database	release
findexer	961	Produce a binary index mapping flow index and packets	release
jsonSink	903	Produce a JSON file	release
kafkaSink	928	Output into an Apache Kafka event streaming platform	beta
liveXtr	962	Extract flagged flows in pcap files during processing	beta
mongoSink	926	Output into a MongoDB database	beta
mysqlSink	925	Output into a MariaDB/MySQL database	beta
netflowSink	904	NetFlow output format for existing Cisco tools	beta
payloadDumper	862	Dump the payload of TCP/UDP flows to files	beta
pcapd	960	Store packets from specific flows in pcap files	release
pgsqlSink	923	Output into a PostgreSQL database	beta
socketSink	910	Binary output into a TCP/UDP socket	release
sqliteSink	924	Output into a SQLite database	beta
txtSink	901	Text output into a flow file	release

E TODO

This section lists some features, capabilities and plugins which Tranalyzer is currently missing. Feel free to pick a task or two and contribute code, plugins or ideas.

E.1 Features

- Anonymization
- Endianness independence
- Support for NetMon dump files
- Stream reassembly and reordering
- Parallelization

E.2 Plugins

E.2.1 Layer 2 Protocol Plugins

- Extreme Discovery Protocol (EDP)
- Dynamic Trunk Protocol (DTP)
- Port Aggregation Protocol (PAgP)

E.2.2 Layer 3/4 Protocol Plugins

- Datagram Congestion Control Protocol (DCCP)
- Reliable Datagram Sockets (RDS)
- Reliable UDP (RUDP)
- Resource Reservation Protocol (RSVP)
- Signalling Connection Control Part (SCCP)

E.2.3 Layer 7 Protocol Plugins

- Distributed Computing Environment / Remote Procedure Calls (DCE/RPC)
- Dropbox
- Extensible Messaging and Presence Protocol (XMPP)
- Financial Information eXchange Protocol (FIX)
- Internet Message Access Protocol (IMAP)
- Internet Small Computer Systems Interface (iSCSI)
- Microsoft Tabular Data Stream (TDS)

- Network File System (NFS)
- OCSP, PKIX-CRL (include in `sslDecode`?)
- Routing Protocols:
 - Enhanced Interior Gateway Routing Protocol (EIGRP)
 - Routing Information Protocol (RIP)
 - Cisco Hot Standby Router Protocol (HSRP)
 - ...
- Signaling System No. 7 (SS7) protocols
- Torrent, Gnutella
- X11

F FAQ

This section answers some frequently asked questions.

F.1 After ./setup.sh, I receive the following error: -bash: t2: command not found

After running the setup.sh script, you need to open a new bash window or source your .bashrc (or .zshrc, ...) file in order for the new aliases and functions to be available:

```
$ t2  
-bash: t2: command not found  
$ source ~/.bashrc  
$ t2  
Tranalyzer 0.9.0 - High performance flow based network traffic analyzer
```

Usage

```
tranalyzer [OPTION...] <INPUT>
```

...

F.2 If the hashtable is full, how much memory do I need to add?

When T2 warns you that the hashtable is full, it also tells you how to correct the problem:

```
[WRN] Hash Autopilot: main HashMap full: flushing 1 oldest flow(s)  
[INF] Hash Autopilot: Fix: Invoke T2 with '-f 5'
```

T2 calculates an estimate of the multiplication factor HASHFACTOR which you can set with the -f command-line option. By default the main hash autopilot is enabled which maintains the sanity of T2 even if it runs out of flow memory. Nevertheless, T2 will be faster if you feed him the recommended -f factor.

F.3 Can I change the timeout of a specific flow in my plugin?

That is possible because each flow owns a timeout value which can be altered even on packet basis. It enables the user to program stateful protocol plugins. Check out the [tcpStates](#) plugin as an inspiration.

F.4 Can I reduce the maximal flow length?

In `tranalyzer2/src/tranalyzer.h` you will find a constant called FDURLIMIT. Set it to the amount of seconds you like and T2 will terminate every flow with max FDURLIMIT+1 seconds. And create a new flow for the next packet to come.

F.5 How can I change the separation character in the flow file?

The separation character is defined as SEP_CHAR in `utils/bin2txt.h`. It can be set to any character(s), e.g., "," or "||". In addition, the character(s) used for comments, e.g., column names, is controlled by HDR_CHR in the same file. Note that Tranalyzer default values are "\t" and "%", respectively. Be advised that if you changed either of those values, some scripts may not work as expected.

F.6 How can I build all the plugins?

If you invoked the script setup.sh then you may use

t2build -a

otherwise, old school:

```
cd /tralyzer2-0.9.3
./autogen.sh -a
```

F.7 T2 failed to compile: What can I do?

If a dependency is missing, you should see an appropriate message, e.g., *Missing dependency libname*. If no such message is displayed, it could be that the Makefiles are outdated. Then use autogen.sh -r to force the rebuild of the Makefiles. A typical error requiring the use of autogen.sh -r is:

```
...
/bin/bash: line 10: automake-: command not found
Makefile:333: recipe for target 'Makefile.in' failed
make[1]: *** [Makefile.in] Error 127
...
...
```

If you see the following message, then the autotools are not installed.

```
make: Entering directory '/home/user/tralyzer2-0.9.3/tralyzer2/doc'
make: Nothing to be done for 'clean'.
make: Leaving directory '/home/user/tralyzer2-0.9.3/tralyzer2/doc'
../autogen.sh: line 116: autoreconf: command not found
../autogen.sh: line 118: ./configure: No such file or directory
```

Failed to configure tralyzer2

In this case, please refer to the *doc/tutorials/install.pdf*.

F.8 T2 segfaults: What can I do?

T2 never segfaults! Unless he deviates from his cosmic plan and indeed segfaults. The prominent reason are memory inconsistencies with old plugins being resident under `~/.tralyzer/plugins/`.

1. Remove all the plugins: `rm ~/.tralyzer/plugins/*.so`
2. Recompile the plugins, e.g., `cd ~/tralyzer2-0.9.3/ && ./autogen.sh`
3. T2 should behave again.

For the developer:

If that does not fix the problem, recompile T2 in debug mode with `./autogen.sh -d` and try to run tralyzer in `gdb: gdb -args ./tralyzer -r file.pcap -w outpref`. If the error happens while writing flows, try to remove plugins until the error disappears. Finally, run the `segvtrack` script as follows: `segvtrack yourpcap`. This will automatically reduce the PCAP to the smallest set of packets which causes a segfault. If this does not help, send us a bug report at andy@tralyzer.com with this pcap, T2 configuration (the values that differ from the default) and the plugins you are using. Then we will get a fix for you in no time.

F.9 socketSink plugin aborts with “could not connect to socket: Connection refused”

The `socketSink` plugin acts as a client in a socket communication. Therefore, a server listening to SERVADD, DPORT and SOCKTYPE is required. As described in the **Example** Section of the `socketSink` plugin documentation, a simple server can be set up with netcat as follows: `nc -l 127.0.0.1 6666`. Make sure the address and port match the values listed in `socketSink.h`.

F.10 T2 stalls after USR1 interrupt: What can I do?

It is a bug in the libpcap, which somehow is not thread-safe under certain conditions. Check whether T2 is set to default signal threading mode in (`main.h`):

- Set `MONINTTHRD` to 1
- Set `MONINTPSYNC` to 1

Do not forget to recompile T2 with `./autogen.sh` if you had to change the configuration.

Now the process of printing is detached from the packet capture and the output is synchronized to the packet processing main loop. Thus, pcap is never interrupted.

F.11 Can I reuse my configuration between different machines or Tranalyzer versions?

You can write a patch for `t2conf` and use it as follows: `t2conf --patch file.patch`. Revert the patch with the `--rpatch` option. The patch is a simple text file listing the defines to change, e.g., `IPV6_ACTIVATE <tab> 1 <tab> 0 <tab> tranalyzer2/src/networkHeaders.h`. For more details, refer to the documentation of `t2conf`.

F.12 How to contribute code, submit a bug or request a feature?

Contact the Anteater via email at andy@tranalyzer.com, and he will answer you.