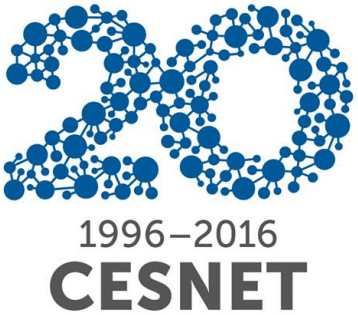


www.liberouter.org

Tools for Security Analysis of Traffic on L7

Practical course

50th TF-CSIRT meeting and FIRST Regional Symposium for Europe

Part I

Introduction

Section 1

Security Tools as a Service

Security Tools as a Service (STaaS)

Increase network security without deep expertise

STaaS provides:

- Network monitoring
- Flow data storage
- Traffic analysis
- Various detections
- Reporting

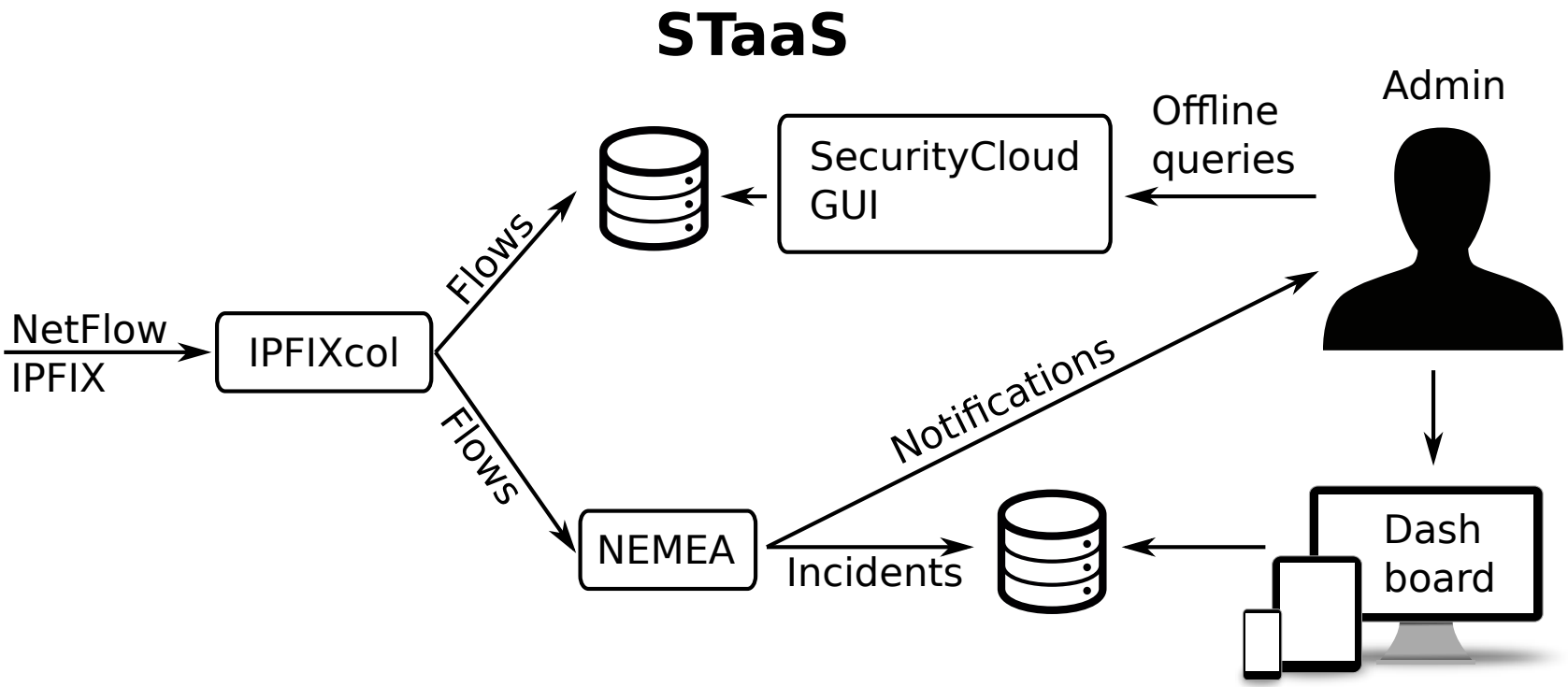
All components are prepared in a virtual machine ready to receive NetFlow/IPFIX

STaaS Components

All components developed by CESNET:

- Exporter (Flow Meter)
- Collector (IPFIXcol)
- Detection framework (NEMEA)
- Report analysis GUI (NEMEA Dashboard)
- Flow data querying tools (fbitdump, fdistdump)
- Data query GUI (SecurityCloud GUI)

STaaS Components



STaaS VM

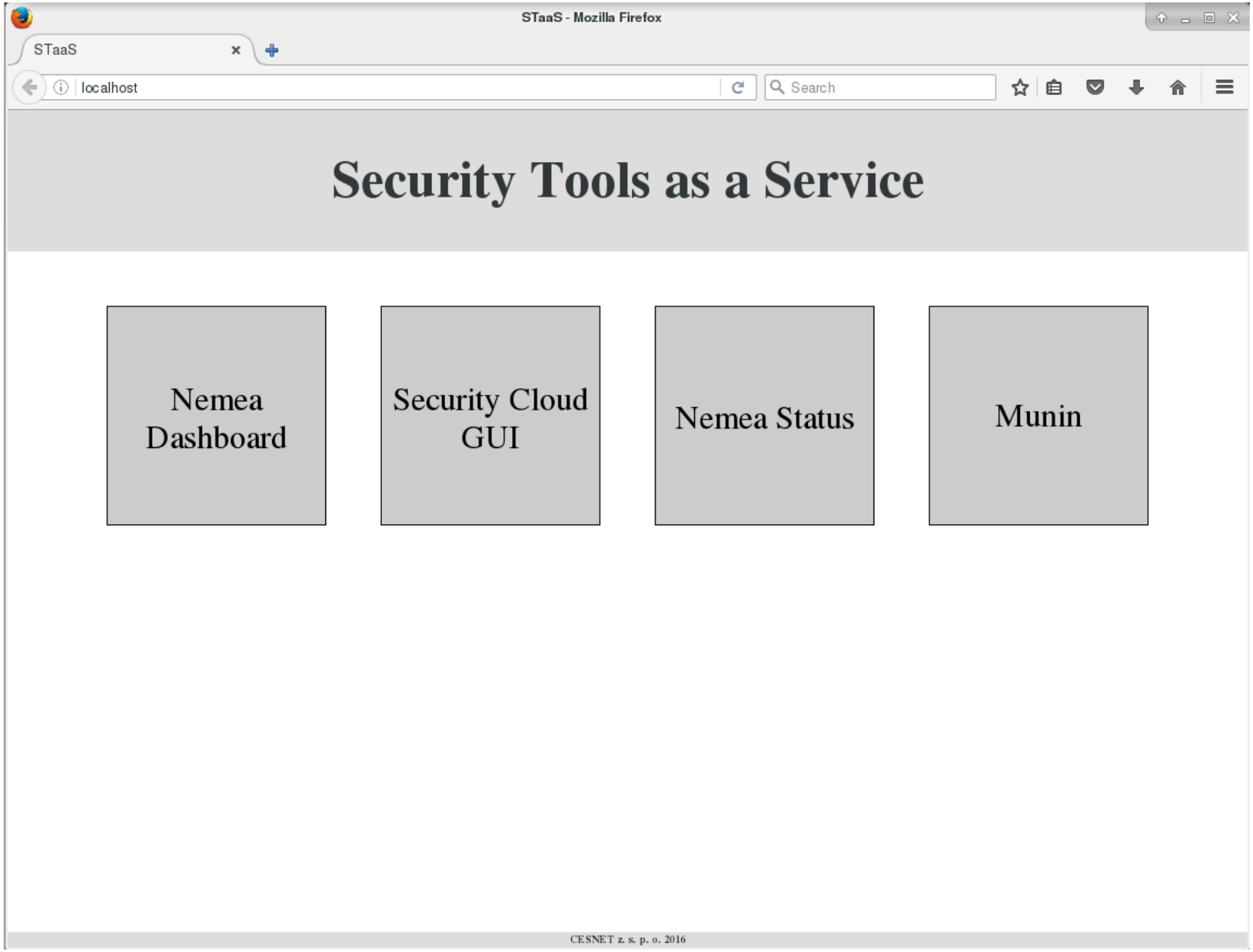
Your virtual machine is an instance of STaaS VM with extra:

- User account
- X Server
- Offline demo data
- Specialized configuration

STaaS VM is built using Ansible orchestration, based on CentOS 7

Several GUIs accessible from a guidepost at `http://localhost`

STaaS Homepage



Section 2

Flow monitoring overview

Flow monitoring

Monitoring of network traffic in terms of metadata about individual L4 connections.

IP flow = set of packets with the same:

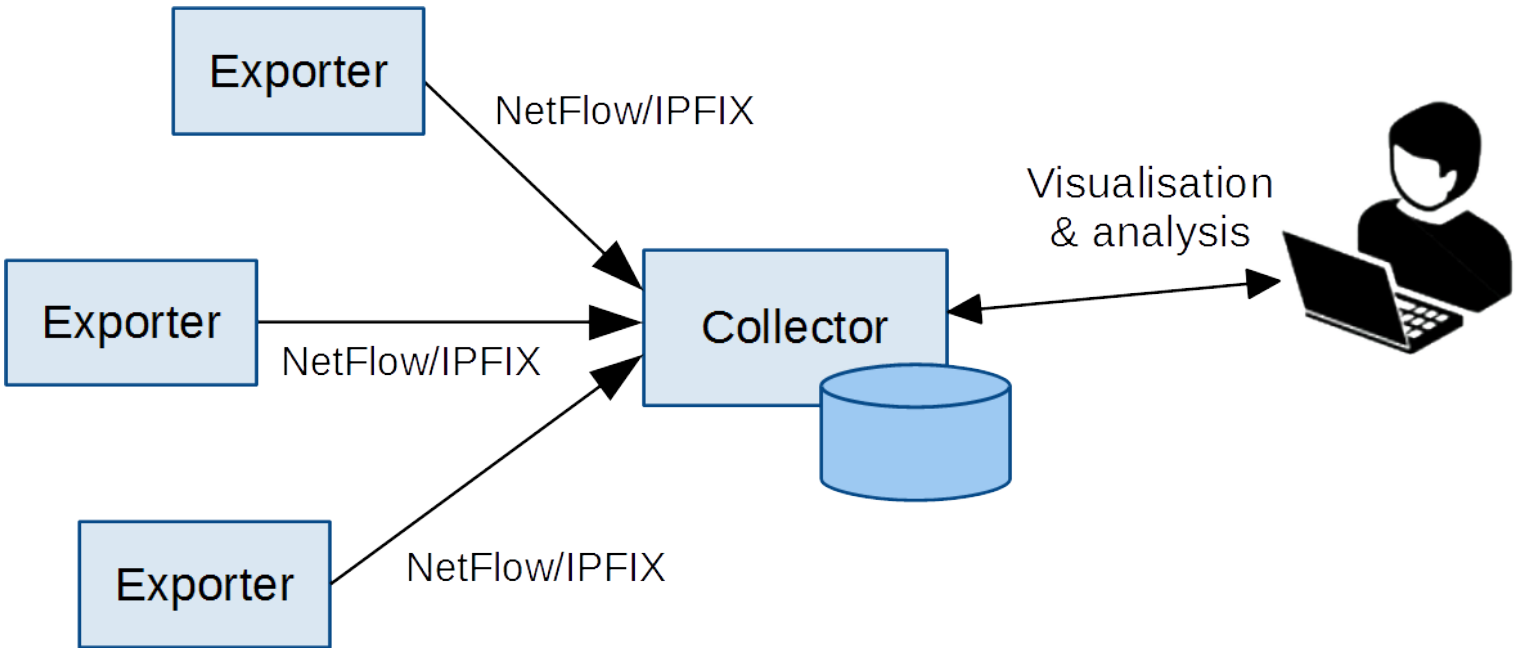
- Source and destination IP address
- L4 protocol (TCP/UDP/ICMP/...)
- Source and destination port
- IP type of service
- Input interface

One TCP/UDP connection consists of two flows – one in each direction.

Flow monitoring architecture

General architecture

- Exporters (sensors, probes) – observe traffic, measure flows
- Collector – stores flow records, allows to query them



Exporter

Exporter

- Router
- Dedicated probe (HW or SW)

Flow exporter aggregates packets into ***flow records***

- IP flow key (addresses, ports, protocol)
- Time of first and last packet of the flow
- Number of packets and bytes
- TCP flags (logical OR of flags field of all packets)
- ToS, input ifc, output ifc, ...

Examples:

- Routers, FlowMon, nProbe, YAF, softflowd, ...

Exporter

Flow record is exported when:

- No packet of the flow arrives for duration of **inactive timeout** (30 s)
- Flow duration is longer than **active timeout** (300 s = 5 min)
- Not enough space in flow cache of the exporter (oldest flows are exported)
- FIN or RST flag is observed in TCP flow (in some implementations)

Collector

Collector

- Storage of flow records
- Manual queries
- Automatic analysis

- Data traditionally stored into files per 5 minutes
  ($\rightarrow$ 5 min = very often used time unit in network monitoring)

- Examples:
  - Nfdump/nfcapd, IPFIXcol, nTop, SiLK, SecurityCloud collector, ...

Protocol

Protocol – format of flow records & transport

- NetFlow (v5, v9) by Cisco
- Jflow, NetStream – NetFlow equivalents of other vendors
- IPFIX – IETF standard

IPFIX fully extensible, any new fields can be introduced.

(sFlow - sampled packets, not flow monitoring)

Flow monitoring

Flow monitoring can tell us:

- Who communicated with who, when, how much data was transferred, etc.
- We don't see data content

Example:

```
Date flow start        Duration  Proto    Src IP:Port          Dst IP:Port Packets Bytes
2015-06-22 12:34:56.123   0.110  TCP    192.0.2.82:8420 ->  198.51.100.5:80     5   742
2015-06-22 12:34:56.567   1.502  TCP  198.51.100.5:80   ->    192.0.2.82:8420  10  2685
2015-06-22 12:34:57.222   0.241  TCP    192.0.2.45:4571 -> 203.0.113.100:5060   3   540
```

Flow data can tell us ...

Statistics

- Top 5 TCP/UDP ports by number of bytes transferred

Port	Flows(%)	Packets(%)	Bytes(%)	pps	bps	bpp
80	6.8 M(24.5)	371.4 M(41.9)	341.2 G(45.5)	168148	1.2 G	918
443	6.4 M(23.0)	255.1 M(28.7)	217.5 G(29.0)	115461	787.5 M	852
1935	46829(0.2)	9.5 M(1.1)	9.8 G(1.3)	4321	35.4 M	1023
22	298078(1.1)	12.9 M(1.5)	9.2 G(1.2)	5840	33.4 M	714
8000	17951(0.1)	8.7 M(1.0)	7.1 G(1.0)	3929	25.9 M	823

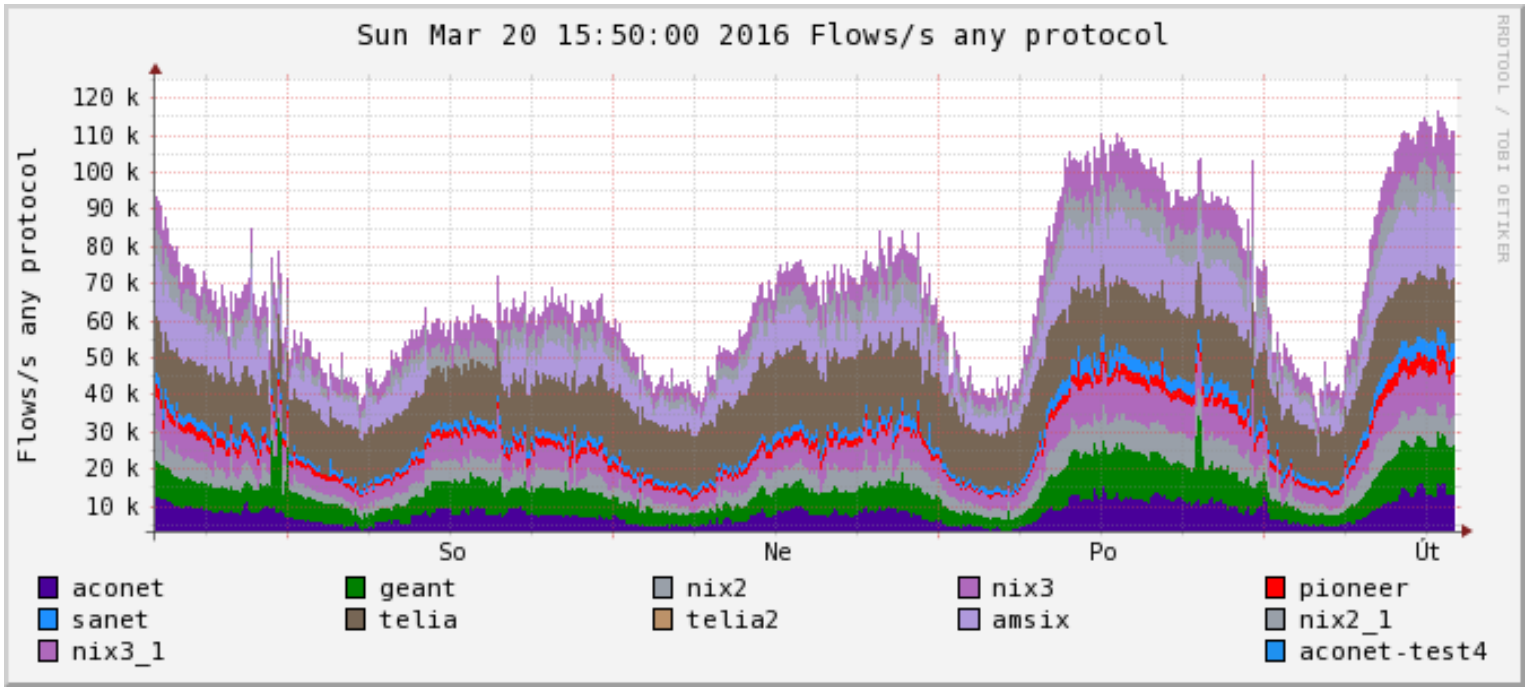
(tcp/1935 = RTMP, Flash video streaming)

- Top 5 TCP/UDP ports by number of flows

Port	Flows(%)	Packets(%)	Bytes(%)	pps	bps	bpp
53	7.2 M(26.1)	7.5 M(0.8)	1.2 G(0.2)	3405	4.4 M	162
80	6.8 M(24.5)	371.4 M(41.9)	341.2 G(45.5)	168148	1.2 G	918
443	6.4 M(23.0)	255.1 M(28.7)	217.5 G(29.0)	115461	787.5 M	852
123	663728(2.4)	1.5 M(0.2)	94.7 M(0.0)	693	342946	61
23	324216(1.2)	988500(0.1)	188.8 M(0.0)	447	683560	190

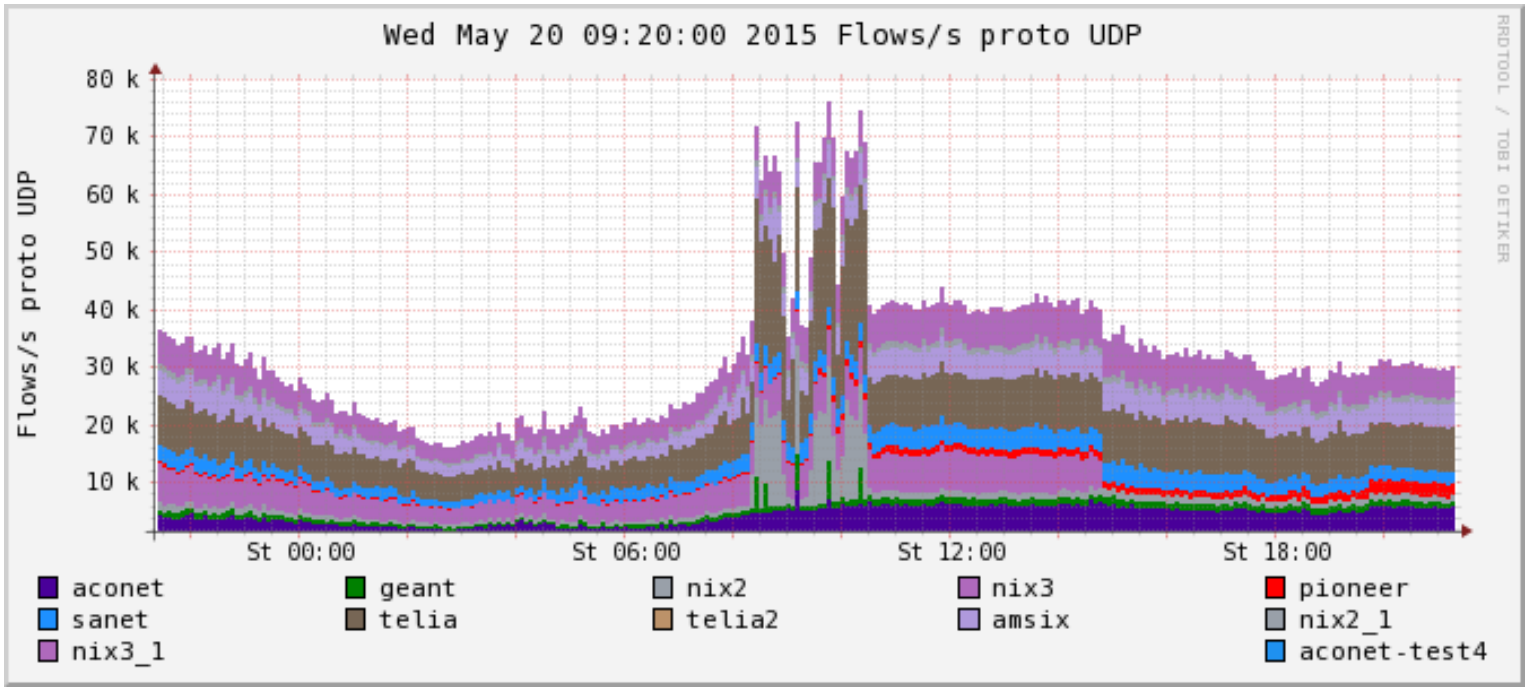
Flow data can tell us ...

Time series of data volume



Flow data can tell us ...

Time series of data volume & anomalies



Flow data can tell us ...

Communication of a particular IP address

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes
2015-05-27 09:16:32.808	290.647	TCP	195.113.219.239:53489	->	173.192.82.196:80	.AP.S.	0	33	2056
2015-05-27 09:17:16.306	290.160	TCP	195.113.219.239:53492	->	46.255.224.22:80	.AP.S.	0	34	3678
2015-05-27 09:17:16.307	290.159	TCP	46.255.224.22:80	->	195.113.219.239:53492	.AP.S.	0	35	3366
2015-05-27 09:17:26.680	290.651	TCP	173.192.82.196:80	->	195.113.219.239:53489	.AP.S.	164	61	2731
2015-05-27 09:19:25.329	0.039	TCP	23.63.28.240:80	->	195.113.219.239:53497	.AP.SF	0	4	483
2015-05-27 09:19:53.588	0.035	TCP	23.63.28.240:80	->	195.113.219.239:53500	.AP.SF	0	4	484
2015-05-27 09:19:55.795	1.565	TCP	23.63.28.240:80	->	195.113.219.239:53502	.AP.SF	0	6	874
2015-05-27 09:20:12.921	0.313	TCP	23.63.28.240:80	->	195.113.219.239:53495	.AP.SF	0	5	834
2015-05-27 09:20:13.072	0.202	TCP	23.63.28.240:80	->	195.113.219.239:53499	.AP.S.	0	10	11800
2015-05-27 09:20:52.082	0.548	TCP	204.154.94.81:443	->	195.113.219.239:53506	.AP.S.	0	9	1903
2015-05-27 09:20:52.479	0.725	TCP	195.113.219.239:53506	->	204.154.94.81:443	.AP.S.	0	9	1616
2015-05-27 09:21:18.325	6.227	TCP	195.113.219.239:53508	->	216.34.181.96:80	.AP.SF	0	14	1859
2015-05-27 09:21:18.339	5.665	TCP	216.34.181.96:80	->	195.113.219.239:53507	.AP.SF	0	27	30879
2015-05-27 09:21:18.398	5.324	TCP	195.113.219.239:53510	->	216.34.181.96:80	.AP.SF	0	10	1070
2015-05-27 09:21:18.444	5.665	TCP	195.113.219.239:53507	->	216.34.181.96:80	.AP.SF	0	14	1865
2015-05-27 09:21:18.470	0.720	TCP	195.113.219.239:53511	->	216.34.181.60:80	.AP.SF	0	6	1948
2015-05-27 09:21:18.715	5.551	TCP	216.34.181.96:80	->	195.113.219.239:53513	.A..SF	0	3	128
2015-05-27 09:21:18.726	5.324	TCP	216.34.181.96:80	->	195.113.219.239:53510	.AP.SF	0	14	15014
2015-05-27 09:21:18.844	5.338	TCP	195.113.219.239:53509	->	216.34.181.96:80	.AP.SF	0	8	988
2015-05-27 09:21:18.862	5.338	TCP	216.34.181.96:80	->	195.113.219.239:53509	.AP.SF	0	11	10134
2015-05-27 09:21:18.865	6.227	TCP	216.34.181.96:80	->	195.113.219.239:53508	.AP.SF	0	22	23168
2015-05-27 09:21:18.888	5.604	TCP	195.113.219.239:53514	->	216.34.181.60:80	.A..SF	0	4	172
2015-05-27 09:21:18.915	0.721	TCP	216.34.181.60:80	->	195.113.219.239:53511	.AP.SF	0	5	1661
2015-05-27 09:21:19.036	5.472	TCP	216.34.181.60:80	->	195.113.219.239:53514	.A..SF	0	3	132
2015-05-27 09:21:19.319	5.686	TCP	195.113.219.239:53513	->	216.34.181.96:80	.A..SF	0	4	172
2015-05-27 09:21:19.381	5.552	TCP	216.34.181.96:80	->	195.113.219.239:53512	.A..SF	0	3	128
2015-05-27 09:21:19.406	5.687	TCP	195.113.219.239:53512	->	216.34.181.96:80	.A..SF	0	4	172
2015-05-27 09:21:54.954	0.000	TCP	204.154.94.81:443	->	195.113.219.239:53506	.A.R..	0	1	40
2015-05-27 09:21:55.995	0.000	TCP	221.9.207.70:63383	->	195.113.219.239:8080	....S.	0	1	40
2015-05-27 09:23:22.334	0.000	UDP	71.6.216.48:17185	->	195.113.219.239:17185	......	0	1	92
2015-05-27 09:23:24.359	0.000	UDP	80.82.78.186:47652	->	195.113.219.239:53	......	0	1	68

Flow data can tell us ...

Communication of a particular IP address

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes
2015-05-27 09:16:32.808	290.647	TCP	195.113.219.239:53489	->	173.192.82.196:80	.AP.S.	0	33	2056
2015-05-27 09:17:16.306	290.160	TCP	195.113.219.239:53492	->	46.255.224.22:80	.AP.S.	0	34	3678
2015-05-27 09:17:16.307	290.159	TCP	46.255.224.22:80	->	195.113.219.239	.AP.S.	0	35	3366
2015-05-27 09:17:26.680	290.651	TCP	173.192.82.196:80	->	195.113.219.			61	2731
2015-05-27 09:19:25.329	0.039	TCP	23.63.28.240:80	->	195.113.219.			4	483
2015-05-27 09:19:53.588	0.035	TCP	23.63.28.240:80	->	195.113.219.239:53500	.AP.SF	0	4	484
2015-05-27 09:19:55.795	1.565	TCP	23.63.28.240:80	->	195.113.219.239:53502	.AP.SF	0	6	874
2015-05-27 09:20:12.921	0.313	TCP	23.63.28.240:80	->	195.113.219.239:53495	.AP.SF	0	5	834
2015-05-27 09:20:13.072	0.202	TCP	23.63.28.240:		195.113.219.239:53499	.AP.S.	0	10	11800
2015-05-27 09:20:52.082	0.548	TCP	204.15					9	1903
2015-05-27 09:20:52.479	0.725	TCP	195.113.					9	1616
2015-05-27 09:21:18.325	6.227	TCP	195.113.219.239:53508	->	216.34.181.96:80	.AP.SF	0	14	1859
2015-05-27 09:21:18.339	5.665	TCP	216.34.181.96:80	->	195.113.219.239:53507	.AP.SF	0	27	30879
2015-05-27 09:21:18.398	5.324	TCP	195.113.219.239:53510	->	216.34.181.96:80	.AP.SF	0	10	1070
2015-05-27 09:21:18.444	5.665	TCP	195.113.219.239:53507	->	216.34.181.96:80	.AP.SF	0	14	1865
2015-05-27 09:21:18.470	0.720	TCP	195.113.219.239:53511	->	216.34.181.60:80	.AP.SF	0	6	1948
2015-05-27 09:21:18.715	5.551	TCP	216.34.181.96:80	->	195.113.219.239:53513	.A..SF	0	3	128
2015-05-27 09:21:18.726	5.324	TCP	216.34.181.96:80	->	195.113.219.239:53510	.AP.SF	0	14	15014
2015-05-27 09:21:18.844	5.338	TCP	195.113.219.239:53509	->	216.34.181.96:80	.AP.SF	0	8	988
2015-05-27 09:21:18.862	5.338	TCP	216.34.181.96:80	->	195.113.219.239:	.AP.SF	0	11	10134
2015-05-27 09:21:18.865	6.227	TCP	216.34.181.96:80	->	195.113.219.			22	23168
2015-05-27 09:21:18.888	5.604	TCP	195.113.219.239:53514	->	216.34.181			4	172
2015-05-27 09:21:18.915	0.721	TCP	216.34.181.60:80	->	195.113.219.239:53511	.AP.SF	0	5	1661
2015-05-27 09:21:19.036	5.472	TCP	216.34.181.60:80	->	195.113.219.239:53514	.A..SF	0	3	132
2015-05-27 09:21:19.319				->	216.34.181.96:80	.A..SF	0	4	172
2015-05-27 09:21:19.381				->	195.113.219.239:53512	.A..SF	0	3	128
2015-05-27 09:21:19.406	5.687		195.113.219.239:53512	->	216.34.181.96:80	.A..SF	0	4	172
2015-05-27 09:21:54.954	0.000	TCP	204.154.94.81:443	->	195.113.219.239:53506	.A.R..	0	1	40
2015-05-27 09:21:55.995	0.000	TCP	221.9.207.70:63383	->	195.113.219.239:8080	....S.	0	1	40
2015-05-27 09:23:22.334	0.000	UDP	71.6.216.48:17185	->	195.113.219.239:17185	.....	0	1	92
2015-05-27 09:23:24.359	0.000	UDP	80.82.78.186:47652	->	195.113.219.239:53	.....	0	1	68

gm-pool.centrum.cz

a23-63-28-240.deploy.static.akamaitechnologies.com

projects.sourceforge.net

70.207.9.221.adsl-pool.jlccptt.net.cn

NXDOMAIN

Flow data can tell us ...

Port scanning

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes
2016-03-26 14:09:02.974	0.000	TCP	192.0.2.16:42149	->	198.51.11.13:23	....S.	0	1	60
2016-03-26 14:08:58.290	0.000	TCP	192.0.2.16:33548	->	198.51.10.255:23	....S.	0	1	60
2016-03-26 14:09:03.049	0.000	TCP	192.0.2.16:44087	->	198.51.11.18:23	....S.	0	1	60
2016-03-26 14:09:02.992	0.000	TCP	192.0.2.16:54404	->	198.51.11.21:23	....S.	0	1	60
2016-03-26 14:08:58.414	0.000	TCP	192.0.2.16:40069	->	198.51.11.2:23	....S.	0	1	60
2016-03-26 14:09:07.189	0.000	TCP	192.0.2.16:37117	->	198.51.11.79:23	....S.	0	1	60
2016-03-26 14:09:07.191	0.000	TCP	192.0.2.16:42858	->	198.51.11.83:23	....S.	0	1	60
2016-03-26 14:09:07.240	0.000	TCP	192.0.2.16:40563	->	198.51.11.137:23	....S.	0	1	60
2016-03-26 14:09:07.170	0.000	TCP	192.0.2.16:35695	->	198.51.11.74:23	....S.	0	1	60
2016-03-26 14:09:07.178	0.000	TCP	192.0.2.16:57156	->	198.51.11.91:23	....S.	0	1	60
2016-03-26 14:09:07.171	0.000	TCP	192.0.2.16:39550	->	198.51.11.76:23	....S.	0	1	60
2016-03-26 14:08:57.609	0.000	TCP	192.0.2.16:56841	->	198.51.11.0:23	....S.	0	1	60
2016-03-26 14:09:03.234	0.000	TCP	192.0.2.16:50386	->	198.51.11.72:23	....S.	0	1	60
2016-03-26 14:08:57.604	0.000	TCP	192.0.2.16:44978	->	198.51.10.254:23	....S.	0	1	60
2016-03-26 14:09:03.162	0.000	TCP	192.0.2.16:52435	->	198.51.11.23:23	....S.	0	1	60
2016-03-26 14:09:07.162	0.000	TCP	192.0.2.16:44402	->	198.51.11.92:23	....S.	0	1	60
2016-03-26 14:09:03.142	0.000	TCP	192.0.2.16:43832	->	198.51.11.10:23	....S.	0	1	60
2016-03-26 14:09:07.137	0.000	TCP	192.0.2.16:55152	->	198.51.11.75:23	....S.	0	1	60
2016-03-26 14:09:03.120	0.000	TCP	192.0.2.16:48476	->	198.51.11.25:23	....S.	0	1	60
2016-03-26 14:08:57.503	0.000	TCP	192.0.2.16:59112	->	198.51.10.233:23	....S.	0	1	60
2016-03-26 14:09:07.105	0.000	TCP	192.0.2.16:37002	->	198.51.11.84:23	....S.	0	1	60
2016-03-26 14:08:57.533	0.000	TCP	192.0.2.16:53655	->	198.51.10.252:23	....S.	0	1	60
2016-03-26 14:09:03.098	0.000	TCP	192.0.2.16:36861	->	198.51.11.20:23	....S.	0	1	60
2016-03-26 14:08:57.508	0.000	TCP	192.0.2.16:52513	->	198.51.10.244:23	....S.	0	1	60
2016-03-26 14:09:03.092	0.000	TCP	192.0.2.16:38909	->	198.51.11.9:23	....S.	0	1	60
2016-03-26 14:09:07.221	0.000	TCP	192.0.2.16:45407	->	198.51.11.96:23	....S.	0	1	60
2016-03-26 14:09:07.367	0.000	TCP	192.0.2.16:46191	->	198.51.11.98:23	....S.	0	1	60

Flow data can tell us ...

Part of DNS amplification DDoS attack

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Packets	Bytes
2016-03-16 04:49:34.939	0.000	UDP	114.99.41.106:4444	->	195.113.18.52:53	1	65
2016-03-16 04:49:26.306	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:26.298	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.291	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.252	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.238	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.216	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.202	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.191	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.160	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.156	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.106	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.098	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.076	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.061	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:36.041	0.000	UDP	195.113.18.52:53	->	114.99.41.106:4444	3	3366
2016-03-16 04:49:38.326	0.000	UDP	114.99.41.106:4444	->	195.113.18.52:53	1	65
2016-03-16 04:49:38.326	0.000	UDP	114.99.41.106:4444	->	195.113.18.52:53	1	65
2016-03-16 04:49:38.326	0.000	UDP	114.99.41.106:4444	->	195.113.18.52:53	1	65
2016-03-16 04:49:38.326	0.000	UDP	114.99.41.106:4444	->	195.113.18.52:53	1	65
2016-03-16 04:49:38.326	0.000	UDP	114.99.41.106:4444	->	195.113.18.52:53	1	65
2016-03-16 04:49:38.326	0.000	UDP	114.99.41.106:4444	->	195.113.18.52:53	1	65
2016-03-16 04:49:38.326	0.000	UDP	114.99.41.106:4444	->	195.113.18.52:53	1	65
2016-03-16 04:49:38.326	0.000	UDP	114.99.41.106:4444	->	195.113.18.52:53	1	65
2016-03-16 04:49:38.326	0.000	UDP	114.99.41.106:4444	->	195.113.18.52:53	1	65
2016-03-16 04:49:38.326	0.000	UDP	114.99.41.106:4444	->	195.113.18.52:53	1	65
2016-03-16 04:49:38.326	0.000	UDP	114.99.41.106:4444	->	195.113.18.52:53	1	65

Flow monitoring

- Such attacks are usually easy to recognize when we see their traffic only.
- It is much more complicated to find them in tons of other communication.
- Detection covered by later sessions ...

Section 3

Flow monitoring extended by application layer information

L7 extended flows

Traditional flows

- Only network and transport layer (L3 & L4).

L7 extended flows

- Exporter parses headers of selected L7 protocols
- The most important fields are added to flow records
- Examples:
  - HTTP: Method, URL, Host, UserAgent, Response code, ContentType
  - DNS: queried domain name, returned IP address
  - SMTP: From, To, Cc, Bcc, Subject
  - SIP: message type, From, To, UserAgent
- Allows analysis impossible with traditional flows, with only small impact on data size

L7 extended flows – example

Traditional flows

```
Date flow start        Duration  Proto    Src IP:Port          Dst IP:Port Packets Bytes
2015-06-22 12:34:56.123   0.110  TCP    192.0.2.82:8420 ->  198.51.100.5:80     5   742
2015-06-22 12:34:56.567   1.502  TCP  198.51.100.5:80   ->    192.0.2.82:8420  10  2685
2015-06-22 12:34:57.222   0.241  TCP    192.0.2.45:4571 -> 203.0.113.100:5060   3   540
```

L7 extended flows

```
Date flow start        Duration  Proto    Src IP:Port          Dst IP:Port Packets Bytes
2015-06-22 12:34:56.123   0.110  TCP    192.0.2.82:8420 ->  198.51.100.5:80     5   742
  URL:"/tfcsirt2017/"  Host:"nemea.liberouter.org" Method:GET
  User-Agent:"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.1"

2015-06-22 12:34:56.567   1.502  TCP  198.51.100.5:80   ->    192.0.2.82:8420  10  2685
  ResponseCode:200  ContentType:"text/html"

2015-06-22 12:34:57.222   0.241  TCP    192.0.2.45:4571 -> 203.0.113.100:5060   3   540
  MessageType:INVITE  From:"me@example.com"  To:"you@example.org"  CallID:"1a2f345ef97b"
```

L7 extended flows – how to

L7 extended flow – how to

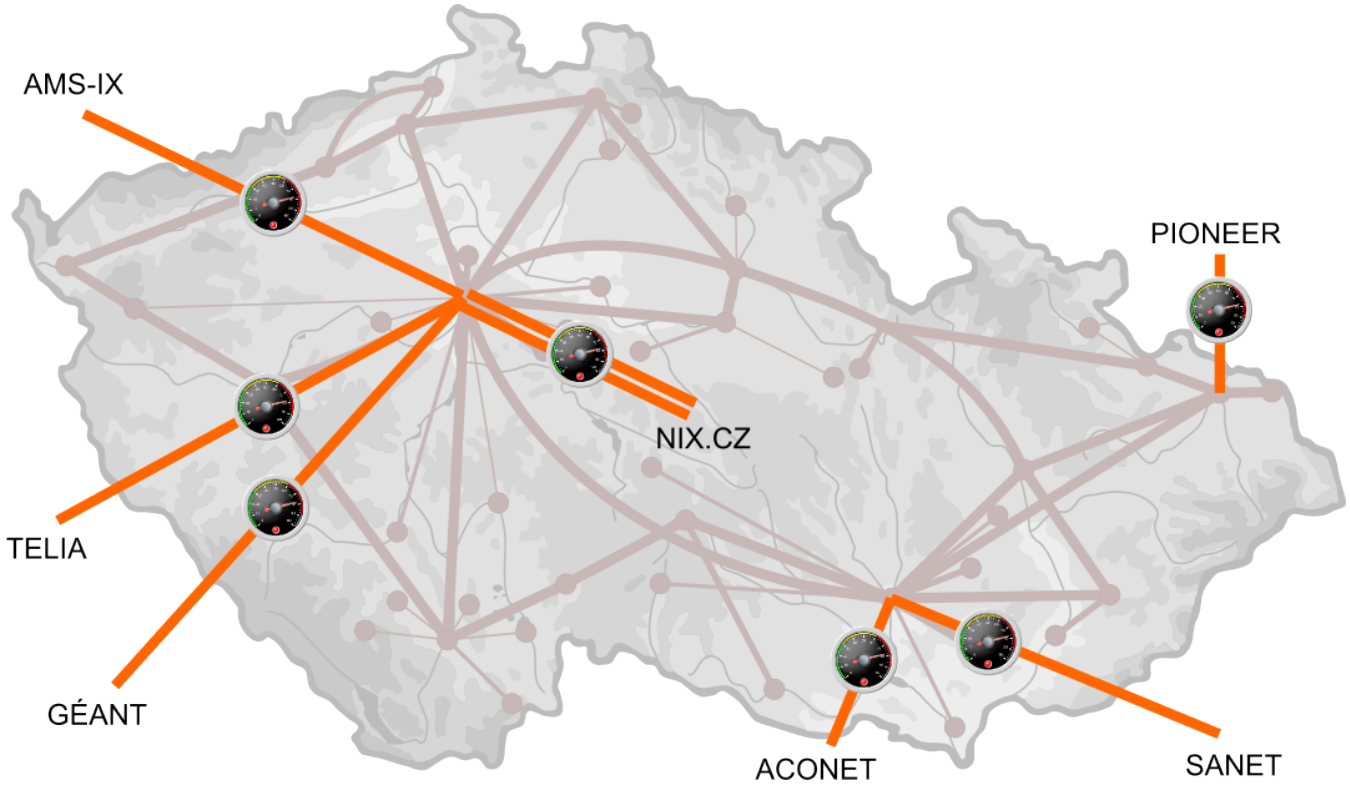
- Protocol: only IPFIX is flexible enough to transfer any such data
- Exporter: must support parsing application protocols
  - Usually via plugins
  - FlowMon
  - YAF
  - ...
- Collector: must support IPFIX including non-standard fields
  - IPFIXcol
  - AnalysisPipeline (SiLK)
  - ...

Section 4

Monitoring infrastructure at CESNET

CESNET monitoring infrastructure

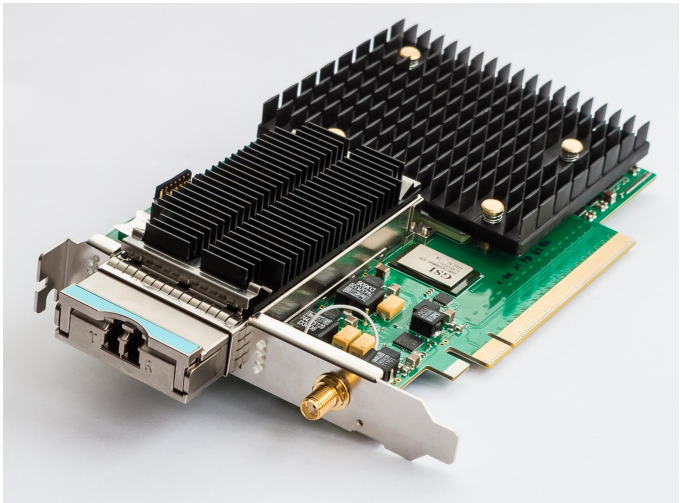
- Dedicated probes on all external links

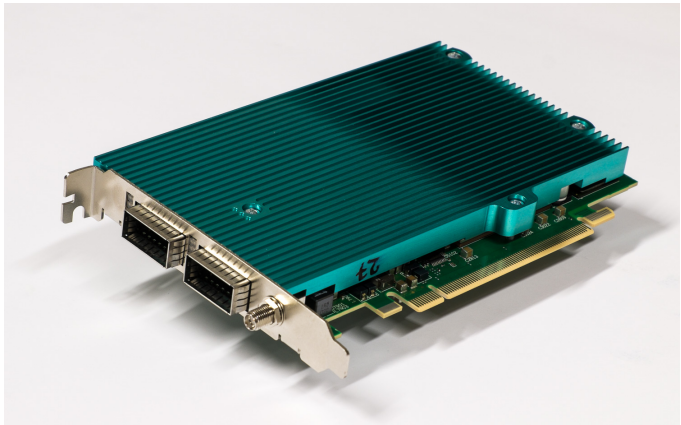


CESNET monitoring infrastructure

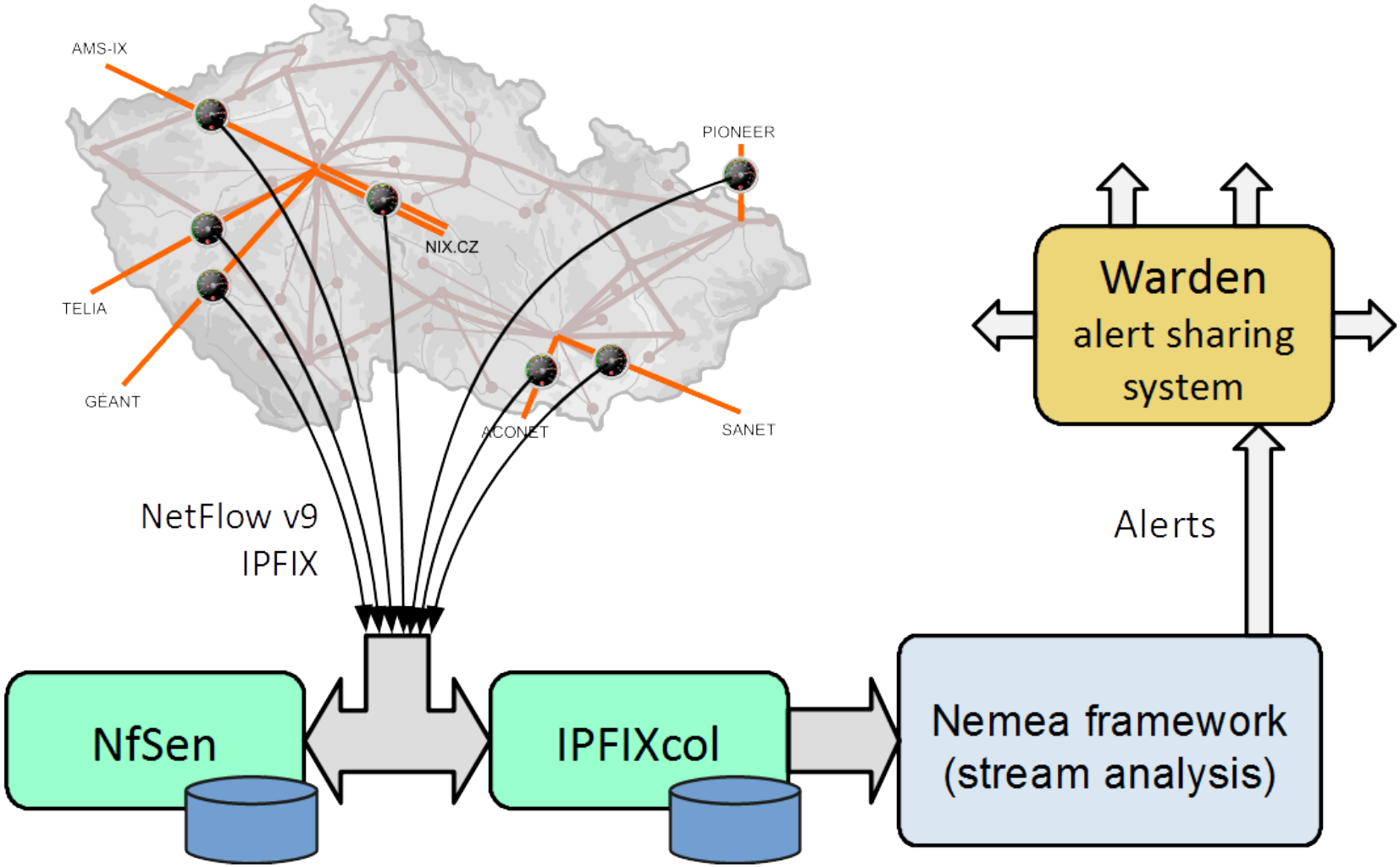
Probes

- Servers with special HW acceleration card
- SW: FlowMon exporter
  (by FlowMon Technologies, formerly INVEA-TECH)
- Throughput up to full 100Gbps
- Plugins for parsing HTTP, DNS, SMTP, VOIP, tunnels





CESNET monitoring infrastructure



Part II

Flow Data Querying

Section 1

General data queries

Data querying

- Data stored by a collector may be queried
  - Manually or automatically
  - Statistics
  - Traffic of particular IP addresses
  - Search for particular traffic patterns
- Security analysis – search for malicious traffic
- This section is about
  - how to query flow data
  - how to interpret results

Data querying

No matter whether we use nfdump, fbitdump, or something else, a query consists of the following:

- Data selection
  - One or more time intervals (5 min)
  - One or more data sources (probes, routers, ODIDs)
- Filtering
- Aggregation
- Sort
- (aggregation + sort -> Top-N stats)

```
nfdump -M /data/nfsen/profiles-data/live/probe1:probe2:probe3
   -r 2016/04/04/nfcapd.201604040800 -o long -c 100
   "src port 80 and bytes > 10000"
```

Data querying – examples

Filter – Google DNS

- Filter
  - "proto udp and port 53 and ip 8.8.8.8"
- Aggregate
  - –

```
nfdump -M /data/nfsen/profiles-data/live/probe1:probe2:probe3
   -r 2016/04/14/nfcapd.201604141305 -c 20
   "proto udp and port 53 and ip 8.8.8.8"
```

Data querying – examples

Filter – Google DNS

```
nfdump -M /data/nfsen/profiles-data/live/probe1:probe2:probe3 -r 2016/04/14/nfcapd.201604141305 -c 20
   "proto udp and port 53 and ip 8.8.8.8"
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Packets	Bytes
2016-04-14 13:04:58.613	0.000	UDP	8.8.8.8:53	->	194.xxx.yy.11:7433	1	65
2016-04-14 13:05:04.376	0.000	UDP	8.8.8.8:53	->	194.xxx.yy.11:13154	1	65
2016-04-14 13:04:54.990	0.000	UDP	8.8.8.8:53	->	138.xxx.yy.153:48971	1	113
2016-04-14 13:04:48.060	0.000	UDP	193.xxx.yyy.155:50391	->	8.8.8.8:53	1	78
2016-04-14 13:04:47.904	0.000	UDP	8.8.8.8:53	->	193.xxx.yyy.155:50391	1	110
2016-04-14 13:04:21.014	0.000	UDP	8.8.8.8:53	->	193.xxx.yyy.68:14295	1	116
2016-04-14 13:04:20.594	0.000	UDP	129.xx.yy.254:59812	->	8.8.8.8:53	1	70
2016-04-14 13:04:23.179	0.000	UDP	193.xxx.yyy.197:59427	->	8.8.8.8:53	1	60
2016-04-14 13:04:55.997	0.000	UDP	138.xxx.yy.153:37370	->	8.8.8.8:53	1	63
2016-04-14 13:04:55.998	0.000	UDP	138.xxx.yy.153:49595	->	8.8.8.8:53	1	64
2016-04-14 13:04:56.007	0.000	UDP	138.xxx.yy.153:59634	->	8.8.8.8:53	1	58
2016-04-14 13:04:19.380	0.000	UDP	8.8.8.8:53	->	193.xxx.yyy.12:2156	1	175
2016-04-14 13:04:21.767	0.000	UDP	193.xxx.yyy.150:17691	->	8.8.8.8:53	1	62
2016-04-14 13:04:24.476	0.000	UDP	8.8.8.8:53	->	193.xxx.yyy.203:22416	1	143
2016-04-14 13:04:20.605	0.000	UDP	8.8.8.8:53	->	193.xxx.yyy.98:59678	1	247
2016-04-14 13:04:24.112	0.000	UDP	8.8.8.8:53	->	193.xxx.yyy.232:65426	1	141
2016-04-14 13:04:23.178	0.000	UDP	8.8.8.8:53	->	193.xxx.yyy.222:57272	1	147
2016-04-14 13:04:22.184	0.000	UDP	8.8.8.8:53	->	78.xxx.yy.131:50037	1	114
2016-04-14 13:04:19.963	0.000	UDP	8.8.8.8:53	->	193.xxx.yyy.77:35274	1	245
2016-04-14 13:04:56.017	0.000	UDP	8.8.8.8:53	->	138.xxx.yy.153:37370	1	120
2016-04-14 13:04:55.986	0.000	UDP	78.xxx.yyy.50:59896	->	8.8.8.8:53	1	69

Data querying – examples

Aggregation – most active IP addresses

- Filter
  - –
- Aggregate
  - -s srcip/flows

```
nfdump -M /data/nfsen/profiles-data/live/probe1:probe2:probe3
   -r 2016/04/14/nfcapd.201604141305 -s ip/flows -n 10
```

Data querying – examples

Aggregation – most active IP addresses

```
nfdump -M /data/nfsen/profiles-data/live/probe1:probe2:probe3 -r 2016/04/14/nfcapd.201604141305
   -s ip/flows -n 10
```

Date first seen	Duration	Proto	IP Addr	Flows(%)	Packets(%)	Bytes(%)
2016-04-14 12:59:27.392	606.310	any	195.113.144.201	123804(3.5)	165014(0.2)	12.7 M(0.0)
2016-04-14 13:00:12.561	567.826	any	208.67.222.222	57203(1.6)	57862(0.1)	6.6 M(0.0)
2016-04-14 13:03:57.453	335.505	any	192.33.14.30	51875(1.5)	62219(0.1)	22.4 M(0.0)
2016-04-14 13:03:57.584	337.865	any	194.0.14.1	47773(1.3)	47797(0.0)	8.3 M(0.0)
2016-04-14 13:02:13.219	446.170	any	195.113.144.194	43218(1.2)	43865(0.0)	5.3 M(0.0)
2016-04-14 13:02:41.096	389.270	any	195.xxx.yyy.66	33175(0.9)	366681(0.4)	288.2 M(0.3)
2016-04-14 13:03:29.245	364.119	any	195.xxx.yyy.90	30598(0.9)	31173(0.0)	3.1 M(0.0)
2016-04-14 13:03:57.623	312.604	any	89.xxx.yy.159	29460(0.8)	29460(0.0)	2.2 M(0.0)
2016-04-14 13:01:58.399	432.986	any	217.xx.yy.34	26727(0.7)	115520(0.1)	14.3 M(0.0)
2016-04-14 13:03:58.837	311.482	any	109.xxx.y.233	24669(0.7)	24670(0.0)	1.9 M(0.0)

Data querying – examples

Aggregation – most active IP addresses (anomaly)

```
nfdump -M /data/nfsen/profiles-data/live/probe4:probe5 -r 2016/04/14/nfcapd.201604141305
   -s ip/flows -n 10
```

Date first seen	Duration	Proto	IP Addr	Flows(%)	Packets(%)	Bytes(%)
2016-04-14 13:00:05.248	499.382	any	150.xxx.yyy.114	455743(20.0)	493728(0.8)	225.0 M(0.5)
2016-04-14 12:59:36.486	608.560	any	31.xx.yy.4	94245(4.1)	3.4 M(5.7)	259.3 M(0.6)
2016-04-14 13:04:19.373	337.318	any	148.xx.yyy.185	66138(2.9)	66153(0.1)	7.8 M(0.0)
2016-04-14 12:59:31.262	624.145	any	31.xx.yy.8	48086(2.1)	715150(1.2)	243.5 M(0.6)
2016-04-14 12:59:47.927	607.310	any	31.yy.yy.36	35036(1.5)	1.3 M(2.2)	429.0 M(1.0)
2016-04-14 13:04:19.652	303.619	any	89.xx.yyy.242	32791(1.4)	32791(0.1)	7.2 M(0.0)
2016-04-14 13:04:19.511	337.131	any	149.xxx.y.252	28754(1.3)	30604(0.1)	9.3 M(0.0)
2016-04-14 12:59:40.094	617.527	any	62.xx.yy.73	20696(0.9)	1.4 M(2.4)	858.1 M(2.0)
2016-04-14 13:00:00.254	581.267	any	31.xx.yy.2	19822(0.9)	202437(0.3)	19.6 M(0.0)
2016-04-14 13:03:48.183	335.128	any	150.xxx.yyy.97	19024(0.8)	157087(0.3)	83.7 M(0.2)

Data querying – examples

Traffic of the most active IP address

- Filter
  - "ip 150.xxx.yyy.114"
- Aggregate
  - –

```
nfdump -M /data/nfsen/profiles-data/live/probe1:probe2:probe3
   -r 2016/04/14/nfcapd.201604141305 -c 20 "ip 150.xxx.yyy.114"
```

Data querying – examples

Traffic of the most active IP address

```
nfdump -M /data/nfsen/profiles-data/live/probe1:probe2:probe3 -r 2016/04/14/nfcapd.201604141305 -c 20
   "ip 150.xxx.yyy.114"
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Packets	Bytes
2016-04-14 13:04:45.929	0.000	UDP	178.xxx.yyy.90:53	->	150.xxx.yyy.114:36116	1	256
2016-04-14 13:04:45.907	0.000	UDP	85.xx.y.205:53	->	150.xxx.yyy.114:49340	1	531
2016-04-14 13:04:45.867	0.000	UDP	5.x.yy.56:53	->	150.xxx.yyy.114:17957	1	531
2016-04-14 13:04:45.844	0.000	UDP	77.xxx.yyy.91:53	->	150.xxx.yyy.114:63763	1	531
2016-04-14 13:04:45.842	0.000	UDP	46.x.yy.141:53	->	150.xxx.yyy.114:27696	1	531
2016-04-14 13:04:45.816	0.000	UDP	213.xxx.yyy.140:53	->	150.xxx.yyy.114:25738	1	537
2016-04-14 13:04:45.948	0.000	UDP	213.xxx.yyy.64:53	->	150.xxx.yyy.114:58007	1	528
2016-04-14 13:04:45.785	0.000	UDP	213.xxx.yy.182:53	->	150.xxx.yyy.114:46788	1	528
2016-04-14 13:04:45.756	0.000	UDP	193.xxx.yyy.250:53	->	150.xxx.yyy.114:2236	1	524
2016-04-14 13:04:45.694	0.000	UDP	131.xxx.yy.199:53	->	150.xxx.yyy.114:41361	1	531
2016-04-14 13:04:45.616	0.000	UDP	5.x.yyy.232:53	->	150.xxx.yyy.114:26361	1	532
2016-04-14 13:04:45.614	0.000	UDP	85.xx.yyy.165:53	->	150.xxx.yyy.114:54306	1	528
2016-04-14 13:04:45.656	0.000	UDP	213.xxx.yyy.2:53	->	150.xxx.yyy.114:61620	1	528
2016-04-14 13:04:45.727	0.000	UDP	213.xxx.yyy.181:53	->	150.xxx.yyy.114:14064	1	528
2016-04-14 13:04:45.538	0.000	UDP	195.x.yyy.62:53	->	150.xxx.yyy.114:13306	1	529
2016-04-14 13:04:45.464	0.000	UDP	88.xxx.yy.137:53	->	150.xxx.yyy.114:39351	1	531
2016-04-14 13:04:45.419	0.000	UDP	213.xxx.yyy.38:53	->	150.xxx.yyy.114:29591	1	531
2016-04-14 13:04:45.460	0.000	UDP	213.xxx.yy.173:53	->	150.xxx.yyy.114:25308	1	528
2016-04-14 13:04:45.336	0.000	UDP	213.xxx.yyy.166:53	->	150.xxx.yyy.114:34860	1	256
2016-04-14 13:04:45.361	0.000	UDP	91.xxx.yy.15:53	->	150.xxx.yyy.114:6768	1	439

Data querying – examples

Port scanning – most active scanners

- Filter
  - "proto tcp and flags S and not flags ARFPU"
- Aggregate
  - -S srcip/flows

```
nfdump -M /data/nfsen/profiles-data/live/probe4:probe5
   -r 2016/04/14/nfcapd.201604141305
   "proto tcp and flags S and not flags ARFPU" -s ip/flows -n 10
```

Data querying – examples

Port scanning – most active scanners

```
nfdump -M /data/nfsen/profiles-data/live/probe4:probe5 -r 2016/04/14/nfcapd.201604141305
   "proto tcp and flags S and not flags ARFPU" -s ip/flows -n 10
```

Date first seen	Duration	Proto	IP Addr	Flows(%)	Packets(%)	Bytes(%)
2016-04-14 13:04:18.962	78.602	any	89.xxx.yyy.192	192503(14.1)	192503(7.7)	7.7 M(6.0)
2016-04-14 13:04:19.077	308.520	any	80.xx.yy.38	28789(2.1)	28789(1.2)	1.2 M(0.9)
2016-04-14 13:04:19.003	18.806	any	89.xxx.yyy.196	20991(1.5)	20991(0.8)	839640(0.7)
2016-04-14 13:02:52.971	326.121	any	58.xxx.yyy.108	20301(1.5)	40551(1.6)	2.4 M(1.9)
2016-04-14 13:02:55.764	393.043	any	216.xxx.yy.2	20173(1.5)	20173(0.8)	806920(0.6)
2016-04-14 13:03:21.995	294.595	any	216.xxx.yyy.124	16264(1.2)	16264(0.7)	650560(0.5)
2016-04-14 13:04:18.873	339.138	any	176.xx.yy.206	13040(1.0)	49617(2.0)	3.0 M(2.3)
2016-04-14 13:04:54.735	273.714	any	74.xx.yy.10	12436(0.9)	12436(0.5)	497440(0.4)
2016-04-14 13:04:11.880	346.159	any	88.xxx.yyy.73	12120(0.9)	34305(1.4)	2.1 M(1.6)
2016-04-14 13:02:53.064	395.438	any	191.xxx.yy.33	10067(0.7)	19570(0.8)	1.0 M(0.8)

Data querying – examples

Who communicated with botnet CC server

- Filter
  - "dst ip 6.6.6.6"
- Aggregate
  - -A srcip
- Long time frame

```
nfdump -M /data/nfsen/profiles-data/live/probe1:probe2:probe3
   -R 2016/04/14/nfcapd.201604140000:2016/04/14/nfcapd.201604140555
   "dst ip 6.6.6.6" -A srcip
```

Data querying – examples

Who communicated with botnet CC server

```
nfdump -M /data/nfsen/profiles-data/live/probe1:probe2:probe3
   -R 2016/04/14/nfcapd.201604140000:2016/04/14/nfcapd.201604140555
   "dst ip 6.6.6.6" -A srcip
```

Date first seen	Duration	Src IP Addr	Packets	Bytes	bps	Bpp	Flows
2016-04-14 03:00:58.268	146.505	147.xx.yyy.221	315	100444	5484	318	6
2016-04-14 05:34:47.713	63.516	195.xxx.yyy.77	184	27540	3468	149	10
2016-04-14 01:22:29.027	90.600	147.xx.yyy.253	6	632	55	105	3
2016-04-14 00:00:15.716	7454.390	147.xx.yy.154	504	107689	115	213	14
2016-04-14 03:04:00.807	1029.272	128.xxx.yy.204	22	2139	16	97	10
2016-04-14 00:10:58.935	114.058	147.xx.yy.64	83	43723	3066	526	13
2016-04-14 00:26:02.829	490.486	195.xxx.yyy.30	416	99237	1618	238	65
2016-04-14 01:48:04.939	170.695	195.xxx.yyy.150	385	82478	3865	214	34
2016-04-14 03:30:49.605	472.525	147.xx.yyy.239	291	158224	2678	543	31
2016-04-14 00:20:39.712	161.666	147.xx.yyy.249	206	99439	4920	482	22
2016-04-14 01:09:36.894	102.709	147.xx.yyy.32	69	21311	1659	308	11
2016-04-14 01:30:37.793	123.409	147.xx.yyy.243	136	73766	4781	542	18

Section 2

SecurityCloud GUI

SecurityCoud GUI

Alternative to nfsen, *work in progress!*

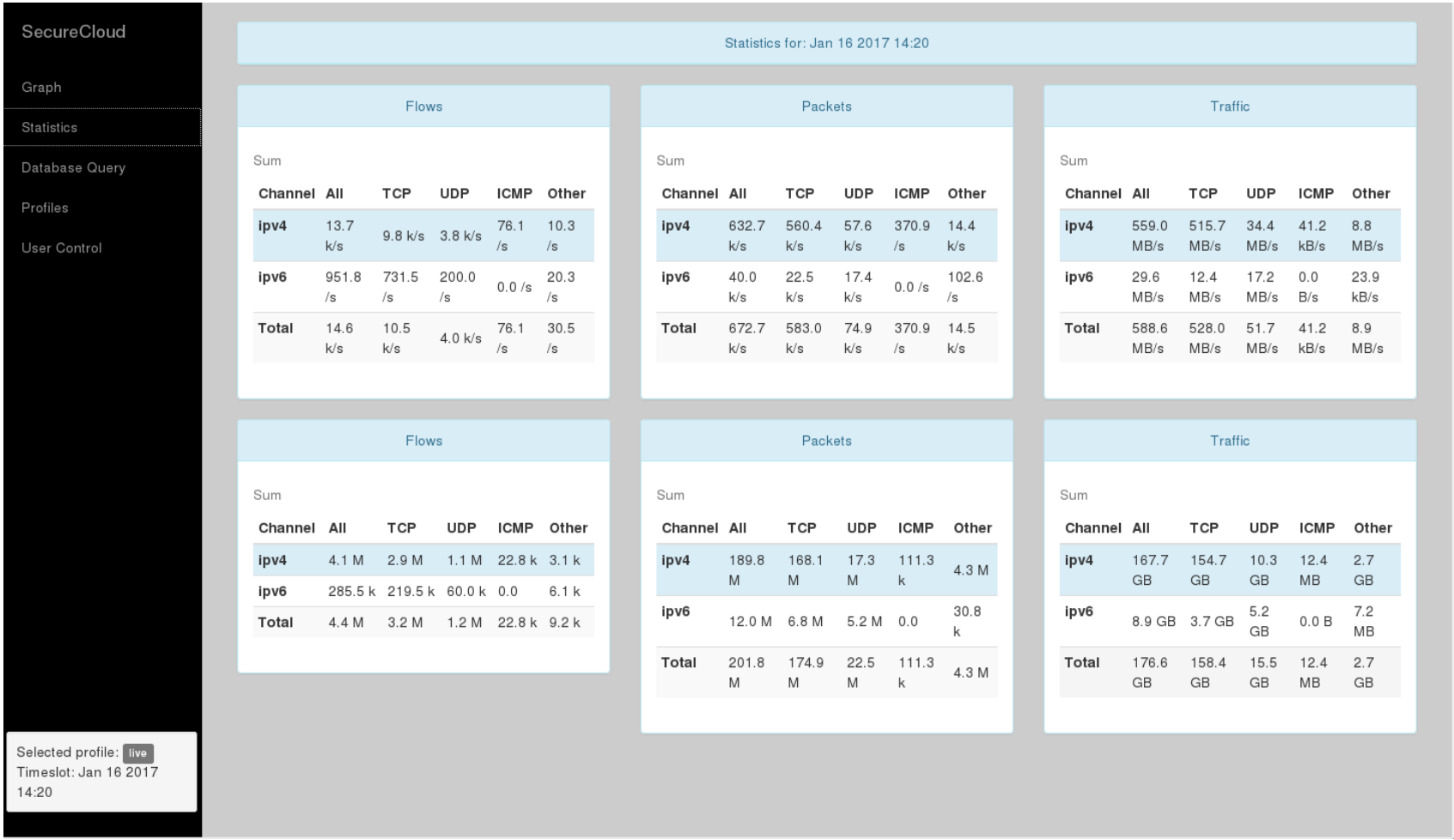
SC GUI provides:

- Traffic graphs
- Statistics
- Profiles
- Parallel queries

Demo at `http://localhost/scgui/`

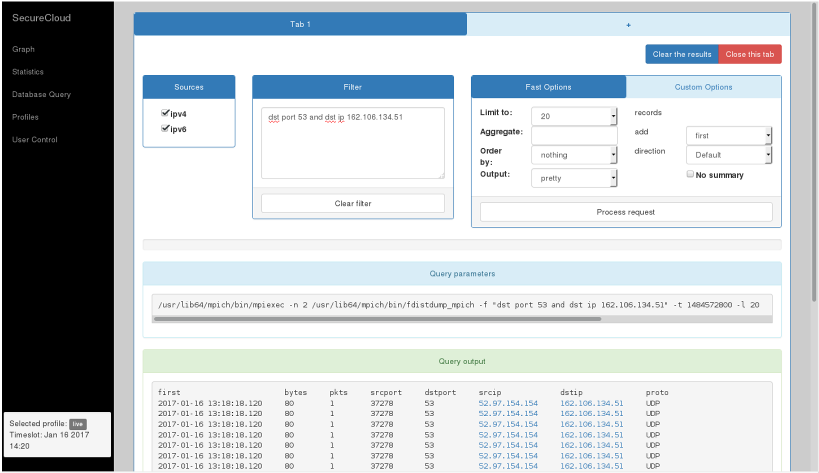
SecurityCoud GUI - Graphs

SecurityCoud GUI - Statistics



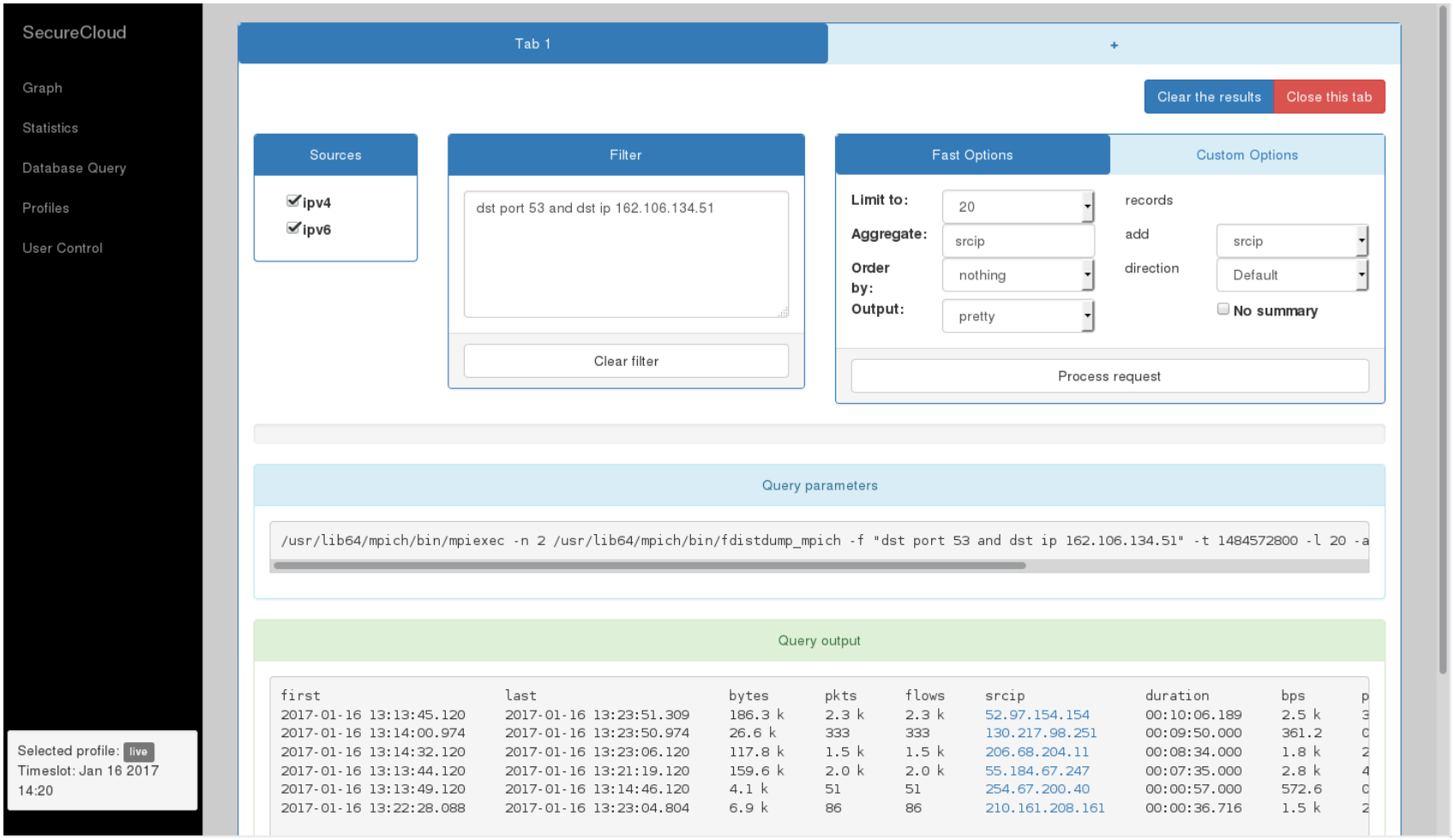
SecurityCoud GUI - Queries I

dst port 53 and dst ip 162.106.134.51



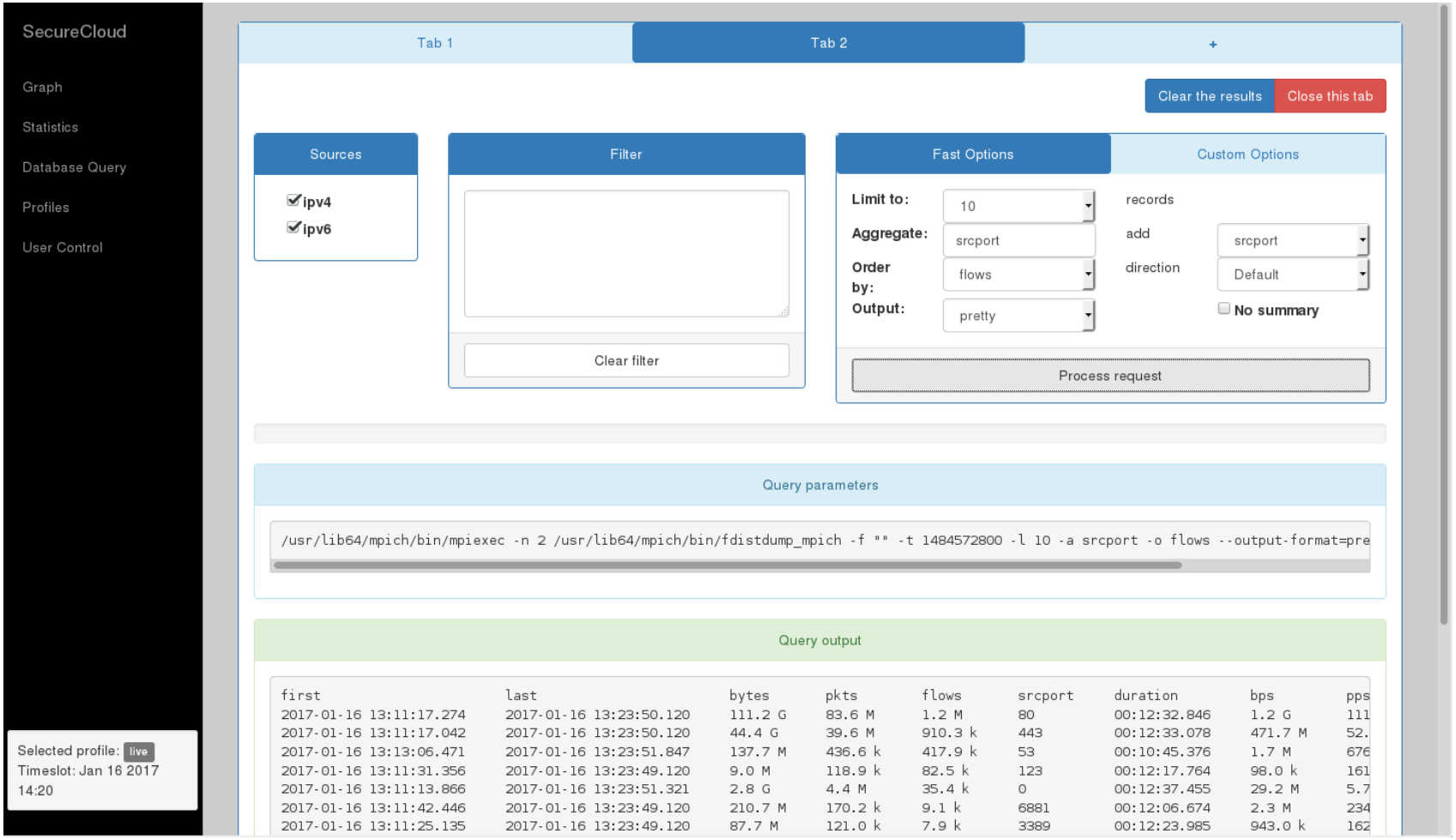
SecurityCoud GUI - Queries II

dst port 53 and dst ip 162.106.134.51, aggregated by source IP



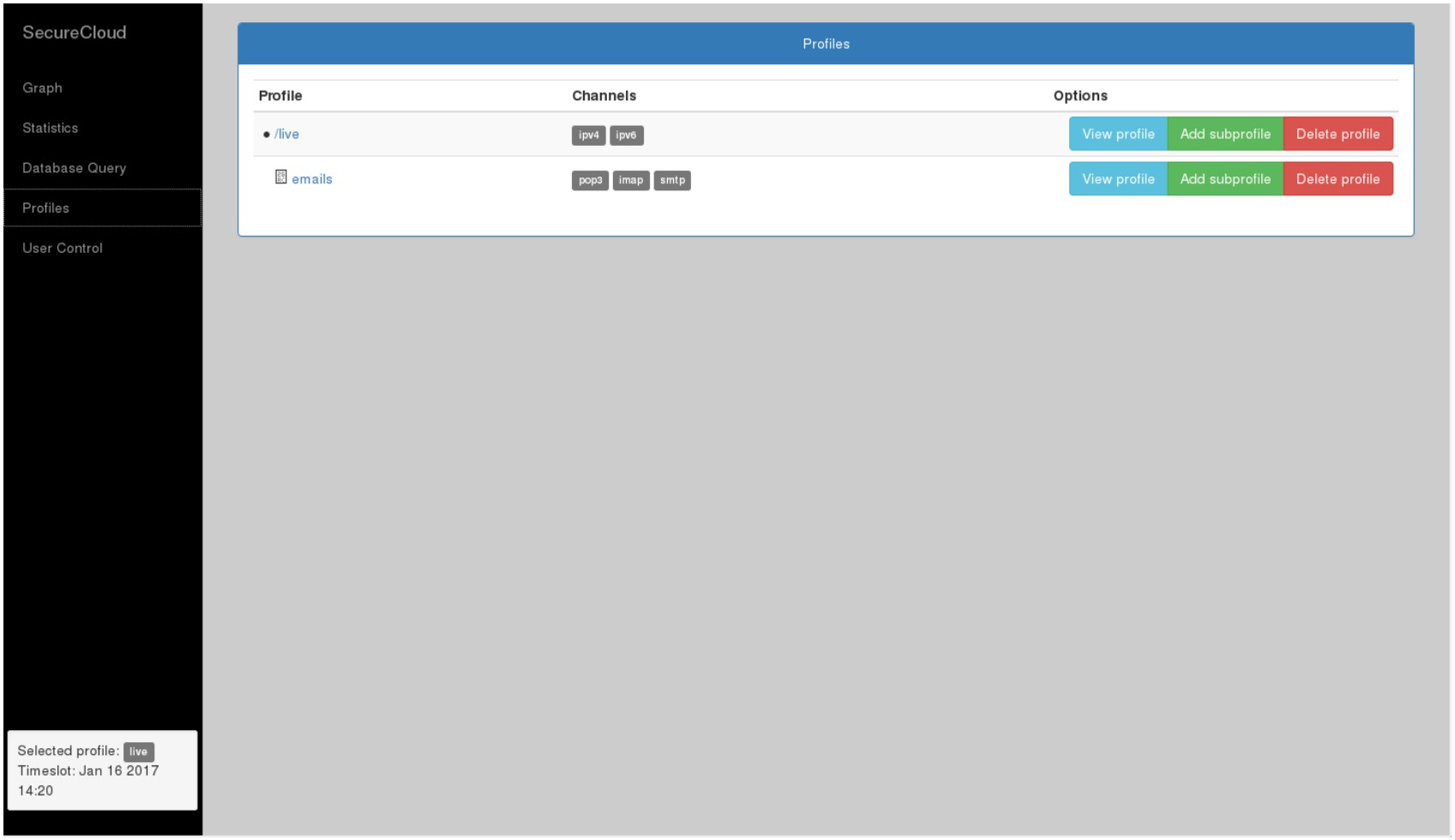
SecurityCoud GUI - Queries III

In a second tab: aggregation by source port ordered by flows



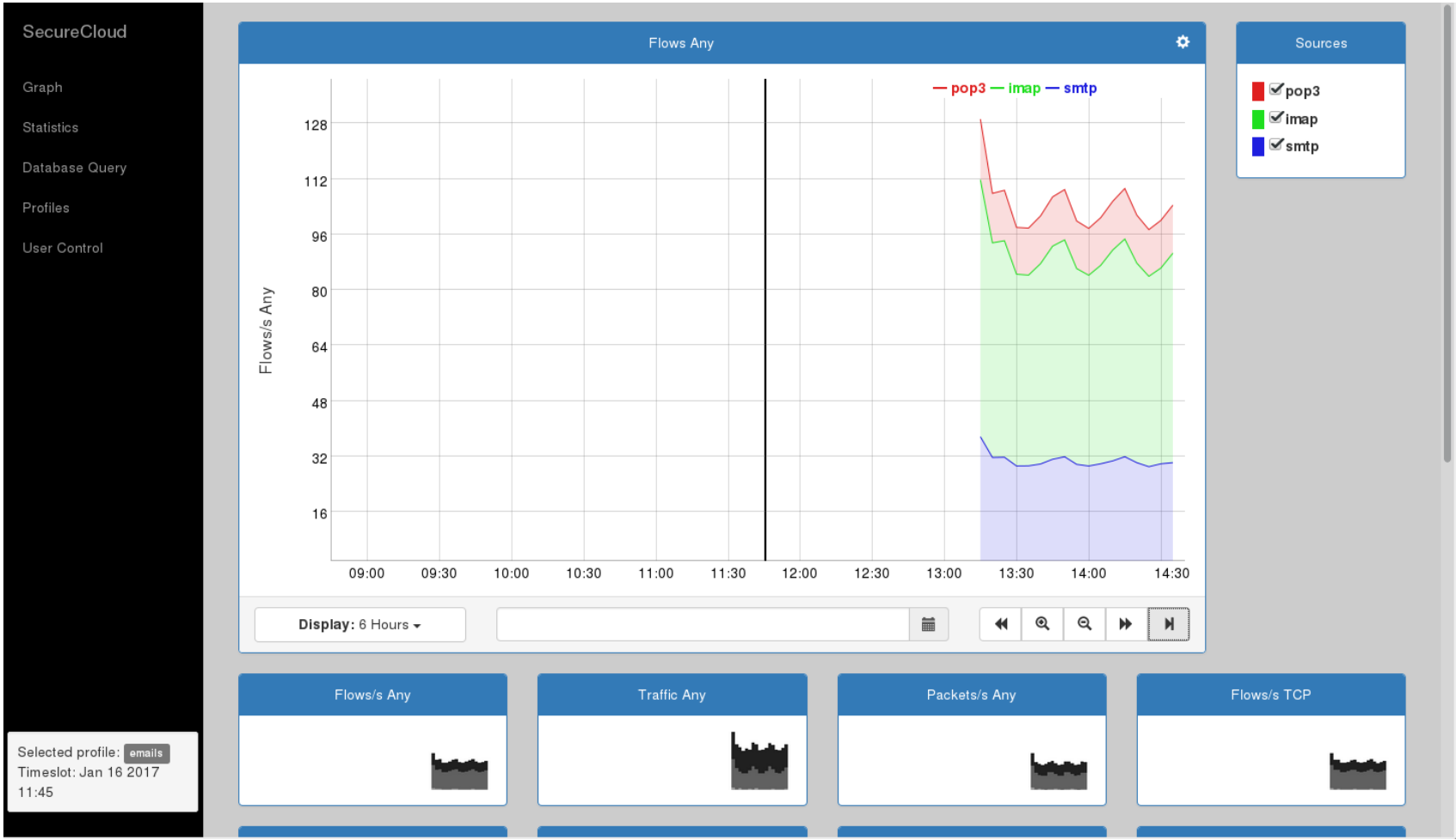
SecurityCoud GUI - Profiles I.

Create and select different profiles



SecurityCoud GUI - Profiles II.

Profiles metadata are stored in RRDs



Part III

IPFIXcol (Overview and Launching)

Section 1

IPFIXcol

IPFIXcol architecture

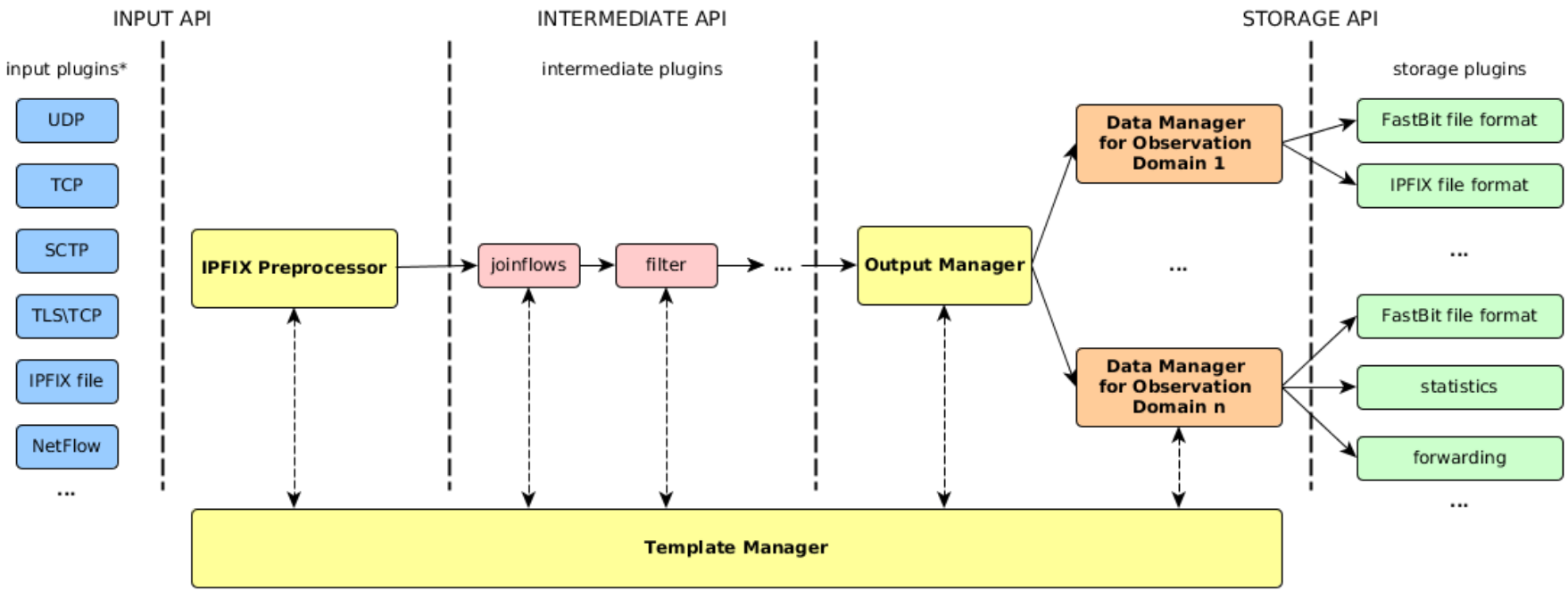
IPFIXcol

- RFC7011 Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information
- IPFIX is a native protocol for the collector
- `https://github.com/CESNET/ipfixcol/`

Modular architecture

- Plugins for data reception (input plugins), manipulation (intermediate plugins), and output (storage plugins)

IPFIXcol architecture



* Select one for IPFIX Preprocessor

IPFIXcol plugins

IPFIXcol provides an interface to write new plugins that extend its functionality

Existing input plugins

TCP, UDP, SCTP Plugins that can receive data using the common protocols. They can also convert NetFlow v5 and v9 to IPFIX.

IPFIX file Plugin that can read IPFIX file format

nfdump Plugin that allows to process data stored by nfdump

IPFIXcol

Existing intermediate plugins

GeoIP	Plugin for performing geolocation of the flows based on destination and source IP addresses
Anonymization	Plugin for IP address anonymization. Uses Crypto-PAn or data truncation for the anonymization.
Filter	Filters flow records based on values of individual elements
Hooks	Calls external programs on certain events, such as when an exporter connects or disconnects
JoinFlows	Allow to merge data from different Observation Domain IDs to single ODID
ODIP	Adds IP address of exporter to flow records

IPFIXcol plugins

Existing output plugins

Forwarding	Allows to send data to other collectors. Also supports round robin data distribution
IPFIX	Stores data in IPFIX file format
JSON	Converts flow records to JSON documents. Useful for connecting to big data analysis tools
PostgreSQL	Stores data in PostgreSQL database
nfdump	Stores data in nfdump format
FastBit	Stores data in FastBit format. FastBit is a noSQL column database with support for fast indexing
UniRec	Sends data using UniRec format. This plugin is used to pass data to the Nemea framework

Running IPFIXcol

Configuration

- IPFIXcol stores its configuration in the `/etc/ipfixcol/` directory.
- `ipfix-elements.xml` contains a description of the known IPFIX elements assigned by IANA `http://www.iana.org/assignments/ipfix/ipfix.xml`.
- `internalcfg.xml` contains configuration of plugins used in `startup.xml`. Can be viewed/edited with ipfixconf tool.
- `startup.xml` describes how IPFIXcol is configured at startup, which plugins are used and where the data will be stored.
- Path to every configuration file can be provided using command line switch

Running IPFIXcol

Statistics

- IPFIXcol can print runtime statistics to either stdout or files
- Following direction in the `<collectingProcess>`:

```
<statisticsFile>
    /tmp/ipfixcol_stat.log
</statisticsFile>
```

- Shows number of processed packets and flows, CPU utilization for each thread and other useful information

Running IPFIXcol

Reconfiguration

- Collector can be reconfigured at runtime by sending SIGUSR1 signal. When this signal is received, startup configuration is reloaded and chages are processed.
- Reconfiguration can:
  - Change input plugin
  - Add/remove intermediate plugin(s)
  - Add/remove storage plugin(s)
  - Change plugin settings (plugin is reloaded)
  - Reorder intermediate plugins (they're removed and loaded in the new order)

Section 2

IPFIXcol Hands-On

IPFIXcol Hands-On

Task 1

- Starting up the IPFIXcol
- Sending data to IPFIXcol
- Using statistics

Task 1 - Starting up the IPFIXcol

Startup configuration in `startup-task1.xml`
(in `/home/nemea/data/IPFIXcol/`)

1. Where to listen for data: collectingProcess
2. What to do with the data: exportingProcess
3. Data transformation and processing: intermediatePlugins

Prepare dataset:
`cd /home/nemea/data/IPFIXcol`

Run: `ipfixcol -c startup-task1.xml -v2`

1. Startup process is reported in verbose level INFO (-v2 parameter)
2. Use Ctrl+C to terminate the collector
3. More options available, see ipfixcol -h

Task 1 - Starting up the IPFIXcol

Run: `ipfixcol -c startup-task1.xml -v2 -S 10`

1. Prints statistics every 10 seconds
2. Leave it running

Sending data to the IPFIXcol

1. In another terminal run:

```
ipfixsend -i data.ipfix -d 127.0.0.1 \
 -t TCP -p 4739 -S 5000 -n 1
```

2. Starts sending 5000 IPFIX packets per second to the collector. End after replaying the source file once
3. Switch to terminal with IPFIXcol to see statistics
4. Notice reports by the Hook plugin

IPFIXcol Hands-On

Task 2

- Writing flows in JSON to file
- Sending JSON data over network

Task 2 - Writing flows in JSON to file

Writing flows in JSON to file

1. `ipfixcol -c startup-task2.1.xml -v2 -S 10`
2. `ipfixsend -i data.ipfix -d 127.0.0.1 -t TCP \ -p 4739 -n 1`
3. Results stored in `/tmp/json/...`
4. Arbitrary file rotation
5. Useful for feeding stored static data to database

Task 2 - Sending data over network

Sending data over network

1. `ipfixcol -c startup-task2.2.xml -v2 -S 10`
   - Sends data to `localhost:4444` over UDP
2. See flows using `nc -u -l 4444 | head -n 1`
3. `ipfixsend -i data.ipfix -d 127.0.0.1 -t TCP \ -p 4739 -n 1`
4. Names of elements come from `/etc/ipfixcol/ipfix-elements.xml`
5. Useful for feeding stream data processing tools

IPFIXcol Hands-On

Task 3

- Saving data to FastBit database

Task 3 - Saving data to FastBit database

Saving data to FastBit database

1. `ipfixcol -c startup-task3.xml`
2. `ipfixsend -i data.ipfix -d 127.0.0.1 -t TCP \ -p 4739 -n 1`
3. Ctrl+C - terminate the collector
4. Saves data to `/tmp/fastbit/...`
5. Time rotation, each IPFIX template is a directory, each file an IPFIX element

Section 3

FastBit database (fbitdump)

FastBit Database

`https://sdm.lbl.gov/fastbit/`

NoSQL, column oriented

- has SELECT, WHERE, GROUP BY, basic aggregation functions
- limited JOIN
- Tables are directories, columns are files

Data types

- 8, 16, 32, 64 bit signed and unsigned integers BLOBs, strings

Indexes

- compressed bitmap indexes
- efficient search and retrieval operations
- slower update

IPFIX Data in FastBit

- Need to map IPFIX data format to FastBit datase schema
- Separate data based on time windows
- IPFIX templates
  - Each template is a directory
  - Each IPFIX element is stored in a column of appropriate type
- Data type conversion
  - Numbers are easy
  - IPv6 addresses - two 64bit numbers
  - MAC addresses - 64bit, unsused two bytes
  - ...

fbitdump Query Tool

fbitdump

- Tool for querying IPFIX data in FastBit database
- Support for network related data types
- Many formatting options
- `https://github.com/CESNET/ipfixcol/tree/master/tools/fbitdump`

fbitdump Configuration

Configuration

- fbitdump takes configuration from `/usr/(local/)share/fbitdump/fbitdump.xml`
- Definition of displayed columns (plain and derived)
- Definition of column groups for easier querying
- Summary columns
- Predefined output formats
- Semantic plugins for data formatting

fbitdump Features

Query types

- Filtering
- Aggregation and statistics
- Sorting

Output formatting

- Predefined formats
- Custom format using -o"fmt:%aliases"

Plugins

- Simple plugins for work with specific data types
- Function for printing formatted database
- Function for parsing formatted query strings
- HTTP request types, status codes, MAC addresses, ...

Section 4

fbitdump Hands-On

FastBit Queries

Task 1

Working with fbitdump output format

Task 1 - Working with fbitdump output format

1. Try basic query to list first 10 records:

   ```
   fbitdump -R /tmp/fastbit/ -c 10
   ```

2. The default output format is called “line”. You can change output format using -o switch. Try the same output format with IPv6 addresses only:

   ```
   fbitdump -R /tmp/fastbit/ -c 10 -o line6
   ```

3. There are many predefined output formats. Use `fbitdump -O` to list all available formats. Format name is on the left, used format string is on the right.

Task 1 - Working with fbitdump output format

4. User can specify their own output format by using `-o "fmt: ..."`. Custom format string must be specified after the `fmt:` keyword. The `line6` output can be achieved by following command:

```
fbitdump -R /tmp/fastbit/ -c 10 \
-o "fmt: %ts %td %pr %sa6:%sp -> "\
"%da6:%dp %pkt %byt %fl"
```

5. Frequently used custom formats can be easily named and stored in configuration file for future use. See section `<output>` in `/usr/share/fbitdump/fbitdump.xml`

FastBit Queries

Task 2

Working with IPFIX templates and FastBit tables

Task 2 - Working with IPFIX templates and FastBit tables

1. IPFIX templates describe different data structures. fbitdump allows users to list stored data structures using `-T` option. Use
   `fbitdump -R /tmp/fastbit/ -T | less`
2. Each template is described in the output. If a column is defined in the `fbitdump.xml` configuration file, more information about the stored element is available. It is very useful to see what data is stored and which columns are available.
3. You can see that element `e39499id51` is not defined yet. Open the `/usr/share/fbitdump/fbitdump.xml` in an editor and uncomment the last `<column>` in `<columns>` definition (line 788 and below). List the templates again. You should see the element `e39499id51` defined now.
4. Optionally, you can extend definition of `voip` and `sip` output formats to include the `%sipua` column.

FastBit Queries

Task 3

Data filtering with fbitdump

Task 3 - Data filtering with fbitdump

1. We have learned to explore available data formats and output records in desired format. However, listing all data is impractical. One way to limit output is simply to use -c switch to limit number of printed records. However, records can also be filtered based on values of individual elements.
2. List available IPv6 records with HTTP path set:

```
fbitdump -R /tmp/fastbit/ -o http6 \
"EXISTS %httpp and EXISTS %sa6" -c 50
```

3. There is a lot of these records. To see how many, just add -A option. This option causes the fbitdump to aggeragate all lines. Without any arguments, it provides useful statistics. You can see that there are 56585 records matching the filter.

Task 3 - Data filtering with fbitdump

4. Let us look for more unusual traffic. Filter out common HTTP traffic on port 80. The filter should be the following:

   ```
   "EXISTS %httpp and EXISTS %sa6 and %port != 80"
   ```

5. There are still too many records. Communication on port 443 is also considered to be in the HTTP category. Let us filter out traffic on this port as well:

   ```
   "EXISTS %httpp and EXISTS %sa6 and %port != 80
   and %port != 443"
   ```

Task 3 - Data filtering with fbitdump

6. There are 740 records left. We can see that request type for all of them has non-zero value, therefore all of these records describe some kind of HTTP request. Request type 11 means that the traffic was HTTPS. The host value for HTTPS is actually taken from TLS handshake SNI field. Host values suggest that it is mostly encrypted email communication. Let us filter that out as well:

   ```
   "EXISTS %httph and EXISTS %sa6 and %port != 80
   and %port != 443 and %httprt != 11"
   ```

7. We have 6 records left. Based on user agent values, this is a BitTorrent traffic.

Task 3 - Data filtering with fbitdump

8. You might have also noticed that the HTTPS traffic does not have HTTP path defined. Thus, the filter can be simplified to:

   `"EXISTS %httpp and EXISTS %sa6 and %dp != 80 and %httpp = '_%'"`,

   where the '_%' indicated that the string must have at least one character (same as in an SQL query).

FastBit Queries

Task 4

Data aggregation with fbitdump

Task 4 - Data aggregation with fbitdump

1. Data aggregation is used to find out how many records with unique properties are present in the data set. We have already used simple aggregation over all records to get their count. All columns in an output format which have "aggregation" defined for their elements will be present in the aggregation output. The values for these columns are computed using specified aggregation function (min, max, sum, avg, count).
2. It is possible to specify "GROUP BY" columns as a parameters to the -A switch. Each row of the output has unique combination of values of the specified columns and appropriate aggregation of the aggregable columns. Let us aggregate based on HTTP request type:

```
fbitdump -R /tmp/fastbit/ -o http4 -A%httprt
```

Task 4 - Data aggregation with fbitdump

3. Any column that is not aggregable can be used for aggregation. Let us work with user agents. The command:

   ```
   fbitdump -R /tmp/fastbit/ -o http4 -A%httpa
   ```

   shows statistics for all found user agents. If we want only the most frequent, we can order the output using the -m switch:

   ```
   fbitdump -R /tmp/fastbit/ -o http4 -A%httpa -m%fl
   ```

4. The most frequent records are now at the bottom. We can use one more feature to get top 10 user agents. We will also use filter to get rid of empty user agents:

   ```
   fbitdump -R /tmp/fastbit/ -o http4 -A%httpa \
    -m"%fl DESC" -c 10 "%httpa = '_%'"
   ```

Task 4 - Data aggregation with fbitdump

3. A similar result can be achieved using a statistics switch:

```
fbitdump -R /tmp/fastbit/ -o http4 -s%httpa \
 "%httpa = '_%'"
```

4. Combinations of columns can be used to compute aggregations and statistics:

```
fbitdump -R /tmp/fastbit/ -o http4 \
 -s%httpa,%httprt "%httpa = '_%'"
```

Part IV

Network Measurements Analysis (NEMEA)

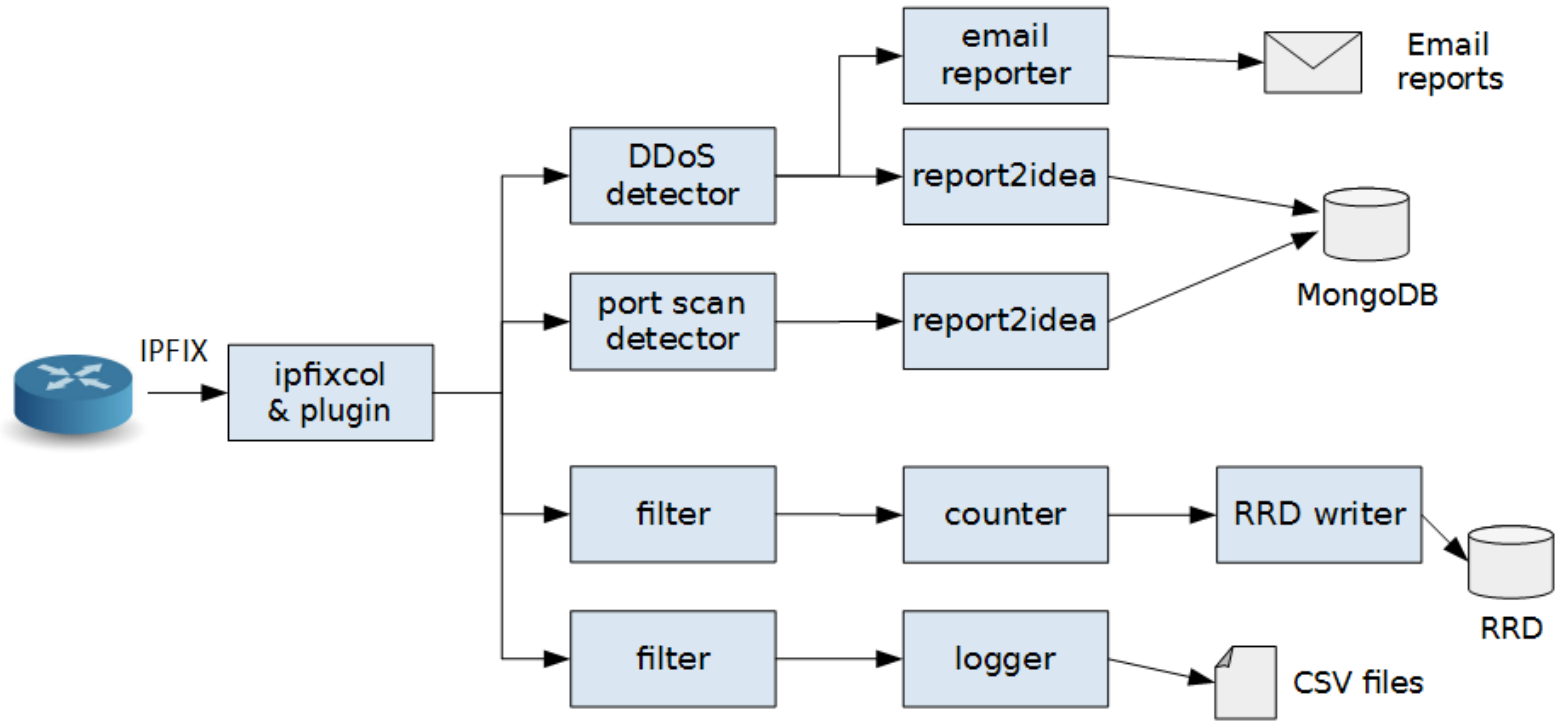
About NEMEA

NEMEA is:

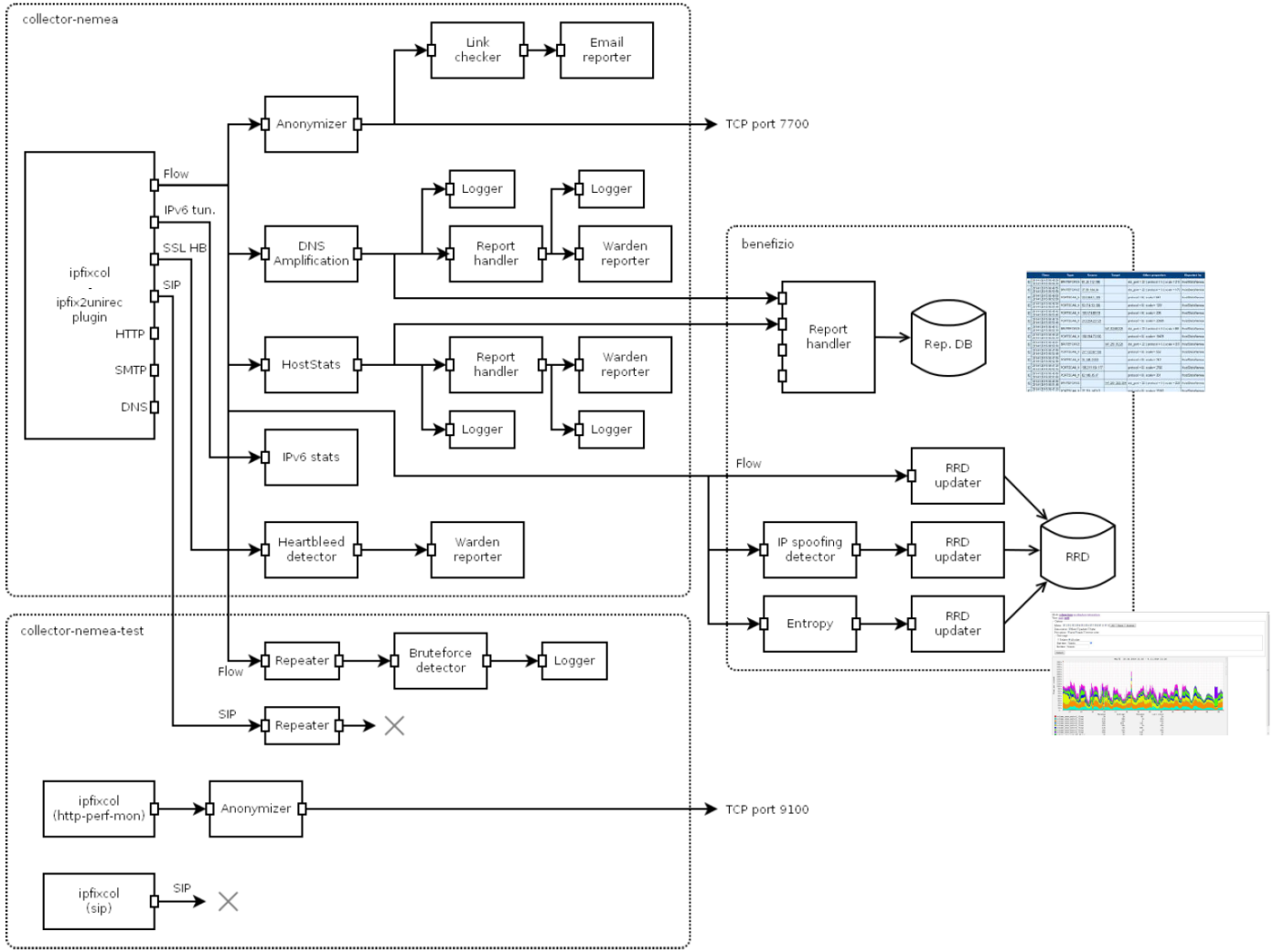
- System for stream-wise automatic processing of (not only) flow data
- Capable of L7 processing
- Independent modules → flexible, extensible, can be distributed

`https://github.com/CESNET/Nemea/`

Example NEMEA configuration

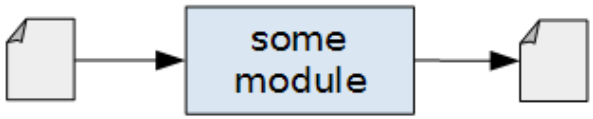


Example NEMEA configuration



Real deployment of an early version of NEMEA system at CESNET. Included just for illustration.

Example NEMEA configuration



NEMEA – key features

NEMEA can be used out-of-the box for detection of malicious traffic.

However, we see it more as a framework which every user can adjust to his/her needs.

- By enabling/disabling various modules
- By configuration of detection modules & reporting
- Even by easy implementation of new modules

Platform

NEMEA module = program using *libtrap* library

- C, C++ or Python

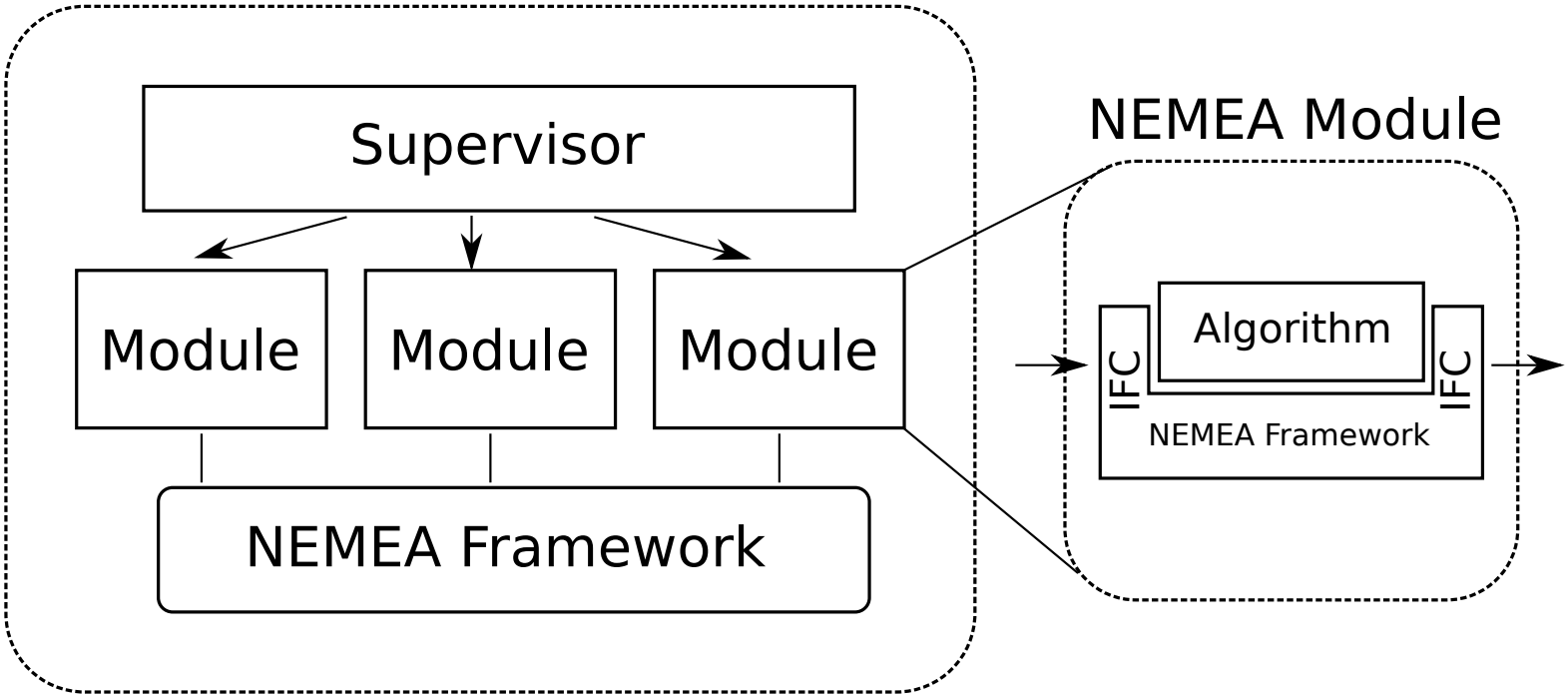
TRAP – Traffic analysis platform

- Library for high-throughput inter-process communication
- Flexible but efficient data format
- Library of common functions and data structures useful for traffic analysis
- (all designed specifically for network data analysis)

- Provides common platform for easy implementation of traffic analysis methods
  - Suitable for operational use as well as research

NEMEA Architecture

NEMEA Architecture



TRAP interfaces

Interfaces (IFC)

- Each module have 0-N input and 0-N output interfaces
- Message passing
  - Unidirectional stream of records
  - Efficient binary data format – UniRec
    (JSON and unstructured data also supported)
- Various IFC types: UNIX socket, TCP socket, File IFC
- Type and parameters of IFC are provided via `-i` argument during module startup
  - i.e. given at runtime, processed by library -> transparent to module internals

Data format – UniRec

UniRec

- Binary data format used by NEMEA
- Similar to plain C structure
- Support for variable-sized fields
- Set of keys specified at runtime, but fixed during lifetime of connection
- A record represents
  - Flow record (with L7 information)
  - Set of statistics
  - Detection result
  - ...

Example of available NEMEA modules

Data sources

- *plugin for IPFIXcol*
- *flow meter* – simple flow sensor sending data directly to other NEMEA modules
- *nfdump reader* – reads flows from nfdump files
- *logreplay* – reads records from CSV file

Output modules

- *logger* – writes records to CSV file
- *report2idea* – converts reports from various detectors to IDEA format and stores them to database or sends them to Warden
- *email reporter* – sends customizable email messages based on incoming records
- *RRD updater* – writes statistics to RRD database

Example of available NEMEA modules

Detectors

- *HostStats*
  - computes statistics about traffic of individual hosts in network
  - applies several rules to statisctics to detect misbehaving hosts
  - detects: horizontal port scans, SYN flood DoS, DNS amplif. DDoS, SSH bruteforce
- *vportscan detector* – detects vertical port scans
- *amplification detector* – detects DNS, NTP and other amplification DDoS attacks
- *brute force detector* – detects brute force / dictionary password guessing on various protocols
- ...

Example of available NEMEA modules

Others

- *anonymizer* – on-the-fly anonymization of IP addresses
- *merger* – merges several streams of data into one
- *unirec filter* – filtres records according to given rule
- ...

Section 1

NEMEA Configuration

NEMEA configuration

NEMEA supervisor

- Allows user to manage the whole NEMEA system
- Based on XML configuration file
- Architecture: **system daemon** and its ***supcli* controller**
- System daemon installed as a **systemd service**

Service control

service nemea-supervisor *

- start, stop, restart, status
- reload - updates the configuration according to the configuration file

NEMEA configuration - functions

Try out *supcli*

1. Connect via supervisor client: *supcli*
2. Show brief status of the prepared configuration: `option 4`
   - List of NEMEA modules divided into groups (profiles)
3. Enable profile *Detection - sources*:
   - Select `option 1`
   - Select number of the disabled profile *Detection - sources*
4. Get status of the configuration in detail: `option 5`
   - Module *flow_meter* should be running
5. Browse the logs with pager:
   - Select `option 9`
   - Select number of the *modules_events*
6. Get the information about current running daemon: `option 8`
7. Disconnect by pressing `Ctrl-C` or typing `Cquit`

Section 2

NEMEA Monitoring

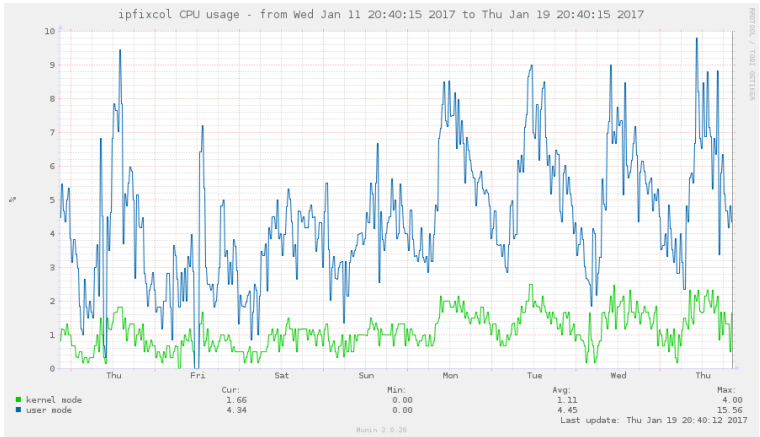
NEMEA monitoring

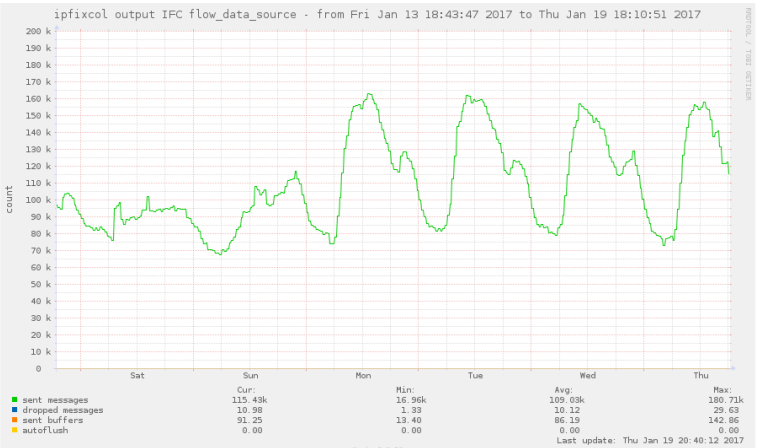
Supervisor

- Module status
- CPU and memory usage of every module
- Statistics of module interfaces - interface counters

Munin

- Contains plugin nemea-supervisor
- Periodically obtains statistics about modules from supervisor and creates graphs





Part V

Alert Reporting

What is an alert?

- Alert is a message generated by a detection module.
- Alert contains information about a detected event.
- Alerts are valuable for CSIRT/CERT people to handle a security incident.

What to do with alerts?

Storage

- Raw data file

  ```
  /usr/bin/nemea/mydetector -i f:myalerts.trapcap:w
  ```

- CSV file

  ```
  /usr/bin/nemea/logger -i u:voipalerts -a myalerts.csv
  ```

- MongoDB (used by NEMEA Dashboard – next session)

Sending alerts

- Email Notifications (afternoon session)
- Warden system for sharing alerts (afternoon session)

Part VI

Monitoring of STaaS

Monitoring the monitoring system

- The system should be working.
- We don't need to know that it is working.
- We need to know when the system is NOT working.
- We should be able to look "inside" how it works.

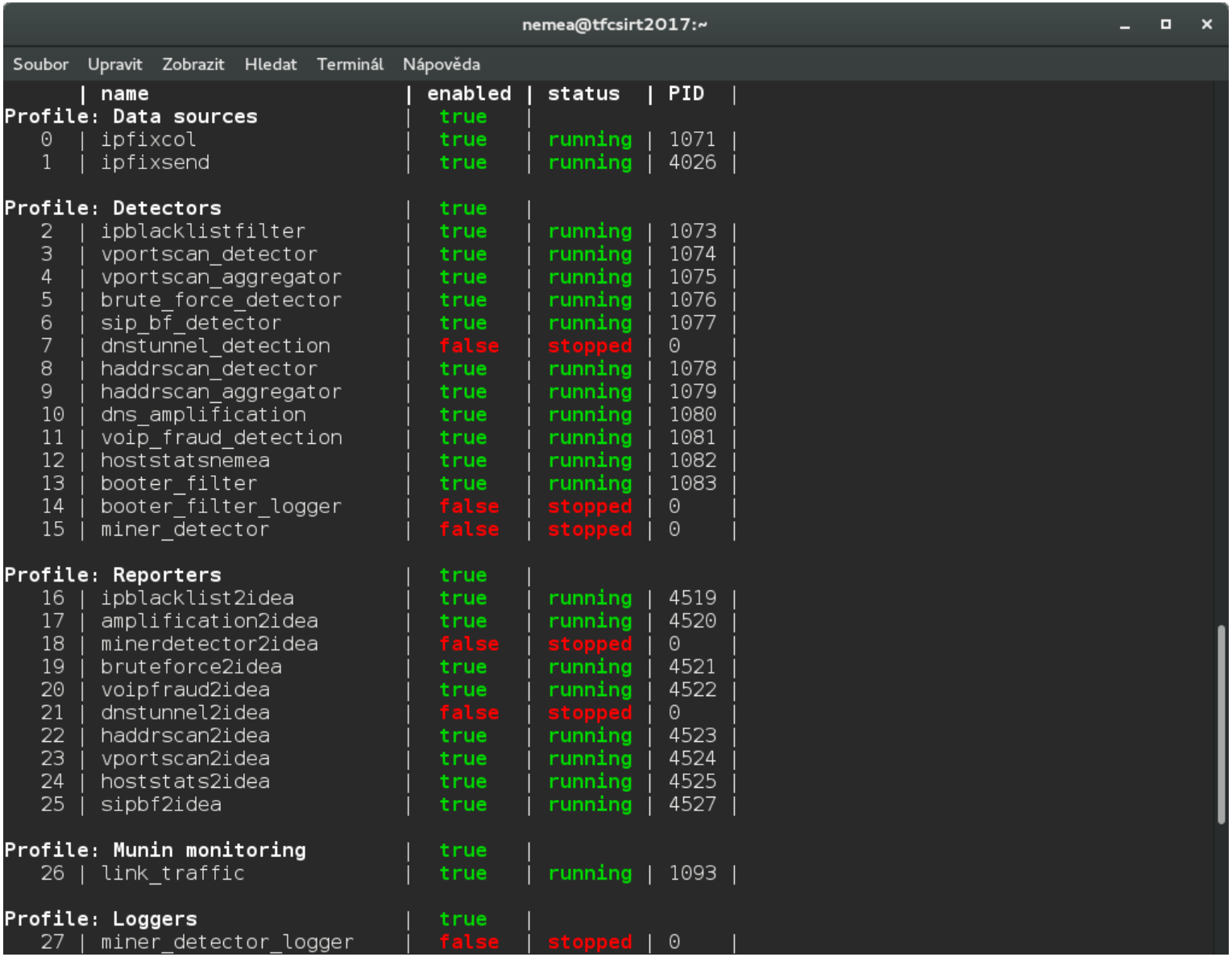
What is being monitored?

- Free memory
- CPU load
- Free disk space
- SWAP usage
- Network interface (NIC) errors
- Dropped UDP messages
- Is NEMEA Supervisor running?
- Are all modules running?
- Total number of dropped messages
- Total number of sent messages
- Number of messages sent by IPFIXcol
- Number of reported alerts
- Volume of traffic per link

What do we use?

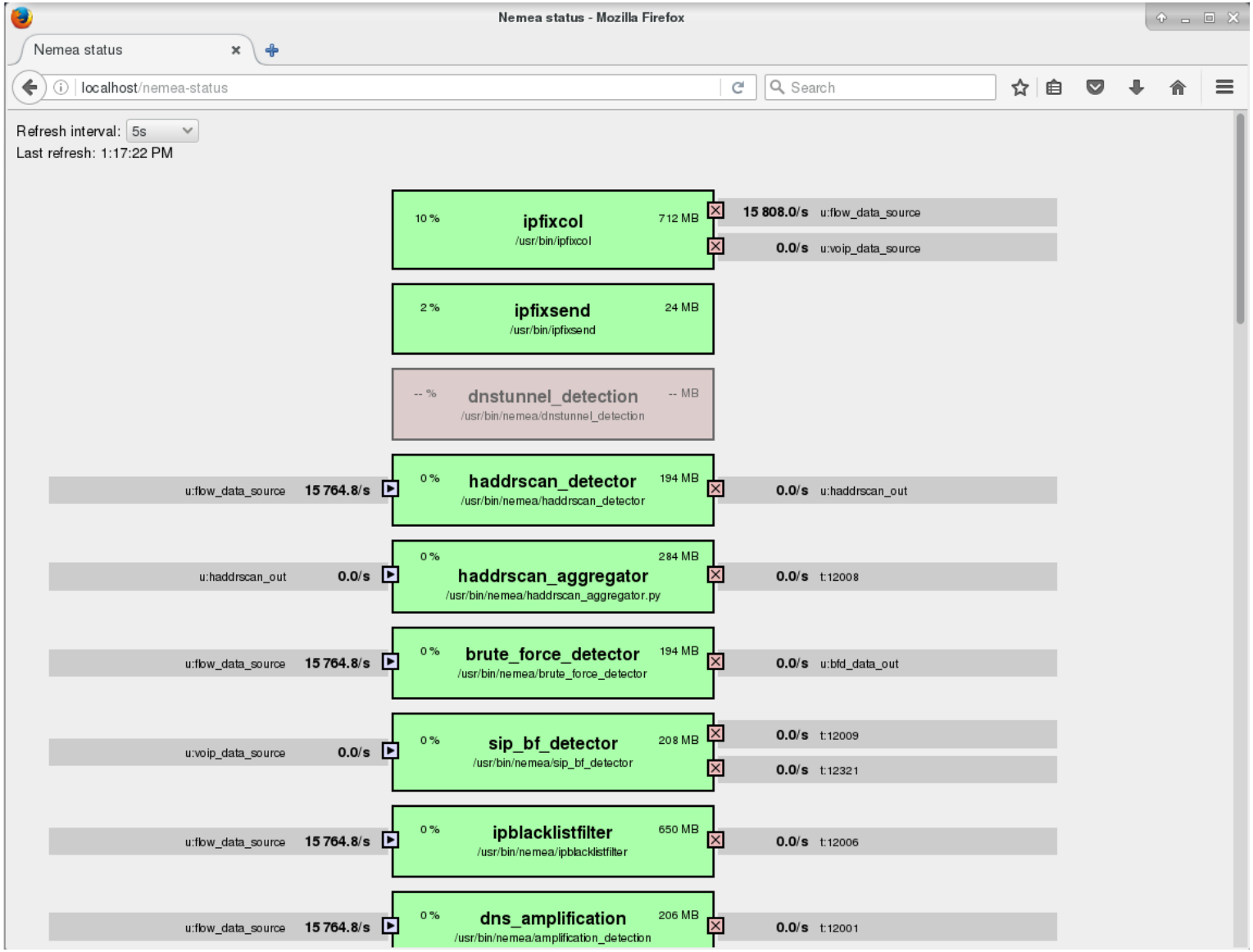
- NEMEA Supervisor
- NEMEA status (`http://localhost/nemea_status/`)
- Munin (`http://localhost/munin/`)
- NEMEA Dashboard
- zabbix / nagios

NEMEA Supervisor

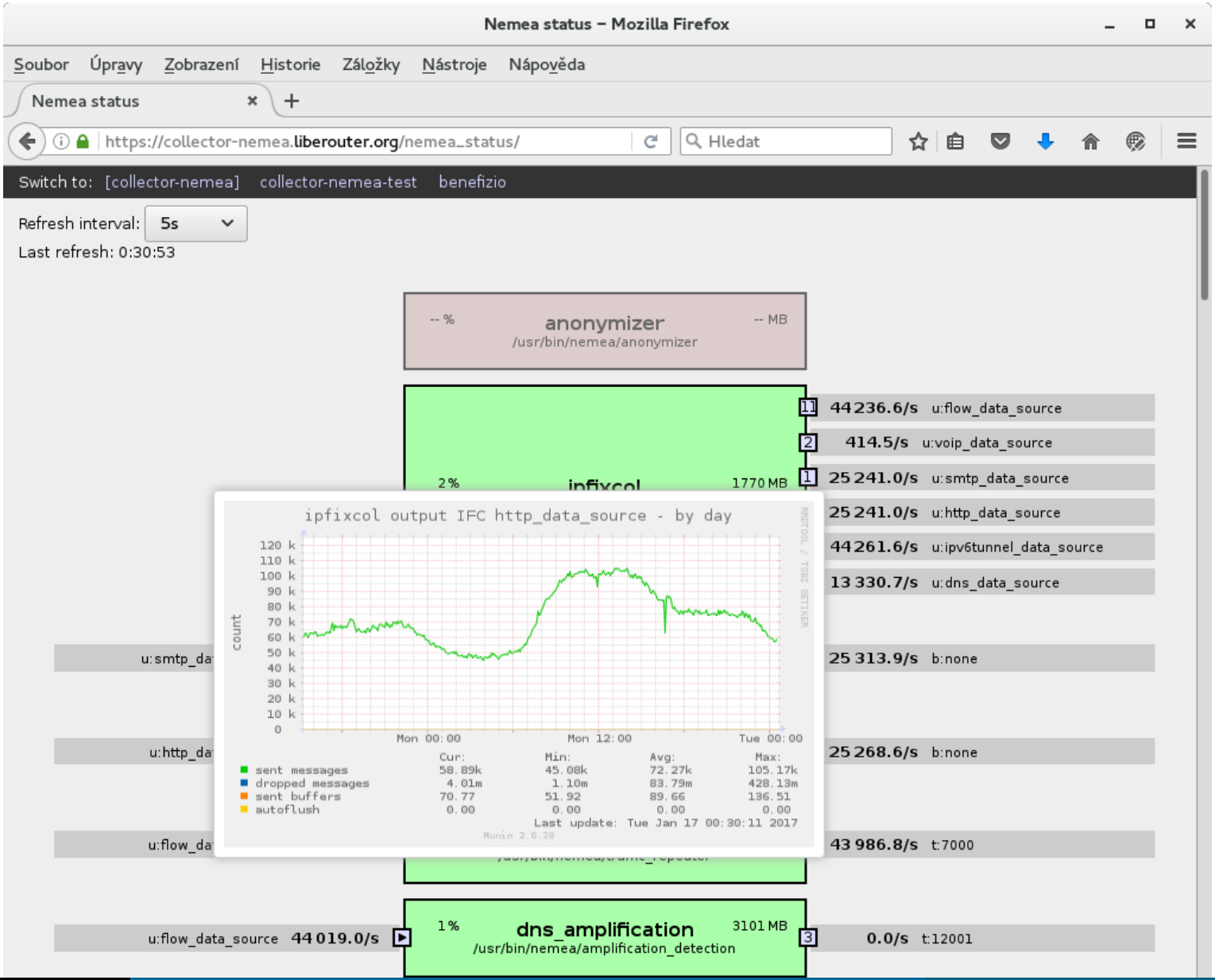


	name	enabled	status	PID
Profile: Data sources		true		
0	ipfixcol	true	running	1071
1	ipfixsend	true	running	4026
Profile: Detectors		true		
2	ipblacklistfilter	true	running	1073
3	vportscan_detector	true	running	1074
4	vportscan_aggregator	true	running	1075
5	brute_force_detector	true	running	1076
6	sip_bf_detector	true	running	1077
7	dnstunnel_detection	false	stopped	0
8	haddrscan_detector	true	running	1078
9	haddrscan_aggregator	true	running	1079
10	dns_amplification	true	running	1080
11	voip_fraud_detection	true	running	1081
12	hoststatsnemea	true	running	1082
13	booter_filter	true	running	1083
14	booter_filter_logger	false	stopped	0
15	miner_detector	false	stopped	0
Profile: Reporters		true		
16	ipblacklist2idea	true	running	4519
17	amplification2idea	true	running	4520
18	minerdetector2idea	false	stopped	0
19	bruteforce2idea	true	running	4521
20	voipfraud2idea	true	running	4522
21	dnstunnel2idea	false	stopped	0
22	haddrscan2idea	true	running	4523
23	vportscan2idea	true	running	4524
24	hoststats2idea	true	running	4525
25	sipbf2idea	true	running	4527
Profile: Munin monitoring		true		
26	link_traffic	true	running	1093
Profile: Loggers		true		
27	miner_detector_logger	false	stopped	0

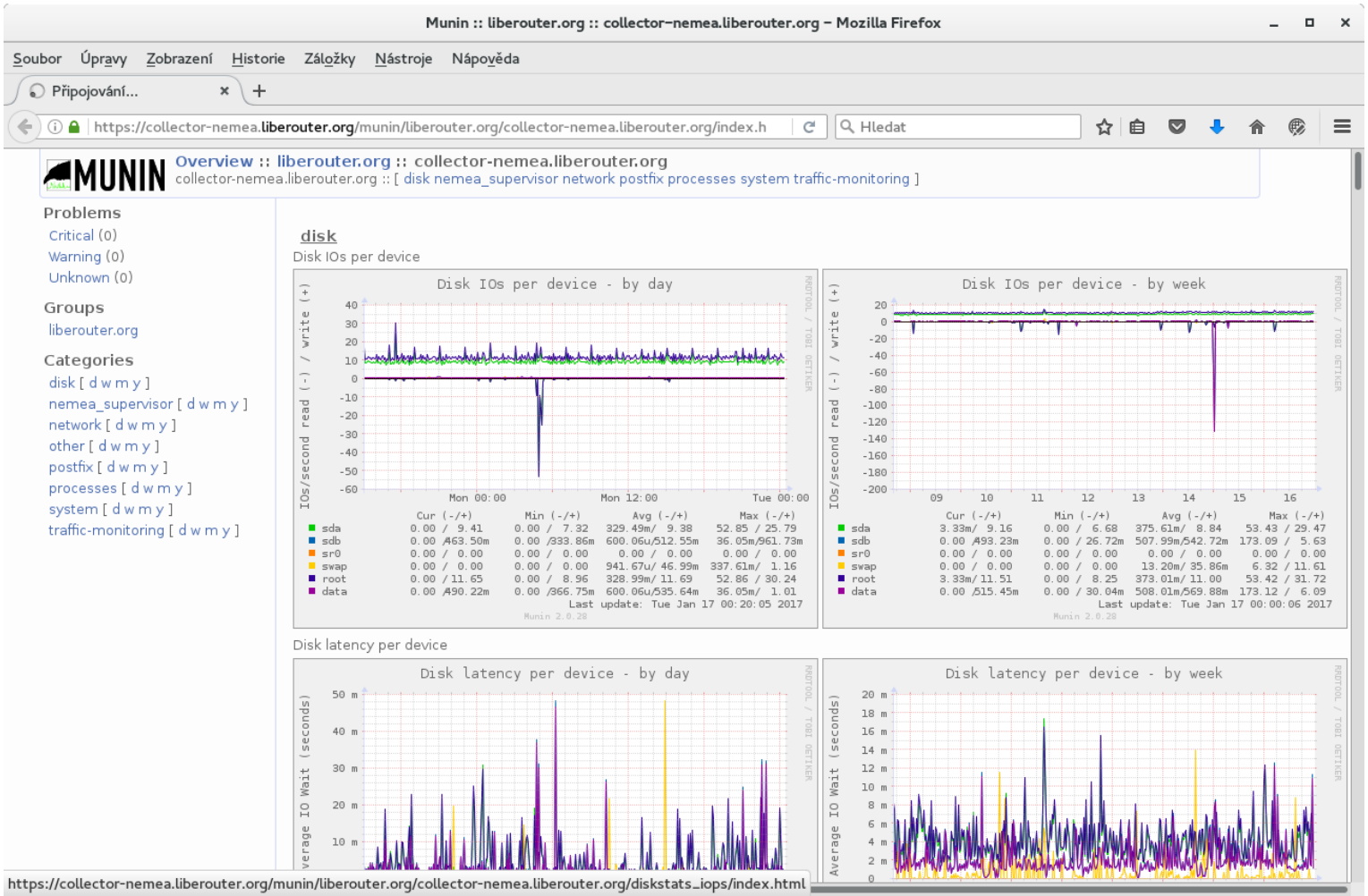
NEMEA Status



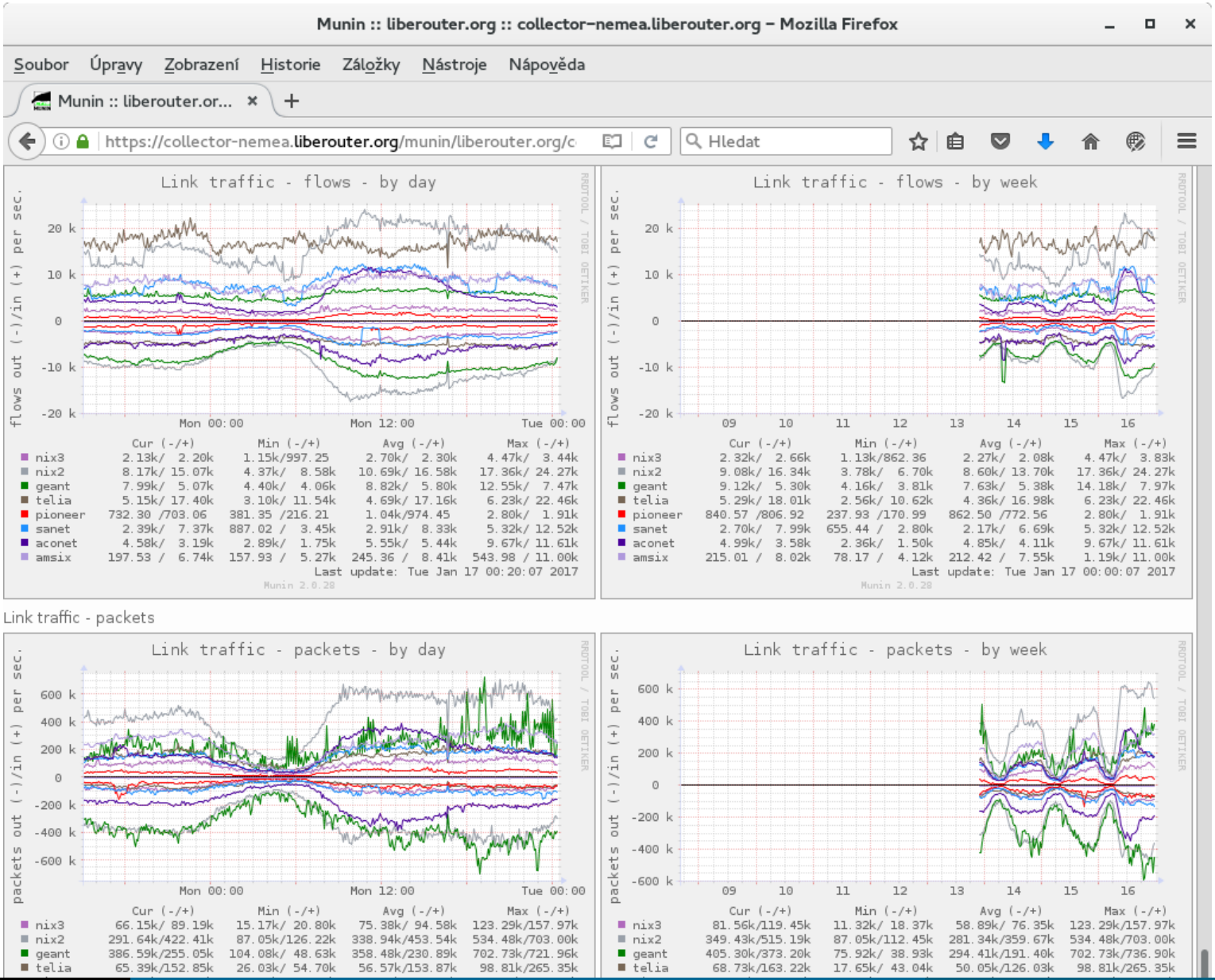
NEMEA Status



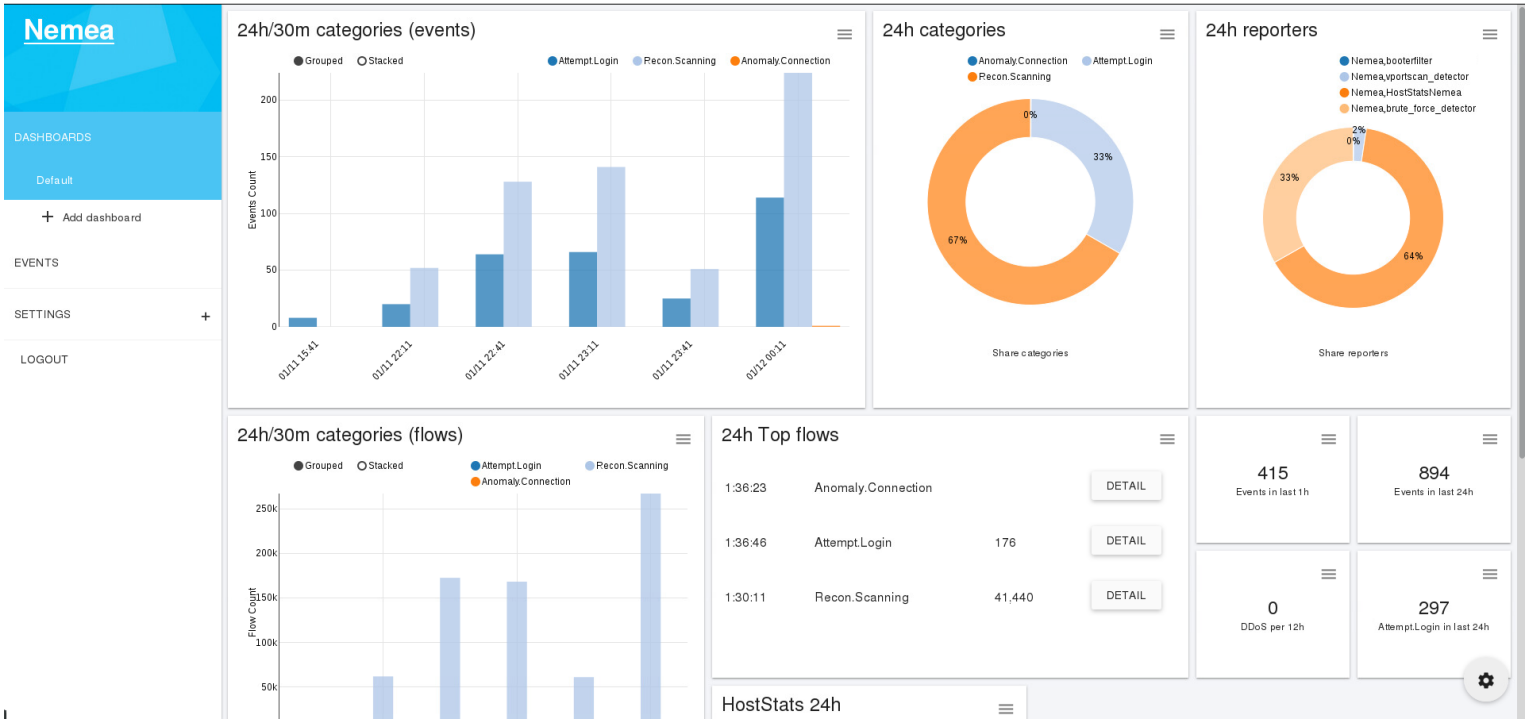
Munin


Munin :: liberouter.org :: collector-nemea.liberouter.org – Mozilla Firefox
Soubor Úpravy Zobrazení Historie Záložky Nástroje Nápověda
Připojování...
https://collector-nemea.liberouter.org/munin/liberouter.org/collector-nemea.liberouter.org/index.h
Hledat
MUNIN
Overview :: liberouter.org :: collector-nemea.liberouter.org
collector-nemea.liberouter.org :: [disk nemea_supervisor network postfix processes system traffic-monitoring]
Problems
Critical (0)
Warning (0)
Unknown (0)
Groups
liberouter.org
Categories
disk [d w m y]
nemea_supervisor [d w m y]
network [d w m y]
other [d w m y]
postfix [d w m y]
processes [d w m y]
system [d w m y]
traffic-monitoring [d w m y]
disk
Disk IOs per device
Disk IOs per device - by day
Disk IOs per device - by week
IOs/second read (-) / write (+)
Mon 00:00
Mon 12:00
Tue 00:00
09 10 11 12 13 14 15 16
Cur (-/+) Min (-/+) Avg (-/+) Max (-/+)
sda 0.00 / 9.41 0.00 / 7.32 329.49m/ 9.38 52.85 / 25.79
sdb 0.00 /463.50m 0.00 /333.86m 600.06u/512.55m 36.05m/961.73m
sr0 0.00 / 0.00 0.00 / 0.00 0.00 / 0.00 0.00 / 0.00
swap 0.00 / 0.00 0.00 / 0.00 941.67u/ 46.99m 337.61m/ 1.16
root 0.00 / 11.65 0.00 / 8.96 328.99m/ 11.69 52.86 / 30.24
data 0.00 /490.22m 0.00 /366.75m 600.06u/535.64m 36.05m/ 1.01
Last update: Tue Jan 17 00:20:05 2017
sda 3.33m/ 9.16 0.00 / 6.68 375.61m/ 8.84 53.43 / 29.47
sdb 0.00 /493.23m 0.00 / 26.72m 507.99m/542.72m 173.09 / 5.63
sr0 0.00 / 0.00 0.00 / 0.00 0.00 / 0.00 0.00 / 0.00
swap 0.00 / 0.00 0.00 / 0.00 13.20m/ 35.86m 6.32 / 11.61
root 3.33m/ 11.51 0.00 / 8.25 373.01m/ 11.00 53.42 / 31.72
data 0.00 /515.45m 0.00 / 30.04m 508.01m/569.88m 173.12 / 6.09
Last update: Tue Jan 17 00:00:06 2017
Disk latency per device
Disk latency per device - by day
Disk latency per device - by week
Average IO Wait (seconds)
https://collector-nemea.liberouter.org/munin/liberouter.org/collector-nemea.liberouter.org/diskstats_iops/index.html

Munin


Munin :: liberouter.org :: collector-nemea.liberouter.org – Mozilla Firefox
Soubor Úpravy Zobrazení Historie Záložky Nástroje Nápověda
Munin :: liberouter.or...
https://collector-nemea.liberouter.org/munin/liberouter.org/c
Hledat
Link traffic - flows - by day
flows out (-)/in (+) per sec.
20 k
10 k
0
-10 k
-20 k
Mon 00:00
Mon 12:00
Tue 00:00
Cur (-/+) Min (-/+) Avg (-/+) Max (-/+)
nix3 2.13k/ 2.20k 1.15k/997.25 2.70k/ 2.30k 4.47k/ 3.44k
nix2 8.17k/ 15.07k 4.37k/ 8.58k 10.69k/ 16.58k 17.36k/ 24.27k
geant 7.99k/ 5.07k 4.40k/ 4.06k 8.82k/ 5.80k 12.55k/ 7.47k
telia 5.15k/ 17.40k 3.10k/ 11.54k 4.69k/ 17.16k 6.23k/ 22.46k
pioneer 732.30 /703.06 381.35 /216.21 1.04k/974.45 2.80k/ 1.91k
sanet 2.39k/ 7.37k 887.02 / 3.45k 2.91k/ 8.33k 5.32k/ 12.52k
aconet 4.58k/ 3.19k 2.89k/ 1.75k 5.55k/ 5.44k 9.67k/ 11.61k
amsix 197.53 / 6.74k 157.93 / 5.27k 245.36 / 8.41k 543.98 / 11.00k
Last update: Tue Jan 17 00:20:07 2017
Munin 2.0.28
Link traffic - flows - by week
flows out (-)/in (+) per sec.
20 k
10 k
0
-10 k
-20 k
09
10
11
12
13
14
15
16
Cur (-/+) Min (-/+) Avg (-/+) Max (-/+)
nix3 2.32k/ 2.66k 1.13k/862.36 2.27k/ 2.08k 4.47k/ 3.83k
nix2 9.08k/ 16.34k 3.78k/ 6.70k 8.60k/ 13.70k 17.36k/ 24.27k
geant 9.12k/ 5.30k 4.16k/ 3.81k 7.63k/ 5.38k 14.18k/ 7.97k
telia 5.29k/ 18.01k 2.56k/ 10.62k 4.36k/ 16.98k 6.23k/ 22.46k
pioneer 840.57 /806.92 237.93 /170.99 862.50 /772.56 2.80k/ 1.91k
sanet 2.70k/ 7.99k 655.44 / 2.80k 2.17k/ 6.69k 5.32k/ 12.52k
aconet 4.99k/ 3.58k 2.36k/ 1.50k 4.85k/ 4.11k 9.67k/ 11.61k
amsix 215.01 / 8.02k 78.17 / 4.12k 212.42 / 7.55k 1.19k/ 11.00k
Last update: Tue Jan 17 00:00:07 2017
Munin 2.0.28
Link traffic - packets
Link traffic - packets - by day
packets out (-)/in (+) per sec.
600 k
400 k
200 k
0
-200 k
-400 k
-600 k
Mon 00:00
Mon 12:00
Tue 00:00
Cur (-/+) Min (-/+) Avg (-/+) Max (-/+)
nix3 66.15k/ 89.19k 15.17k/ 20.80k 75.38k/ 94.58k 123.29k/157.97k
nix2 291.64k/422.41k 87.05k/126.22k 338.94k/453.54k 534.48k/703.00k
geant 386.59k/255.05k 104.08k/ 48.63k 358.48k/230.89k 702.73k/721.96k
telia 65.39k/152.85k 26.03k/ 54.70k 56.57k/153.87k 98.81k/265.35k
Link traffic - packets - by week
packets out (-)/in (+) per sec.
600 k
400 k
200 k
0
-200 k
-400 k
-600 k
09
10
11
12
13
14
15
16
Cur (-/+) Min (-/+) Avg (-/+) Max (-/+)
nix3 81.56k/119.45k 11.32k/ 18.37k 58.89k/ 76.35k 123.29k/157.97k
nix2 349.43k/515.19k 87.05k/112.45k 281.34k/359.67k 534.48k/703.00k
geant 405.30k/373.20k 75.92k/ 38.93k 294.41k/191.40k 702.73k/736.90k
telia 68.73k/163.22k 17.65k/ 43.04k 50.05k/126.03k 98.81k/265.35k

NEMEA Dashboard



Part VII

Data Visualisation – NEMEA Dashboard

NEMEA Dashboard

A practical tour through NEMEA Dashboard

First look

- Open NEMEA Dashboard: icon on desktop or `http://localhost/Nemea-Dashboard`
- Login: nemea
- Password: nemea
- First view – configurable dashboard

Features

- Multiple configurable dashboards (auto-refresh, timeshift)
- Configurable charts
- Searching for alerts: database query / filtering fetched results
- List of reported events (alerts)
- Drill-down analysis of data
- And many more...

Part VIII

Detection

Section 1

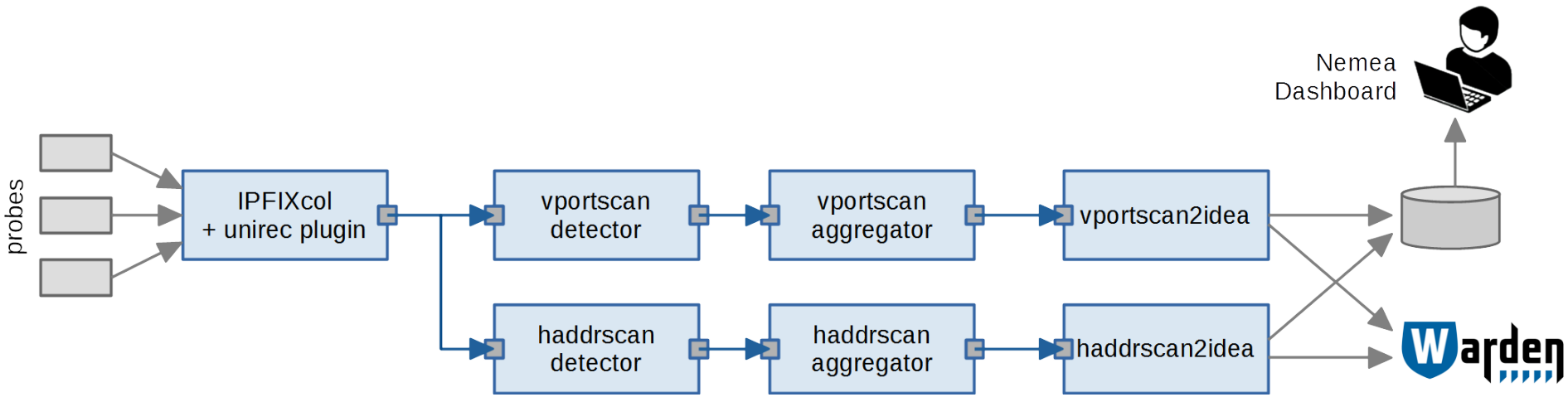
Network scanning

Network scanning

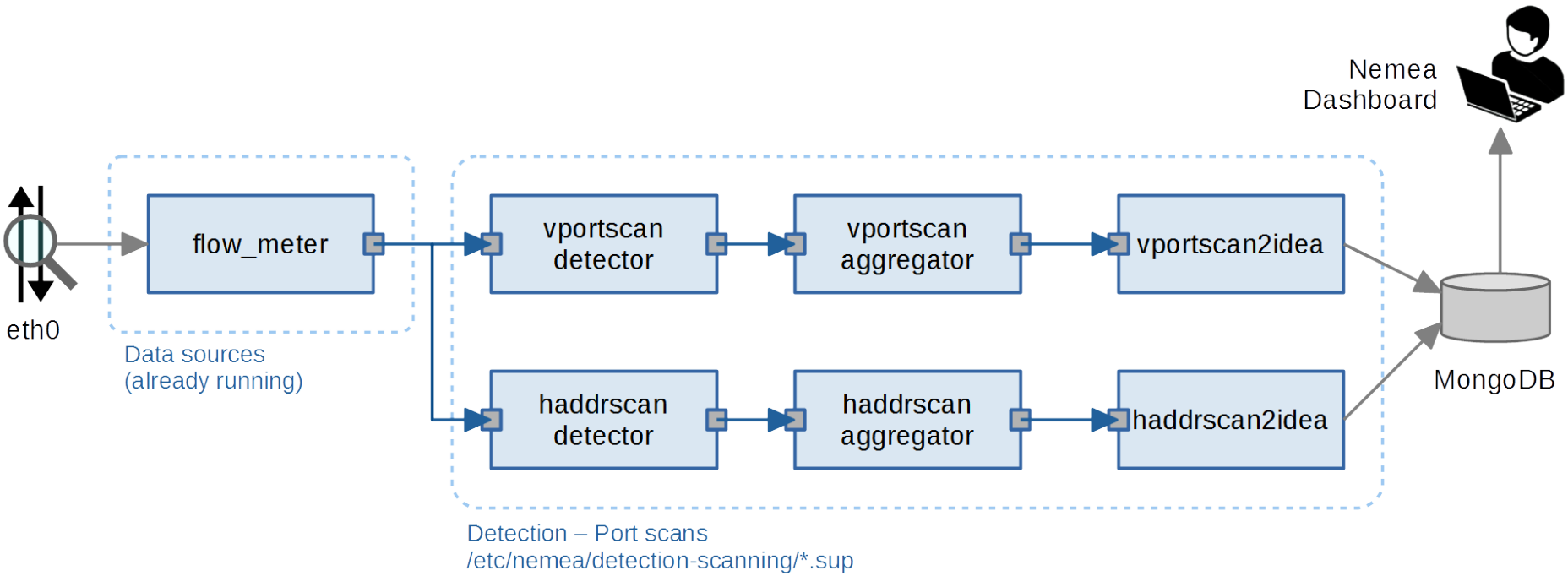
Network scanning

- Harmless and frequent activity
- Sometimes followed by a real attack
- Types of scanning
  - Horizontal – probes more targets (IPs)
  - Vertical – probes more ports of one target
  - Block – combination of both
- Simple NEMEA detectors for TCP SYN scans

Network scanning detection – common real usage



Network scanning detection – our scenario



Network scanning

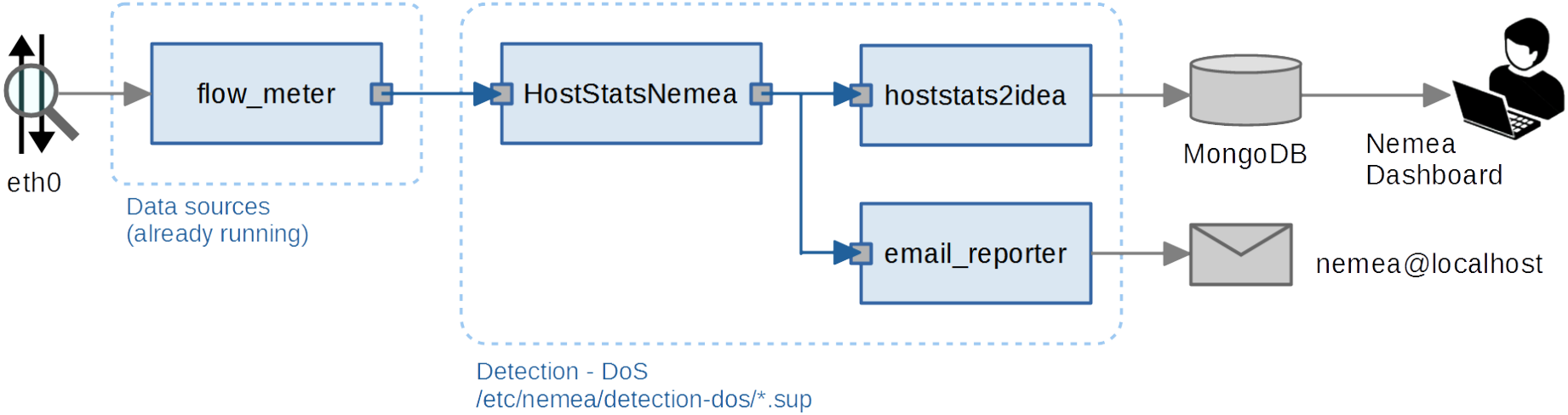
Try out performing and detecting a scan:

- Using *supcli* enable profile:
  `Detection - Scanning`
- Run in terminal:
  1. `sudo nmap -sS -P0 10.3.50.99`
  2. `sudo nmap -sS -P0 -T5 10.128.0.0/16 -p 80,443`
- Wait and check Dashboard.
- Using *supcli* disable profile:
  `Detection - Scanning`

Section 2

Denial of Service

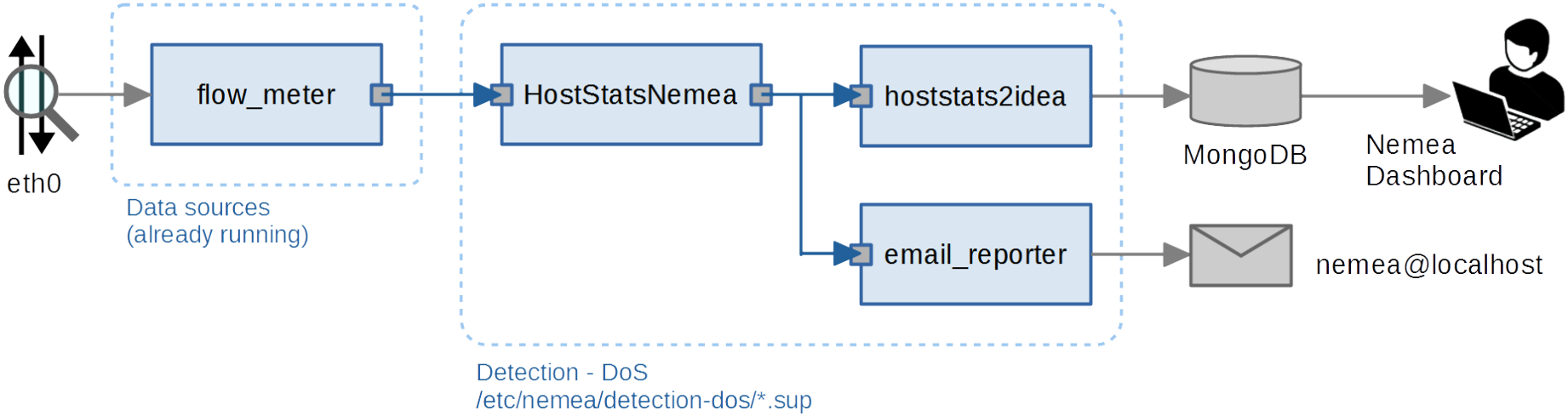
DoS — SYN Flood Detected by hoststatsnemea



- HostStatsNemea
  - Traffic statistics of individual IP addresses
  - Rules for detection of port scans, TCP SYN flood, generic flood, DNS reflection DDoS, SSH bruteforce
  - Default configuration was modified for demo:

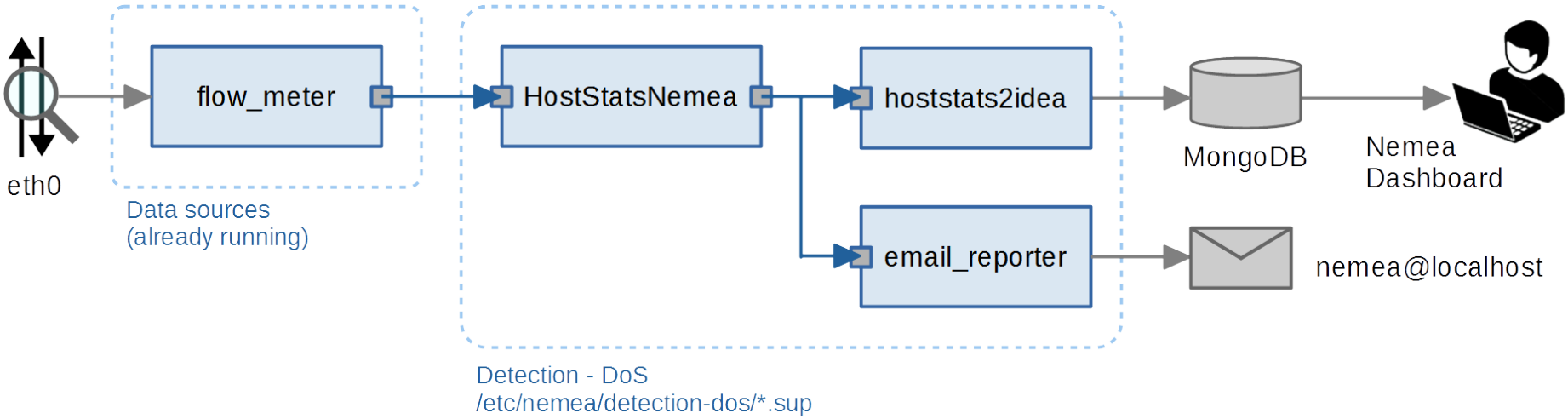
```
[/etc/nemea/hoststats.conf]
dos-victim-connections-synflood = 1000 #default: 270000
dos-attacker-connections-synflood = 1000  #default: 270000
```

DoS — SYN Flood Detected by hoststatsnemea



- We also want to send alerts via email:
  - email_reporter module
  - See the prepared configuration `/etc/nemea/email_reporter/email-reporter.cfg`

DoS — SYN Flood Detected by hoststatsnemea



- Using *supcli* enable profile:

  `Detection - DoS`

- Run:

  ```
  sudo hping3 10.123.1.2 -p 80 -S -i u1000 -q
  ```

- Stop it by `ctrl-C` after a few seconds.
- Wait and check Dashboard.
- Using *supcli* disable profile:

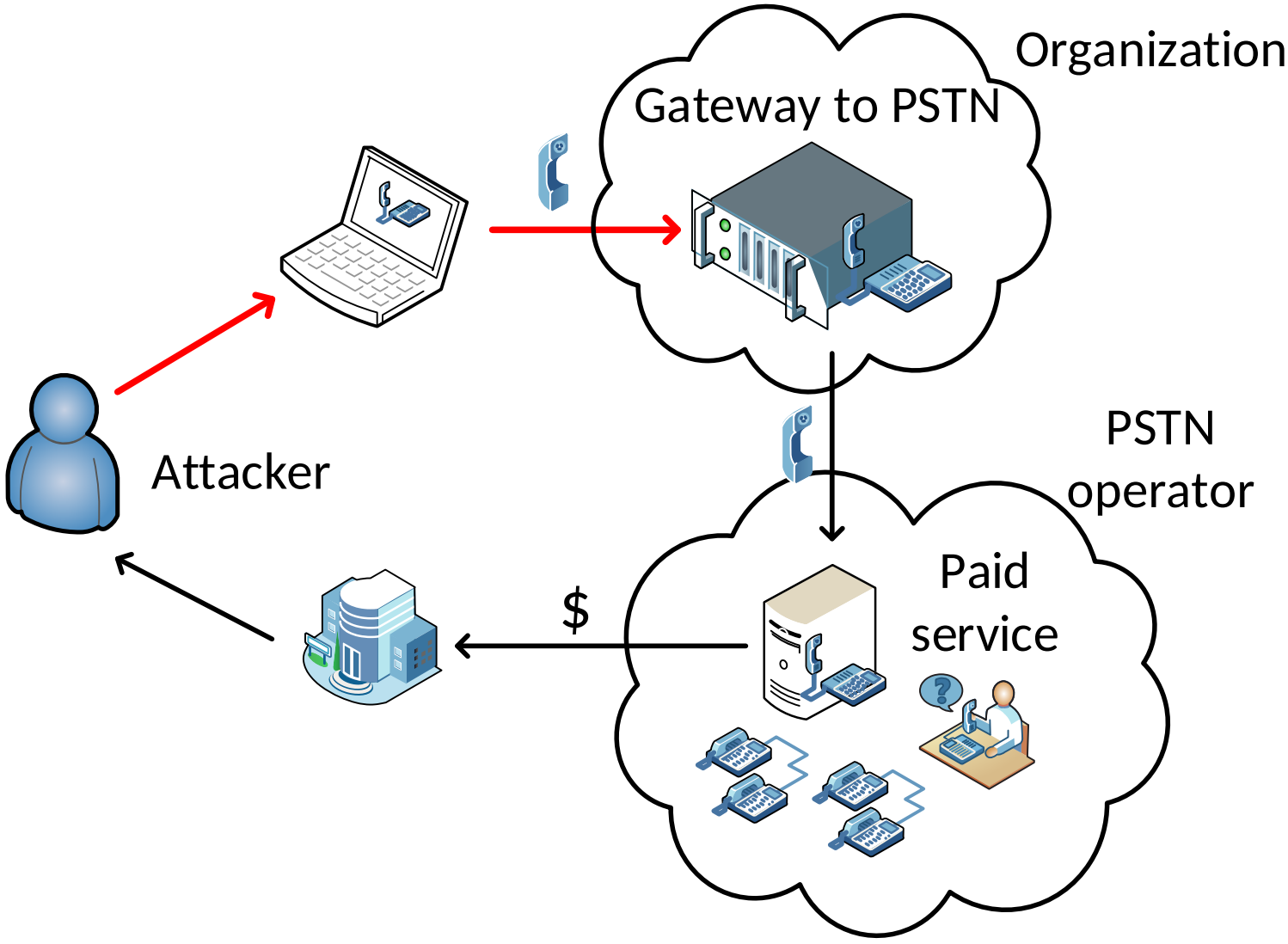
  `Detection - DoS`

Section 3

VoIP – SIP Authentication Attacks and Toll Fraud

Toll Fraud

- An attempt to perform unauthorized long-distance calls or calls to **premium numbers**
- Target of the attack: **Private Branch Exchange** (PBX)
  - A telephone system within an organization that switches calls between users inside the organization and external phone lines
- Attacker's motives:
  - Financial gain
  - Cause the organization a financial loss
- Core of the attack execution: **dial-plan guessing**


Organization
Gateway to PSTN
Attacker
PSTN
operator
Paid
service
$

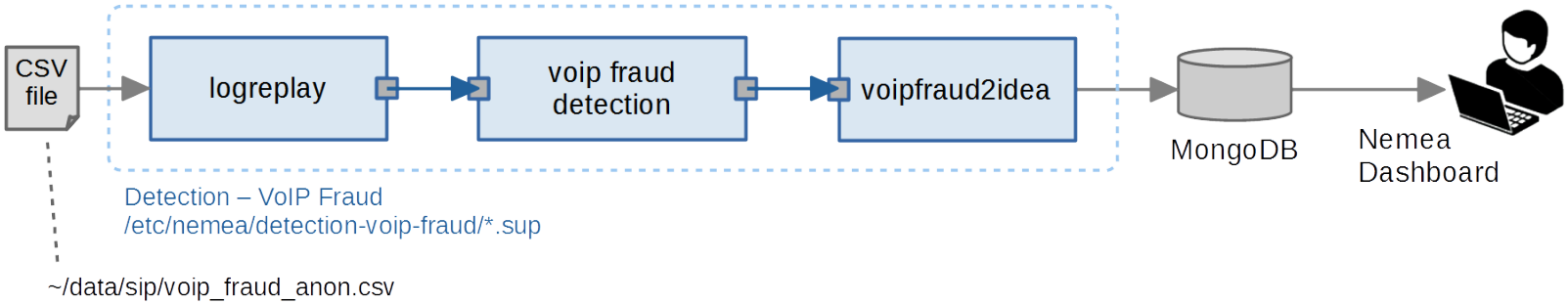
Observed prefixes

00972592577956@...
000972592577956@...
900972592577956@...
972592577956@...
100972592577956@...
800972592577956@...
600972592577956@...
700972592577956@...
400972592577956@...
300972592577956@...
200972592577956@...
500972592577956@...
99900972592577956@...
999900972592577956@...
9999900972592577956@...
99999900972592577956@...
999999900972592577956@...
9999999900972592577956@...
99999999900972592577956@...
9000972592577956@...
0972592577956@...
0000972592577956@...
0000000972592577956@...
00000000972592577956@...
000000000972592577956@...
0000000000972592577956@...
91000972592577956@...
9900972592577956@...
9100972592577956@...

Detection using NEMEA

Data: `voip_fraud_anon.csv`

- Using *supcli* enable profile:
  `Detection - VoIP Fraud`
- Wait and check Dashboard
- Using *supcli* disable profile:
  `Detection - VoIP Fraud`



SIP Authentication Attacks

- An attempt to discover a valid SIP extension (username) on a server and retrieve the **password** associated with the extension
- Often preceeds the toll fraud if the PBX is secured (every call must be authorized)
- Target of the attack: **Private Branch Exchange** (PBX)
- Attacker's motives:
  - Identity theft
  - User's credentials play key role in many other frauds
- Core of the attack execution: **extension scanning, password guessing**

Detection using NEMEA

Data: `sip_bf_anon.csv`

- Using *supcli* enable profile:
  `Detection - SIP BF`
- Wait and check Dashboard
- Using *supcli* disable profile:
  `Detection - SIP BF`

Section 4

Filtering in NEMEA (unirecfilter)

Unirecfilter

Unirecfilter

- NEMEA module for filtering records
- One or more outputs (one output for each filter rule)

Filter

- Filter rule – logical expression with UniRec fields and their values
- simple filter rule can be specified on command line, e.g.:

```
-F "DST_IP == 50.194.29.188 && DST_PORT == 25"
```

- multiple filters can be loaded from file

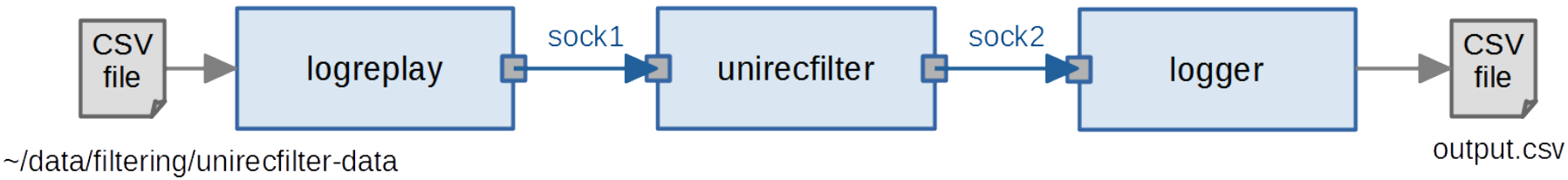
```
-f filter.txt
```

Unirecfilter

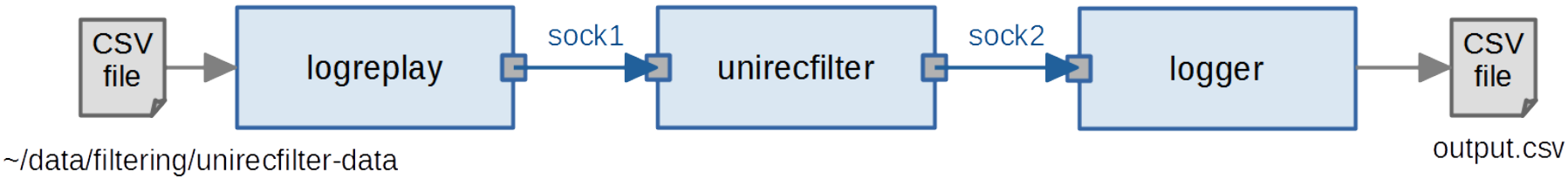
Can be used for

- Pre-filtering flow data for other module(s) **or** splitting data to multiple streams
  - traffic on specific port
  - traffic of specific organization/department
  - ...
- Ad-hoc search for specific traffic
  - HTTP requests to a particular domain (CC server)
  - Shellshock in HTTP requests (`USER_AGENT =~ "^() { "`)
  - ...

Filtering flow records



Filtering flow records



Logreplay

- read records from CSV file (or stdin)
- send data in UniRec format to the output interface

Unirecfilter

- receive data in UniRec format from the input interface
- send filtered records and fields to the output interface

Logger

- receive data in UniRec format from the input interface
- write records to the CSV file (or stdout)

Try it yourself - workflow

Run these modules simultaneously (we need 3 terminals):

- Terminal A: Read the records by logreplay

```
cd ~/data/filtering
/usr/bin/nemea/logreplay -i "u:sock1" \
-f unirecfilter-data
```

- Terminal B: Receive the records by logger

```
/usr/bin/nemea/logger -i "u:sock2"
```

- Terminal C: Do some filtering, see the next slide before pressing Enter

```
/usr/bin/nemea/unirecfilter -i \
"u:sock1,u:sock2" -F [FILTER]
```

Try it yourself - filtering

Replace `[FILTER]` with any of the following:

1. Flows from subnet 93.113.168.0/24

```
"SRC_IP >= 93.113.168.0 && SRC_IP <= 93.113.168.255"
```

2. IP addresses using NTP (port 123)

```
"SRC_PORT == 123 || DST_PORT == 123" \
-O "ipaddr SRC_IP,ipaddr DST_IP"
```

Section 5

L7 Filtering

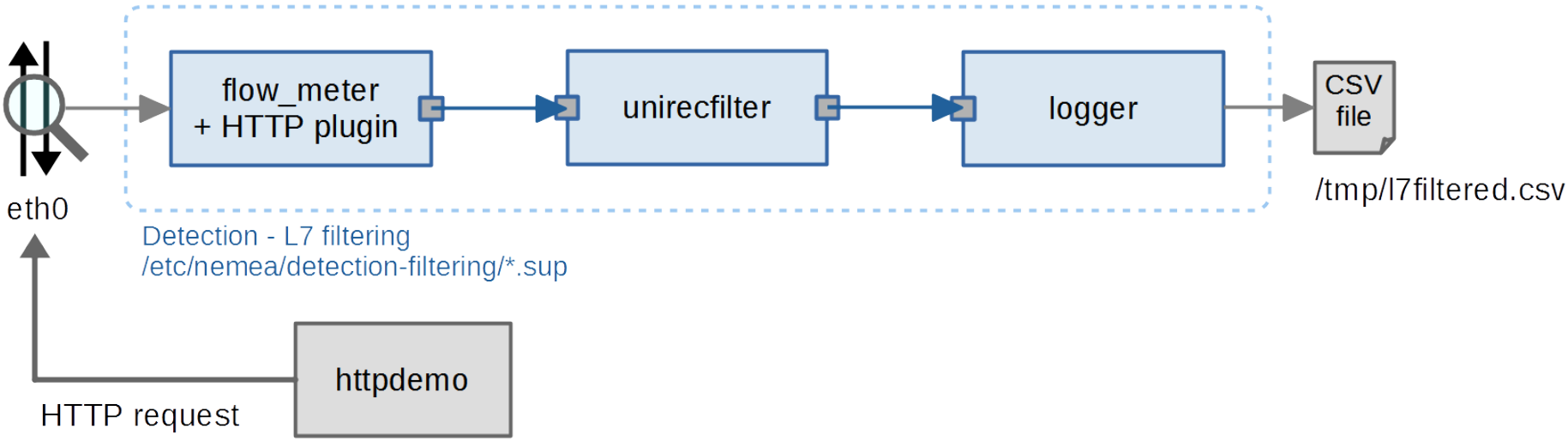
Detect Communication with CC

- `cd ~nemea/httpdemo && make`
- `strings httpdemo | less`
- Using *supcli* disable profile: `Detection - sources`
- Using *supcli* enable profile: `Detection - L7 filtering`
- Filter that was given to `unirecfilter`:

```
'HTTP_USER_AGENT =~ "demo_bot-.*" ||
HTTP_URL =~ "/demo/bot$" ||
HTTP_HOST =~ "evilcorp.com$"'
```

- Run `httpdemo` to generate HTTP requests
- `tail -f /tmp/l7filtered.csv`

L7 filtering



Section 6

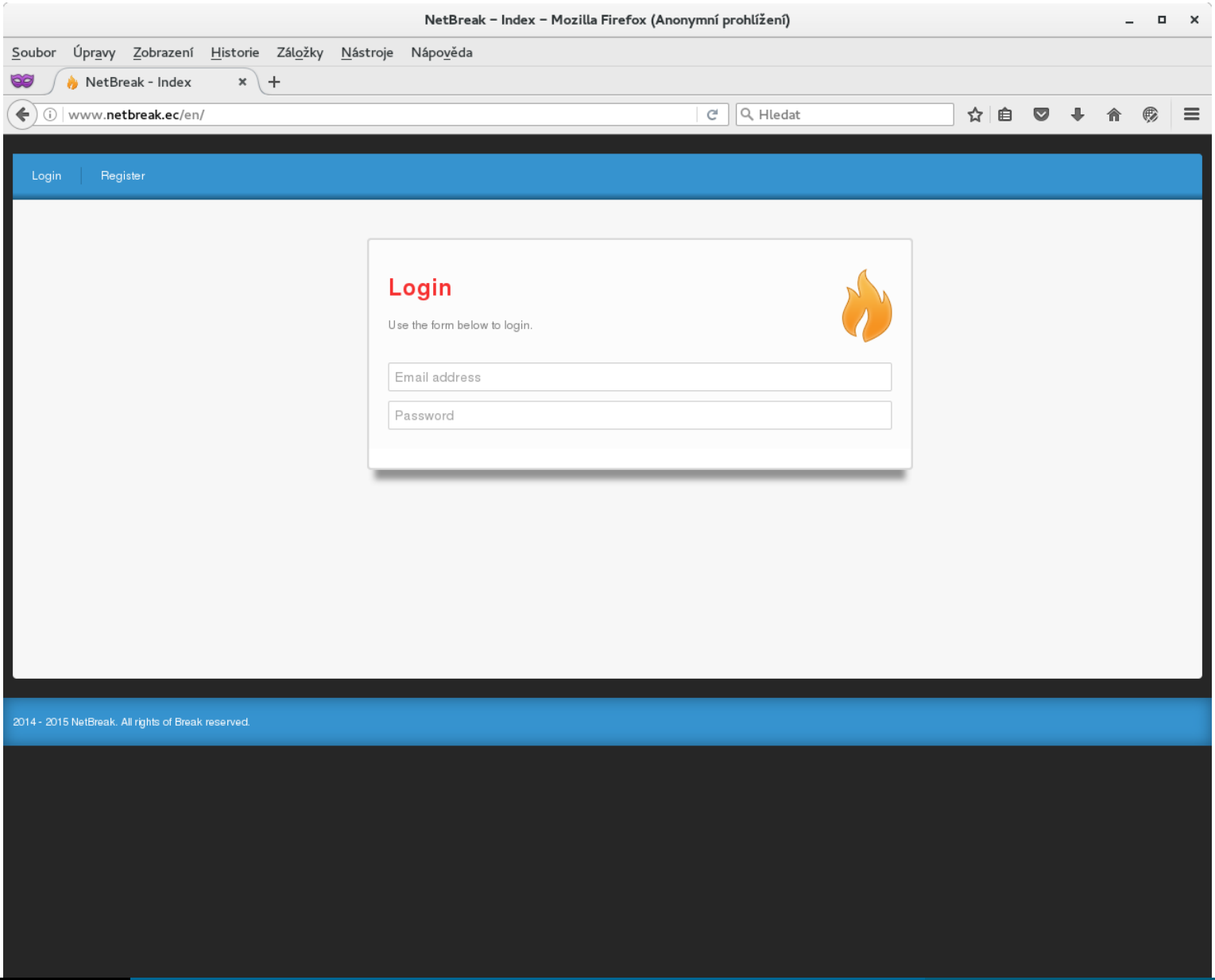
Booters

What is a booter

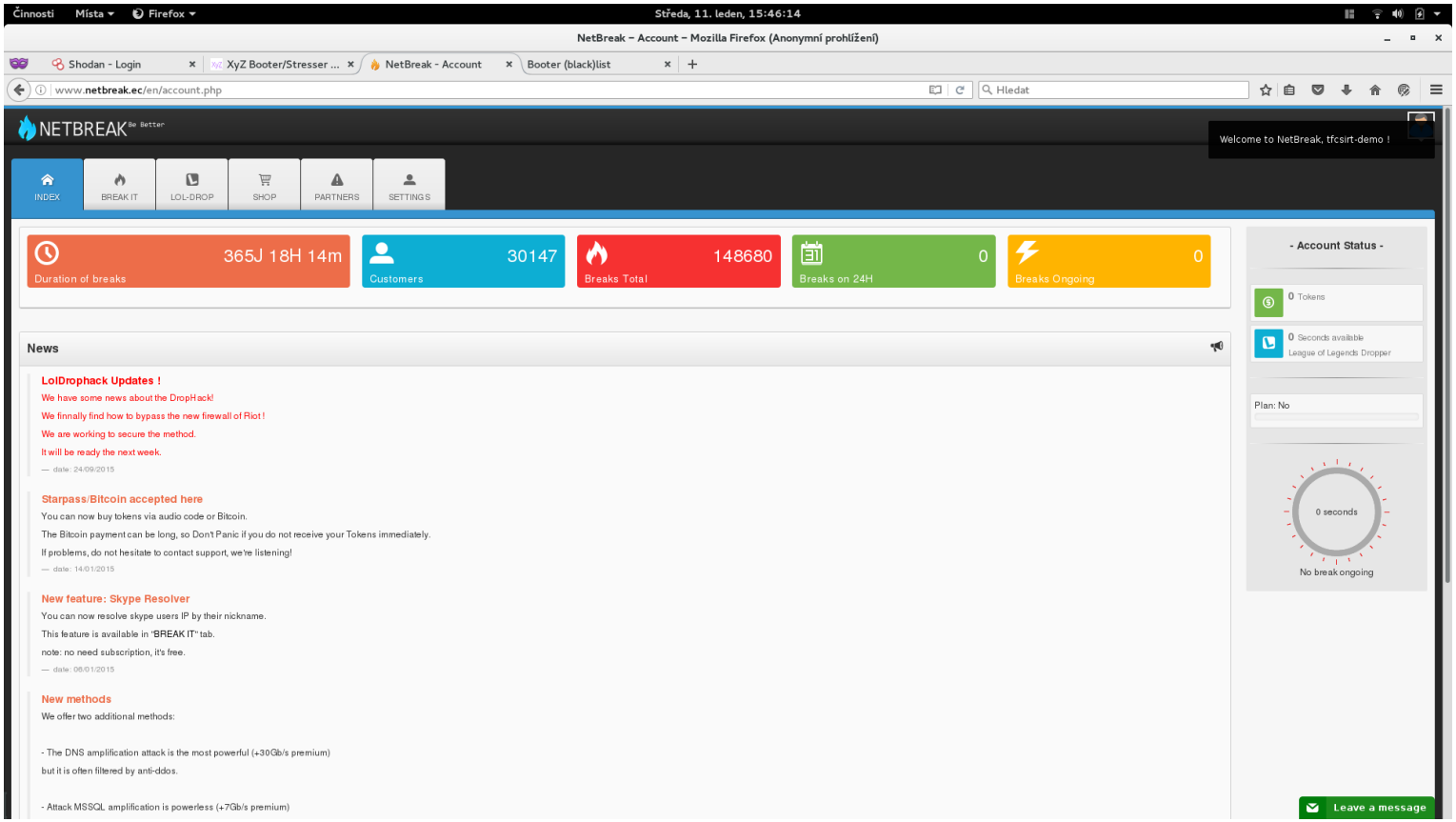
- A publicly available service
- “Stress” test generator
- Cheap service for hire
- Service can be found by any Internet user using Google/Youtube/...

Dangerous tool usable by everyone!

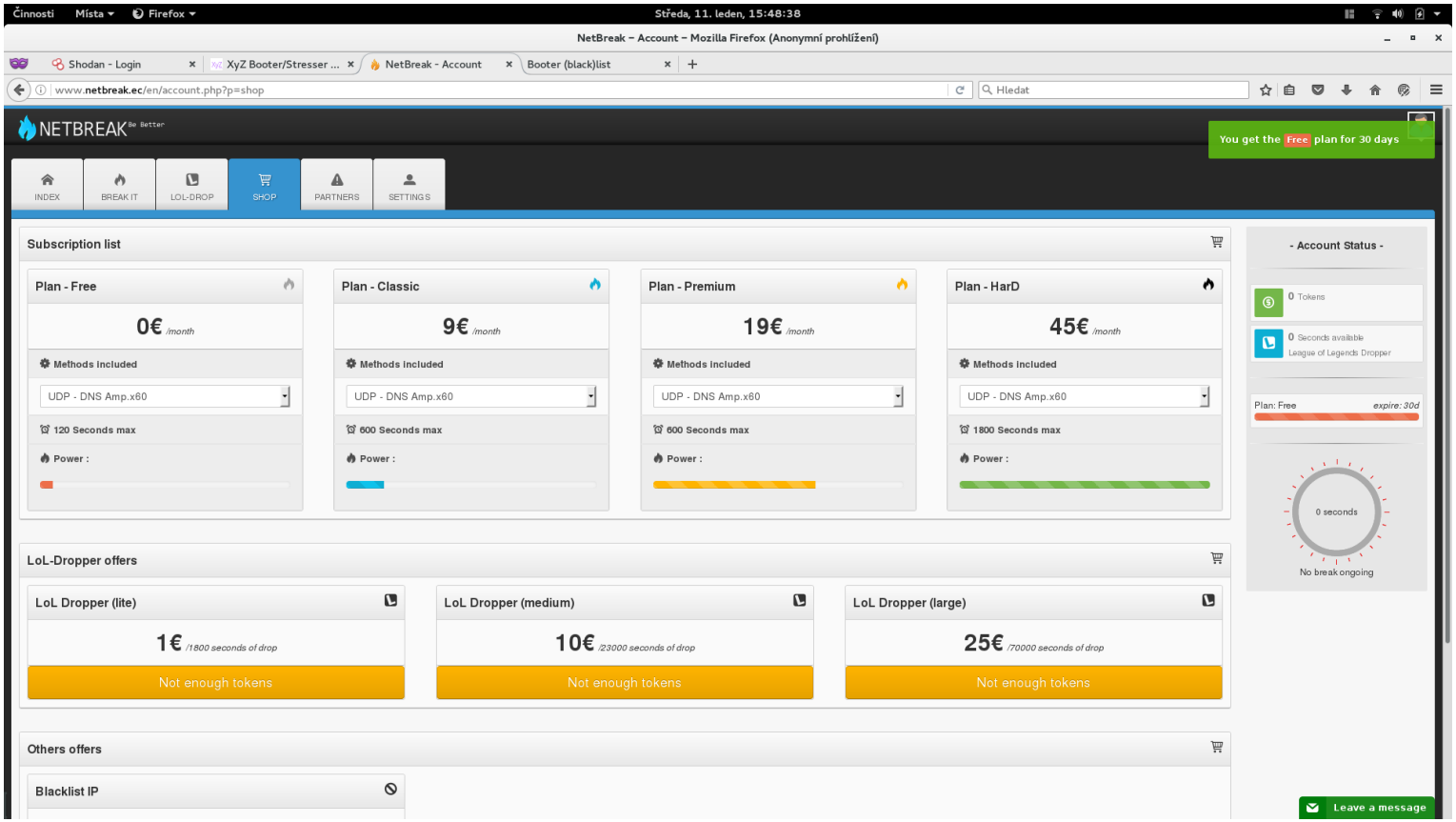
How does it look like?



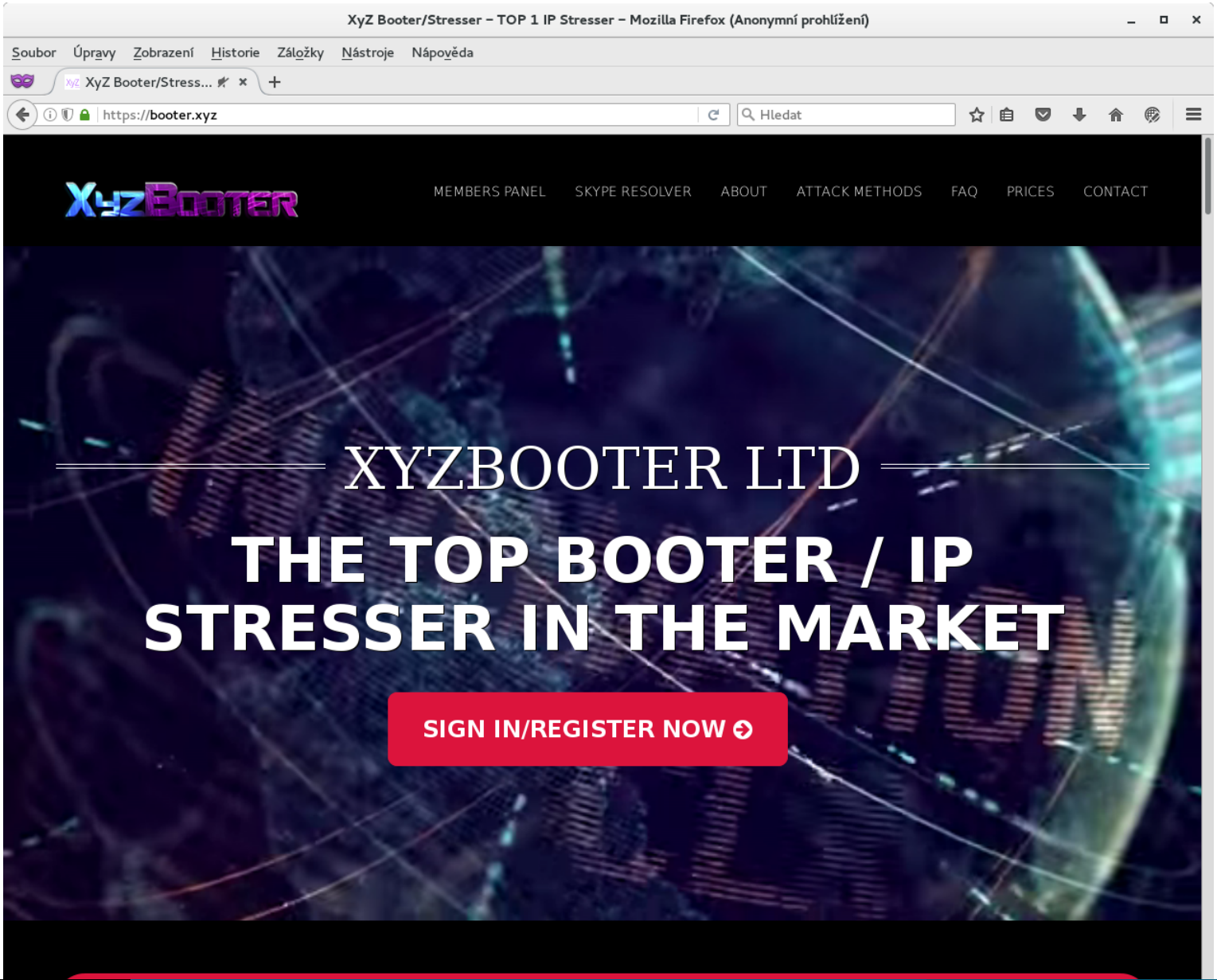
How does it look like?



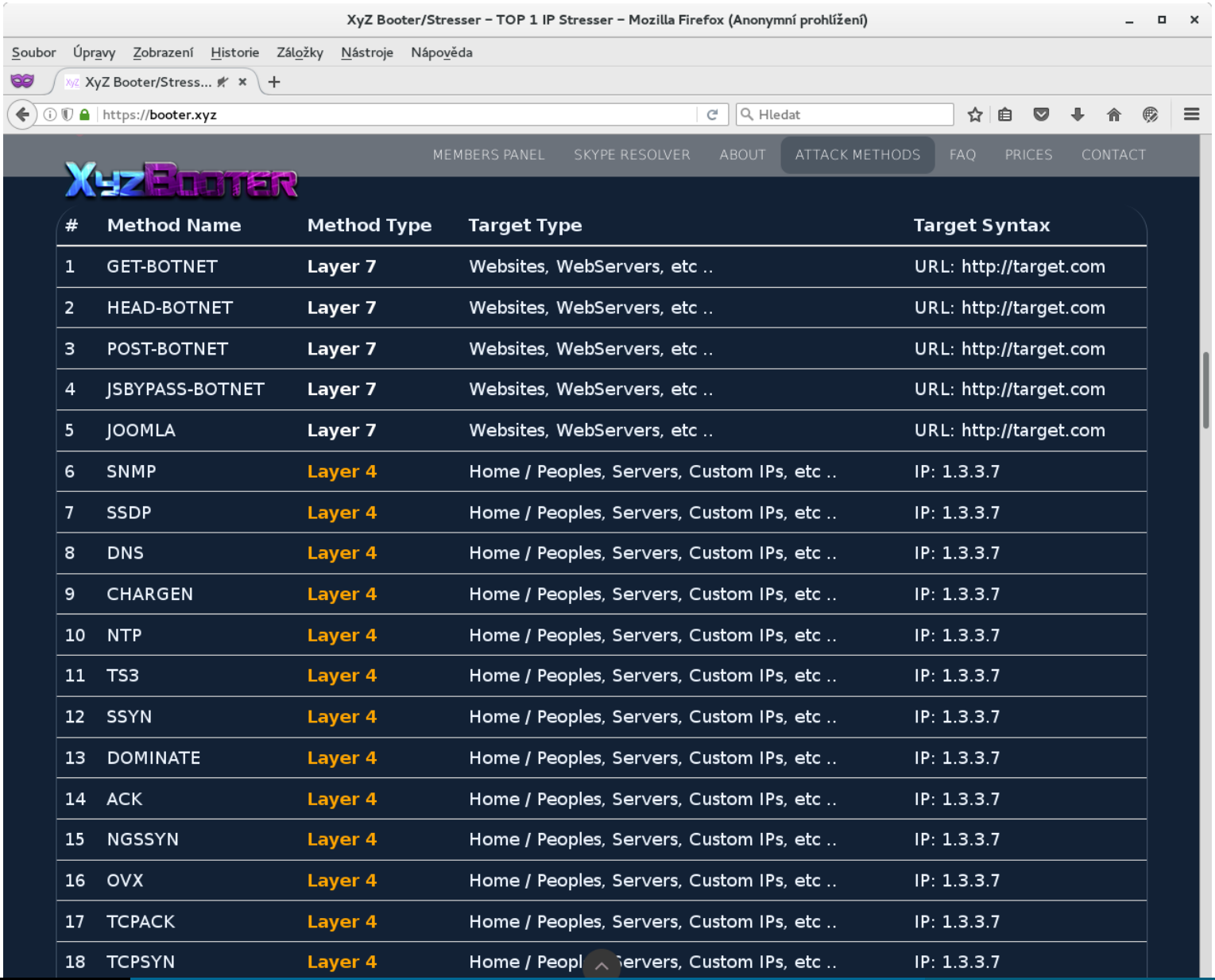
How does it look like?



How does it look like?

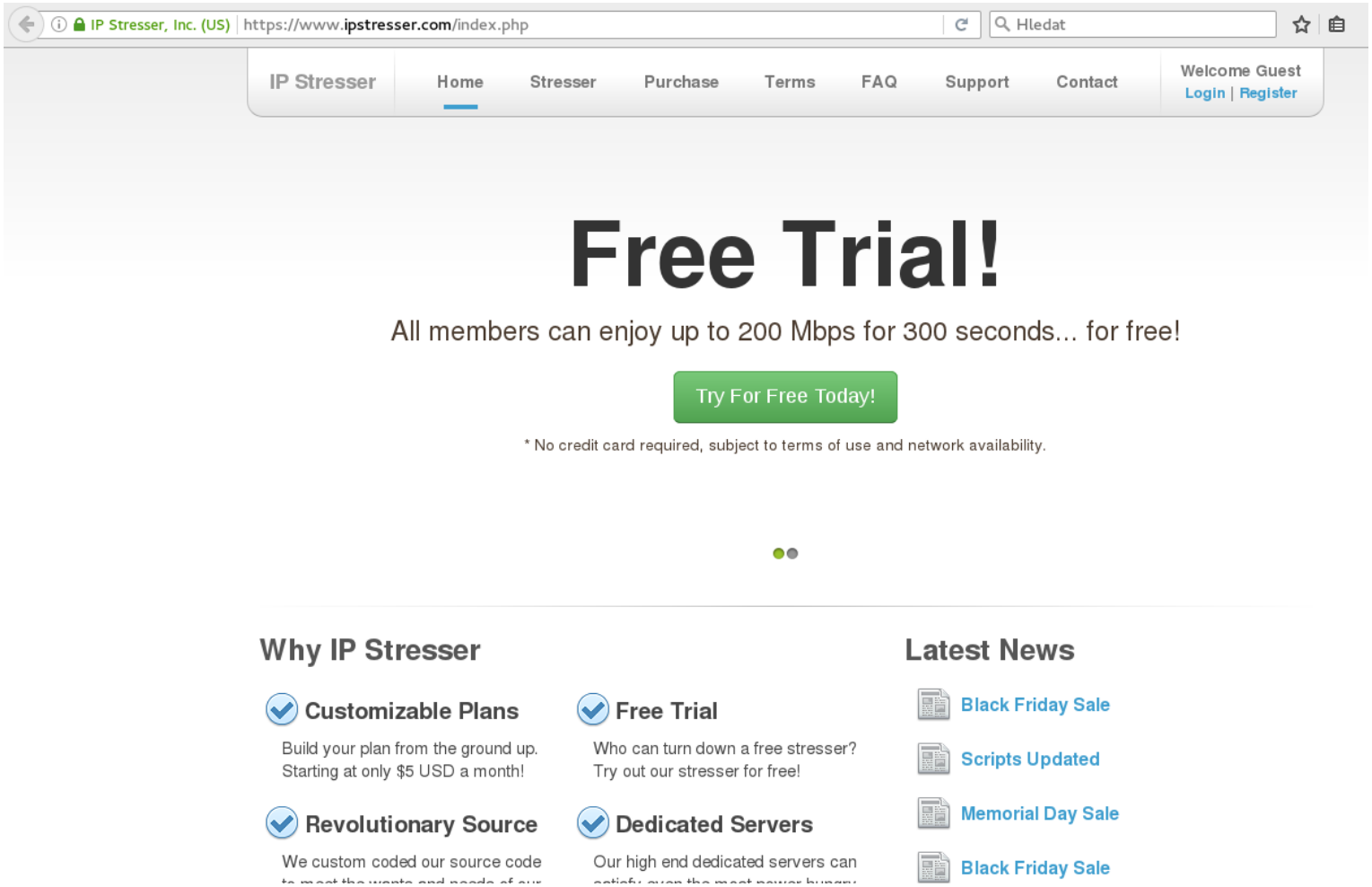


How does it look like?

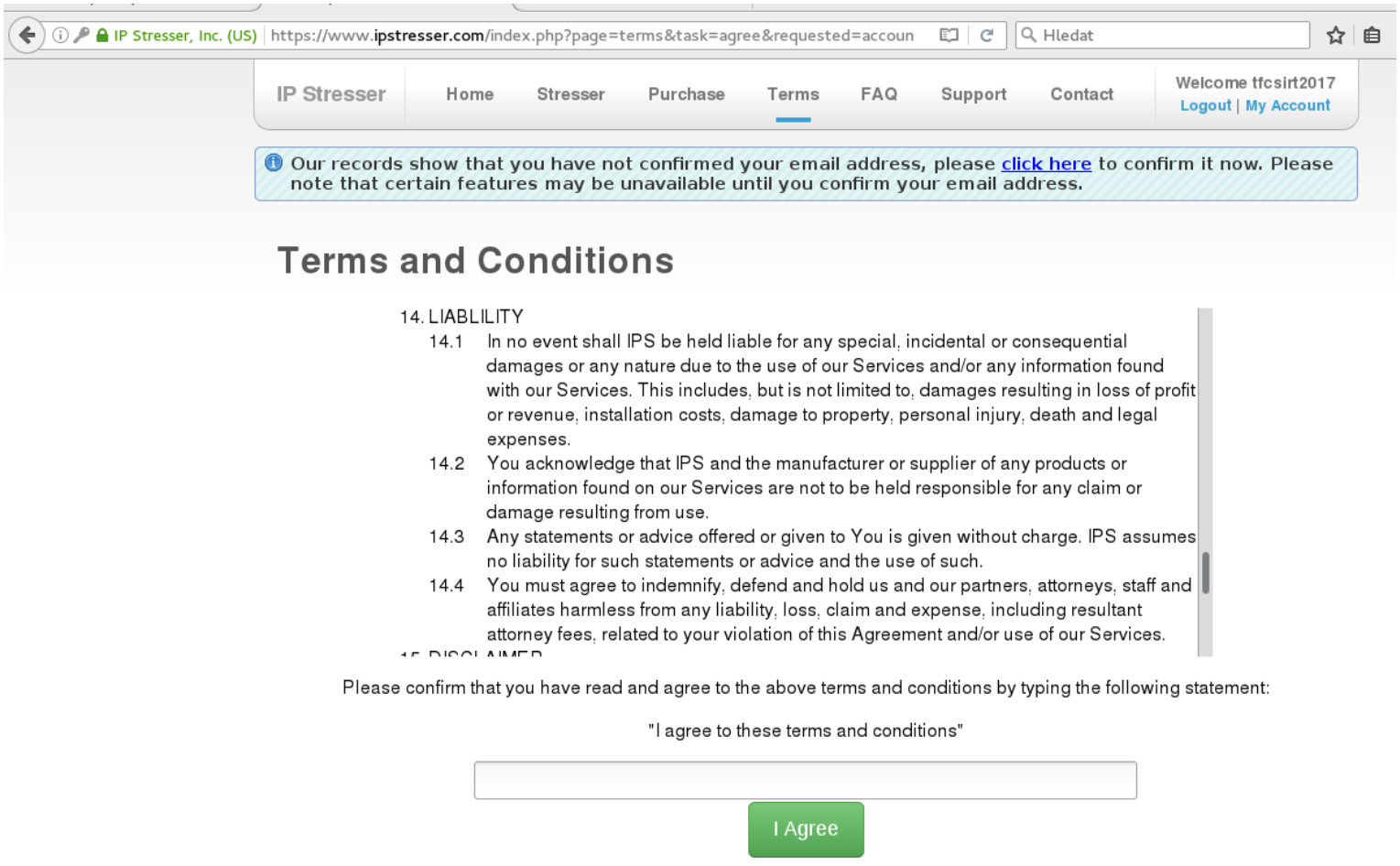


#	Method Name	Method Type	Target Type	Target Syntax
1	GET-BOTNET	Layer 7	Websites, WebServers, etc ..	URL: http://target.com
2	HEAD-BOTNET	Layer 7	Websites, WebServers, etc ..	URL: http://target.com
3	POST-BOTNET	Layer 7	Websites, WebServers, etc ..	URL: http://target.com
4	JSBYPASS-BOTNET	Layer 7	Websites, WebServers, etc ..	URL: http://target.com
5	JOOMLA	Layer 7	Websites, WebServers, etc ..	URL: http://target.com
6	SNMP	Layer 4	Home / Peoples, Servers, Custom IPs, etc ..	IP: 1.3.3.7
7	SSDP	Layer 4	Home / Peoples, Servers, Custom IPs, etc ..	IP: 1.3.3.7
8	DNS	Layer 4	Home / Peoples, Servers, Custom IPs, etc ..	IP: 1.3.3.7
9	CHARGEN	Layer 4	Home / Peoples, Servers, Custom IPs, etc ..	IP: 1.3.3.7
10	NTP	Layer 4	Home / Peoples, Servers, Custom IPs, etc ..	IP: 1.3.3.7
11	TS3	Layer 4	Home / Peoples, Servers, Custom IPs, etc ..	IP: 1.3.3.7
12	SSYN	Layer 4	Home / Peoples, Servers, Custom IPs, etc ..	IP: 1.3.3.7
13	DOMINATE	Layer 4	Home / Peoples, Servers, Custom IPs, etc ..	IP: 1.3.3.7
14	ACK	Layer 4	Home / Peoples, Servers, Custom IPs, etc ..	IP: 1.3.3.7
15	NGSSYN	Layer 4	Home / Peoples, Servers, Custom IPs, etc ..	IP: 1.3.3.7
16	OVX	Layer 4	Home / Peoples, Servers, Custom IPs, etc ..	IP: 1.3.3.7
17	TCPACK	Layer 4	Home / Peoples, Servers, Custom IPs, etc ..	IP: 1.3.3.7
18	TCPSYN	Layer 4	Home / Peopl ervers, Custom IPs, etc ..	IP: 1.3.3.7

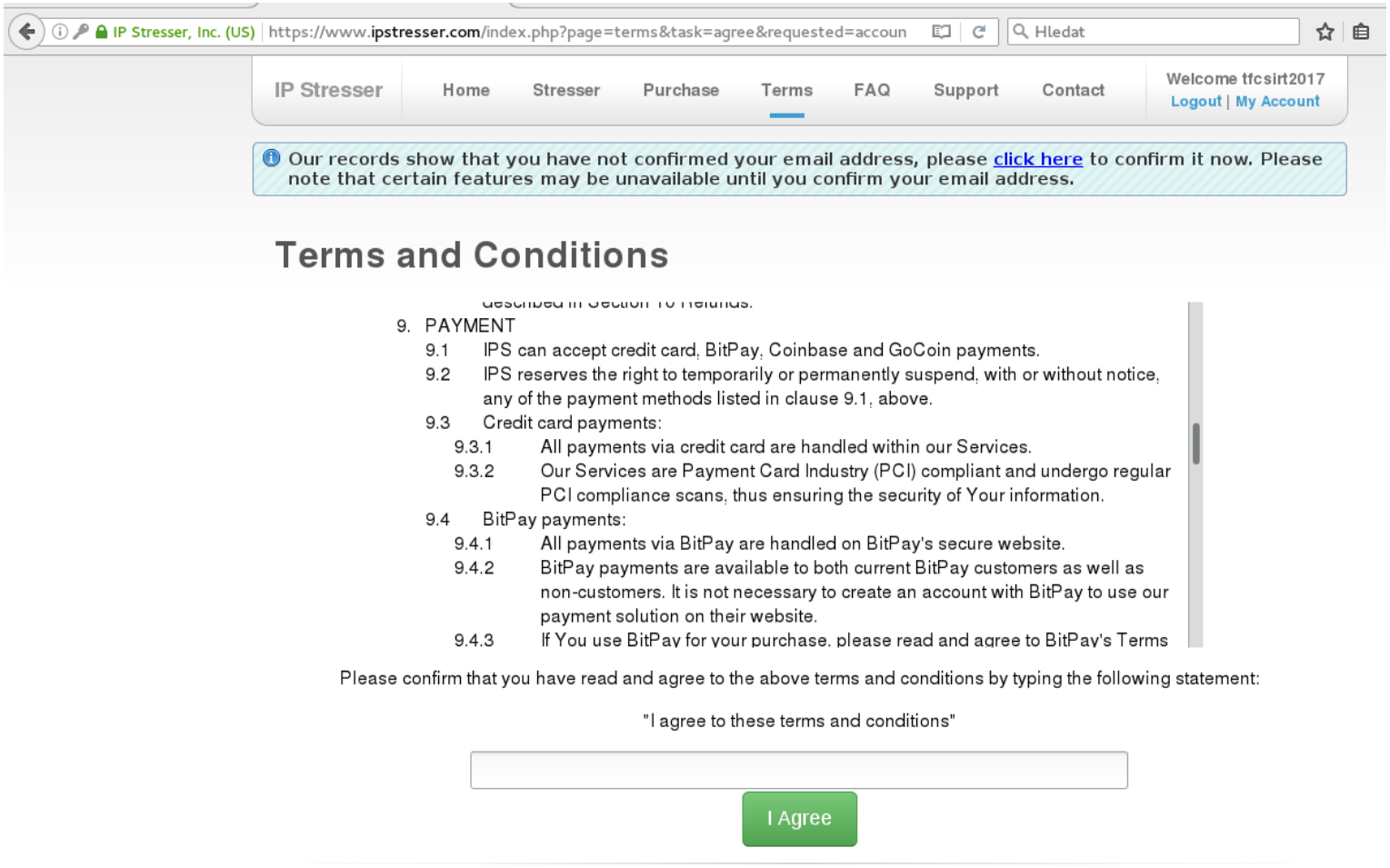
Another Real Use-Case



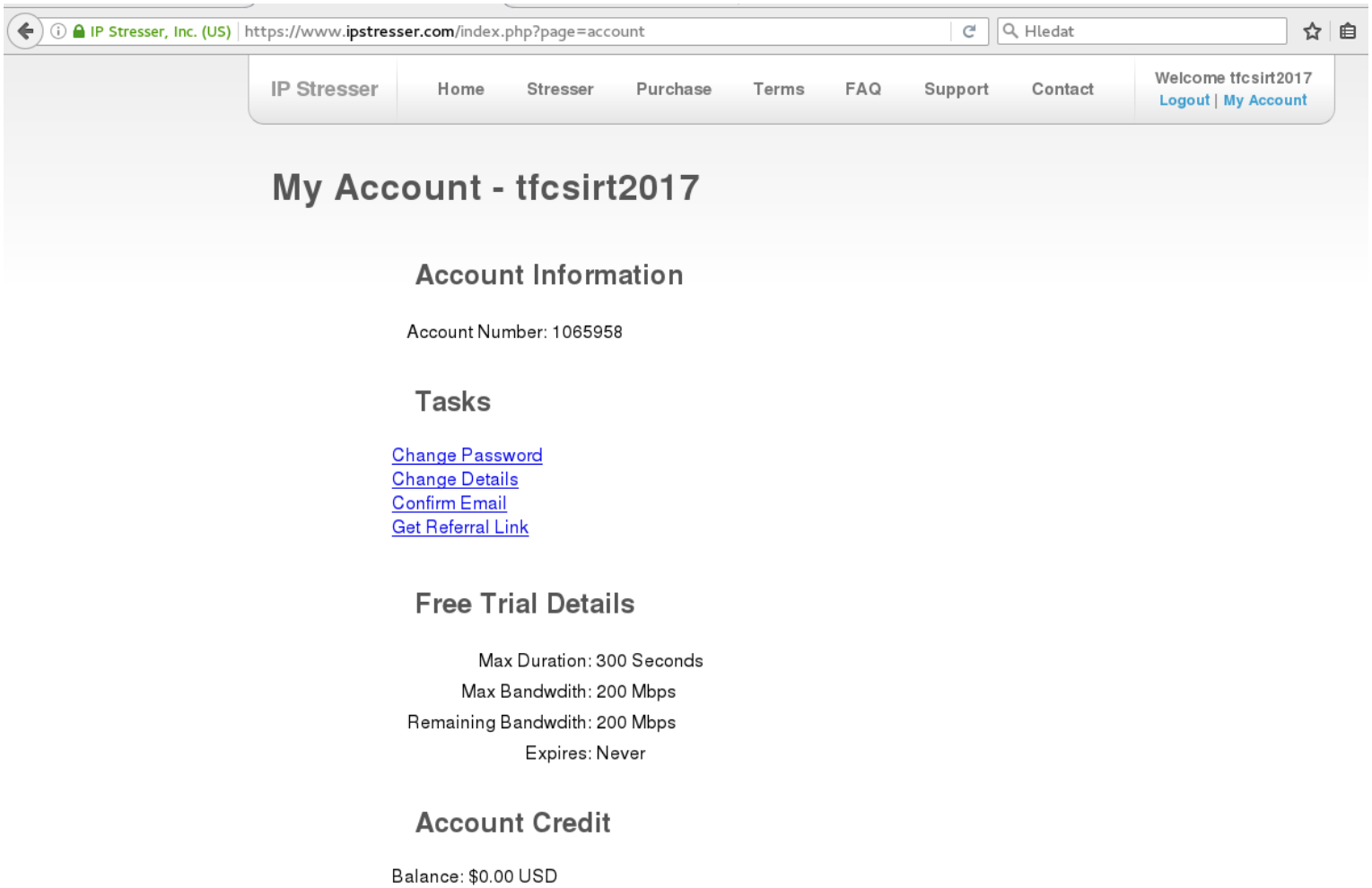
Another Real Use-Case



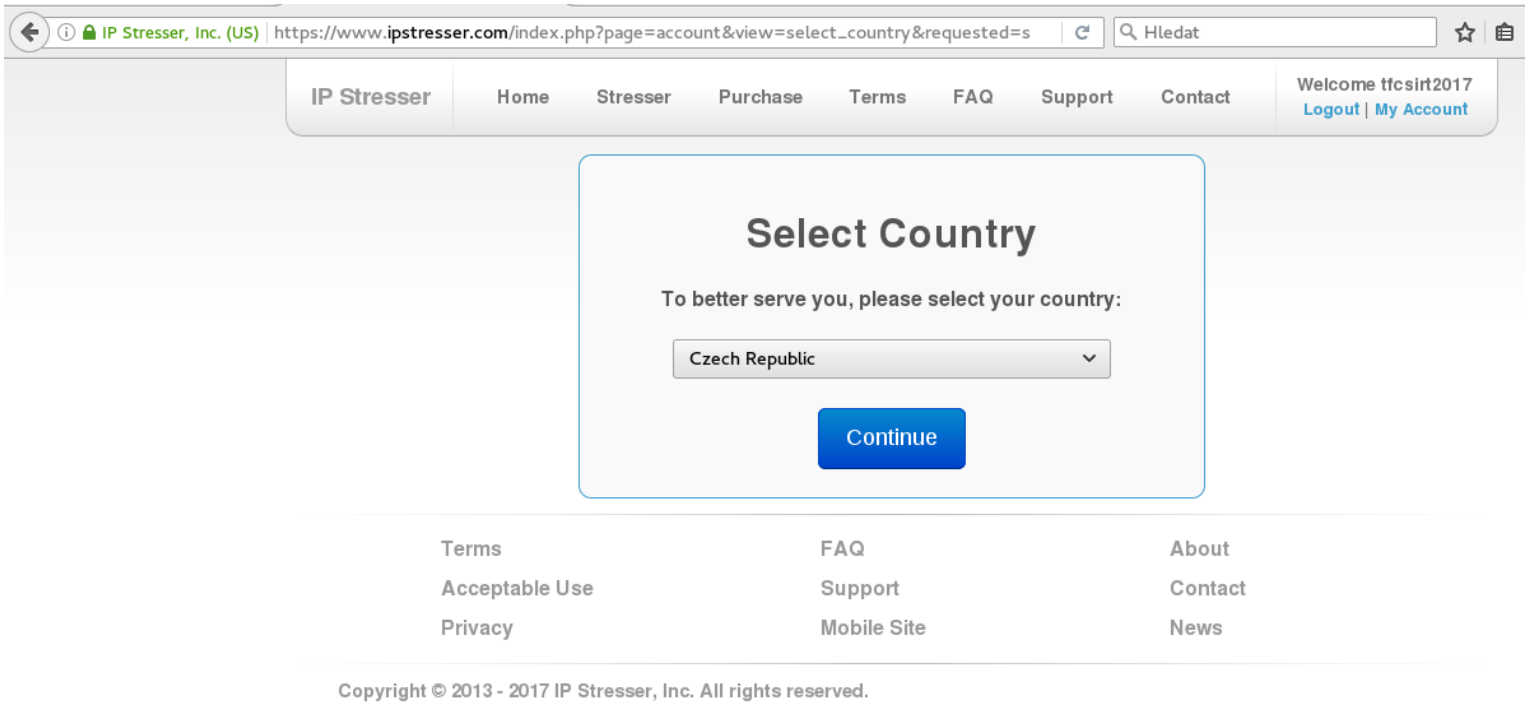
Another Real Use-Case



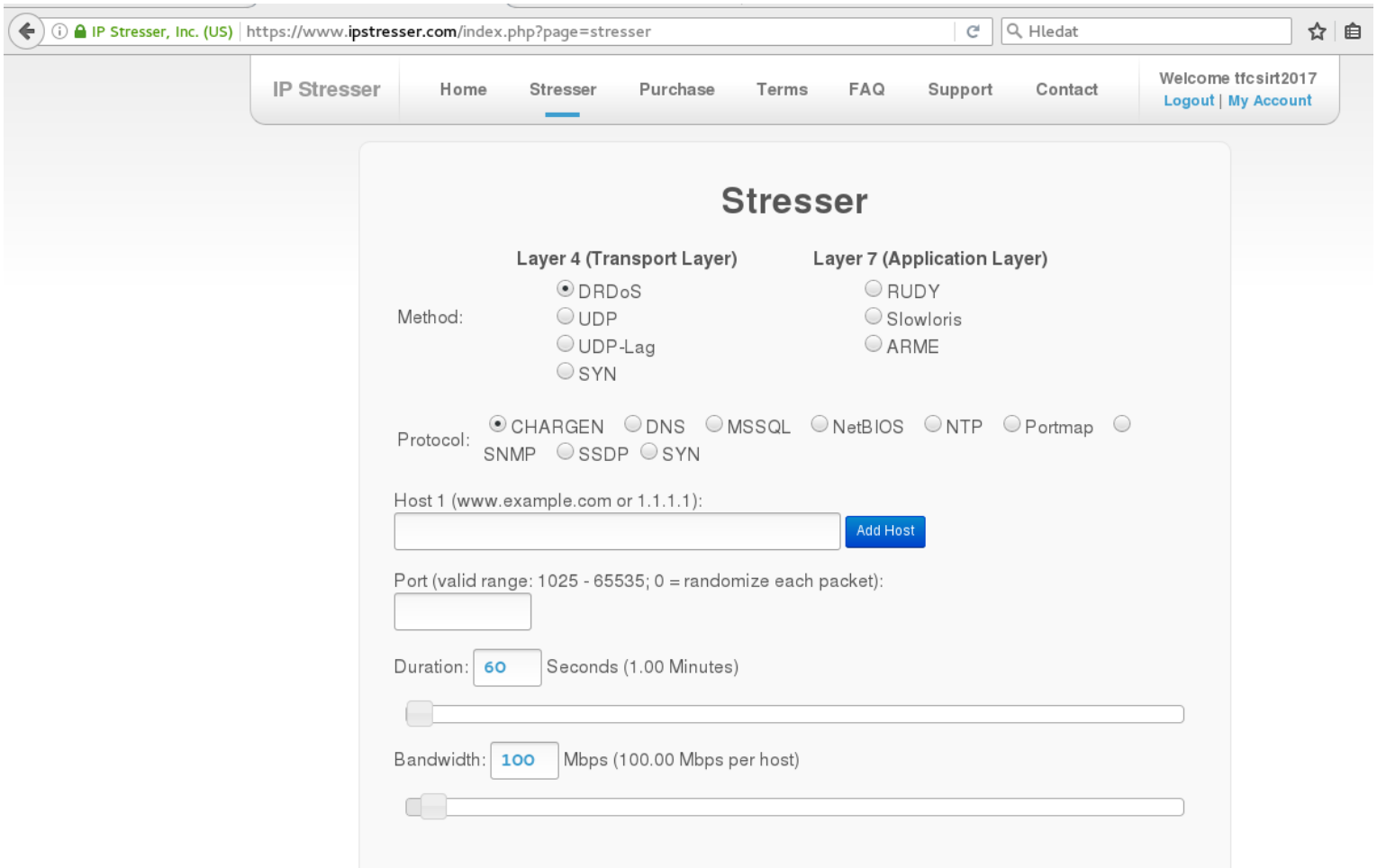
Another Real Use-Case



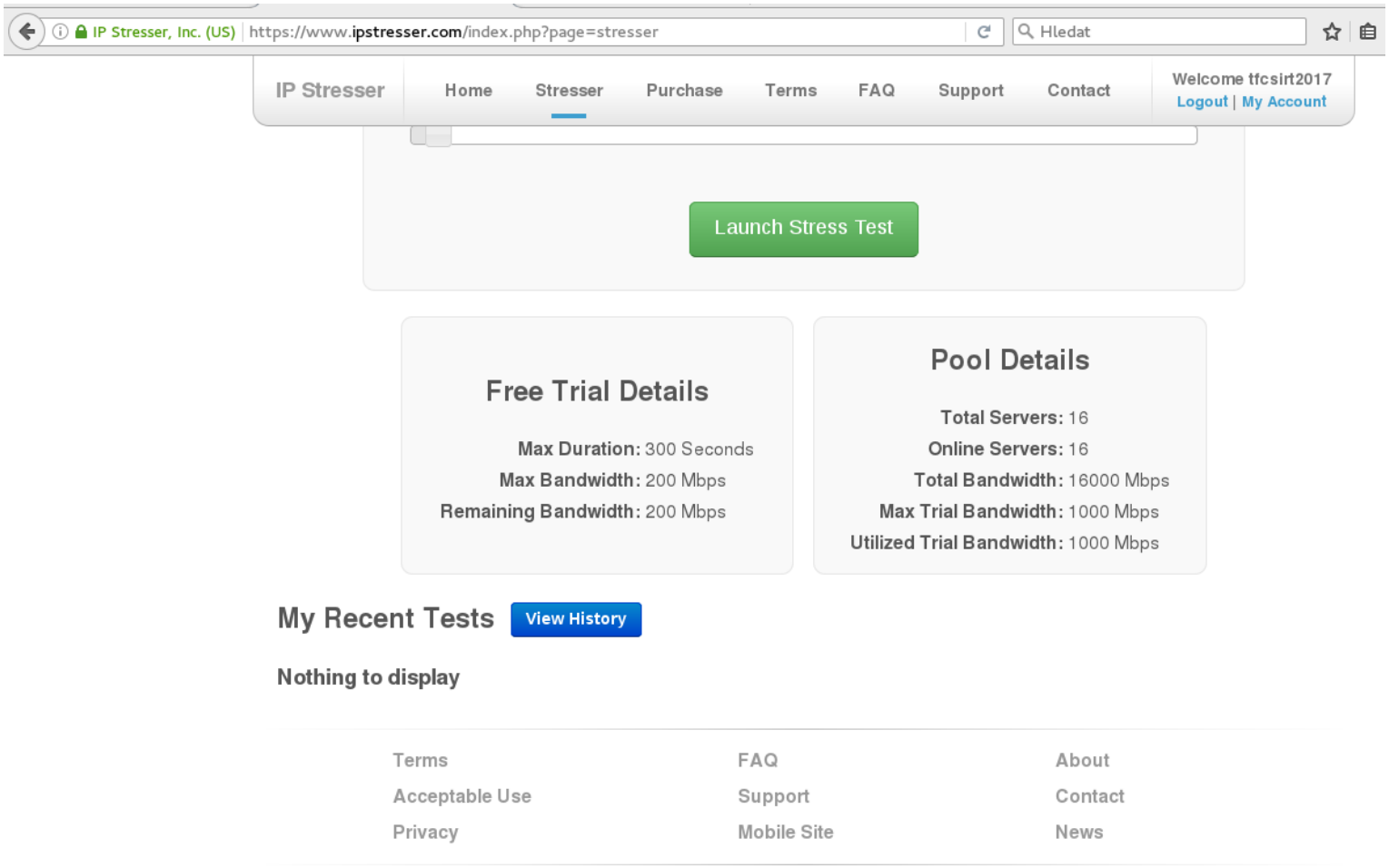
Another Real Use-Case



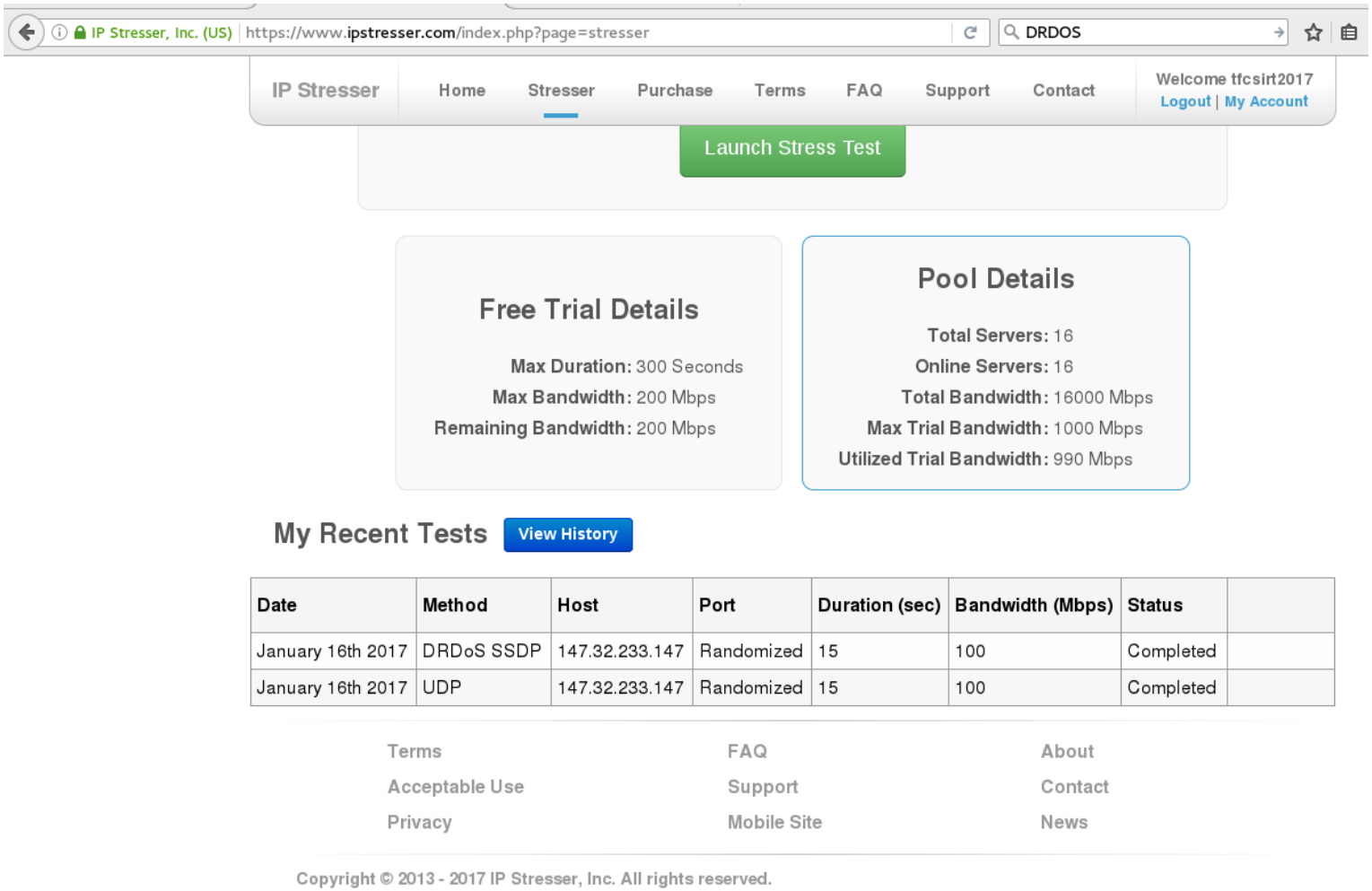
Another Real Use-Case



Another Real Use-Case



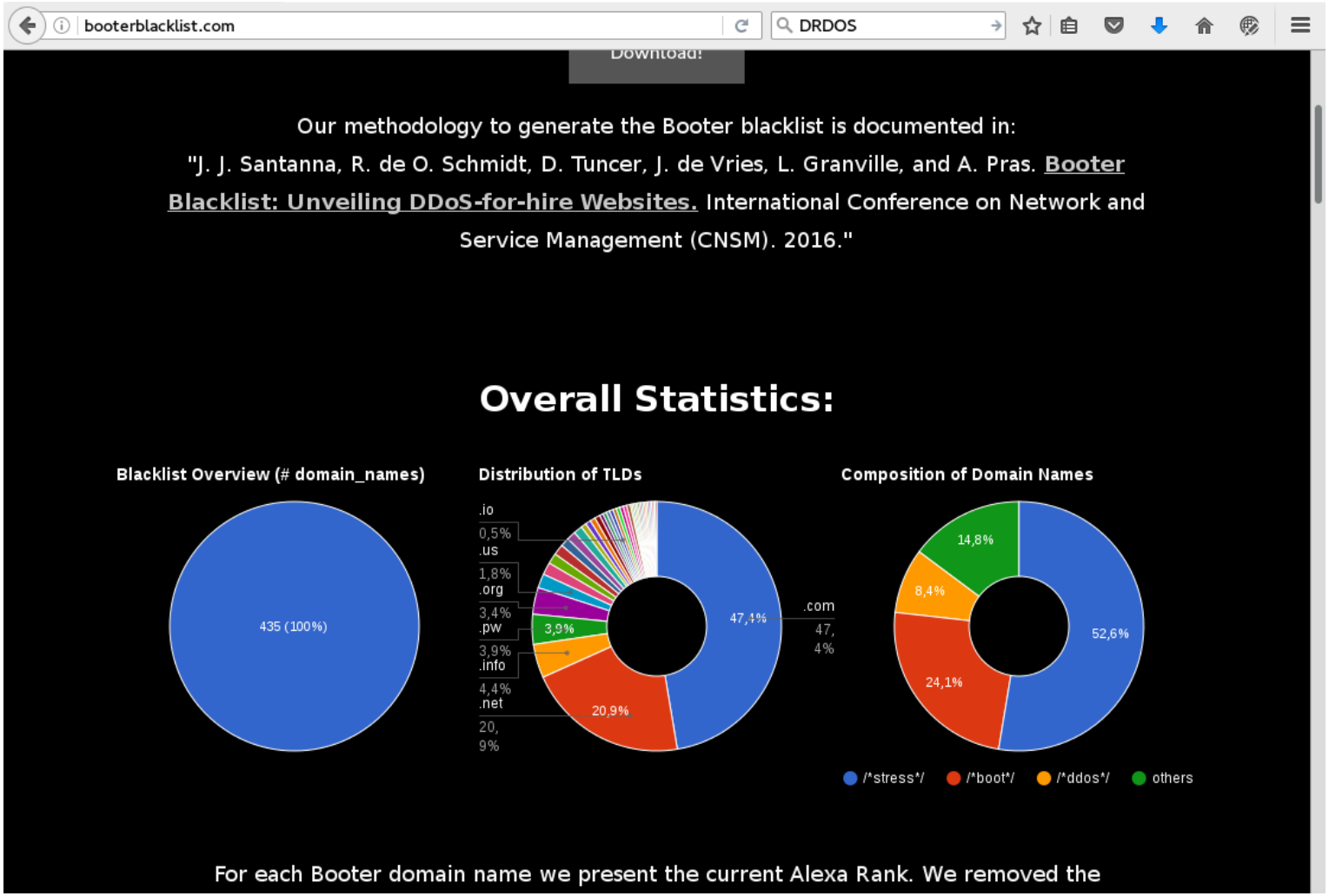
Another Real Use-Case



Date	Method	Host	Port	Duration (sec)	Bandwidth (Mbps)	Status	
January 16th 2017	DRDoS SSDP	147.32.233.147	Randomized	15	100	Completed	
January 16th 2017	UDP	147.32.233.147	Randomized	15	100	Completed	

Booter blacklist

http://booterblacklist.com



Blacklist for detection

Let's have a look into `/home/nemea/booters/`

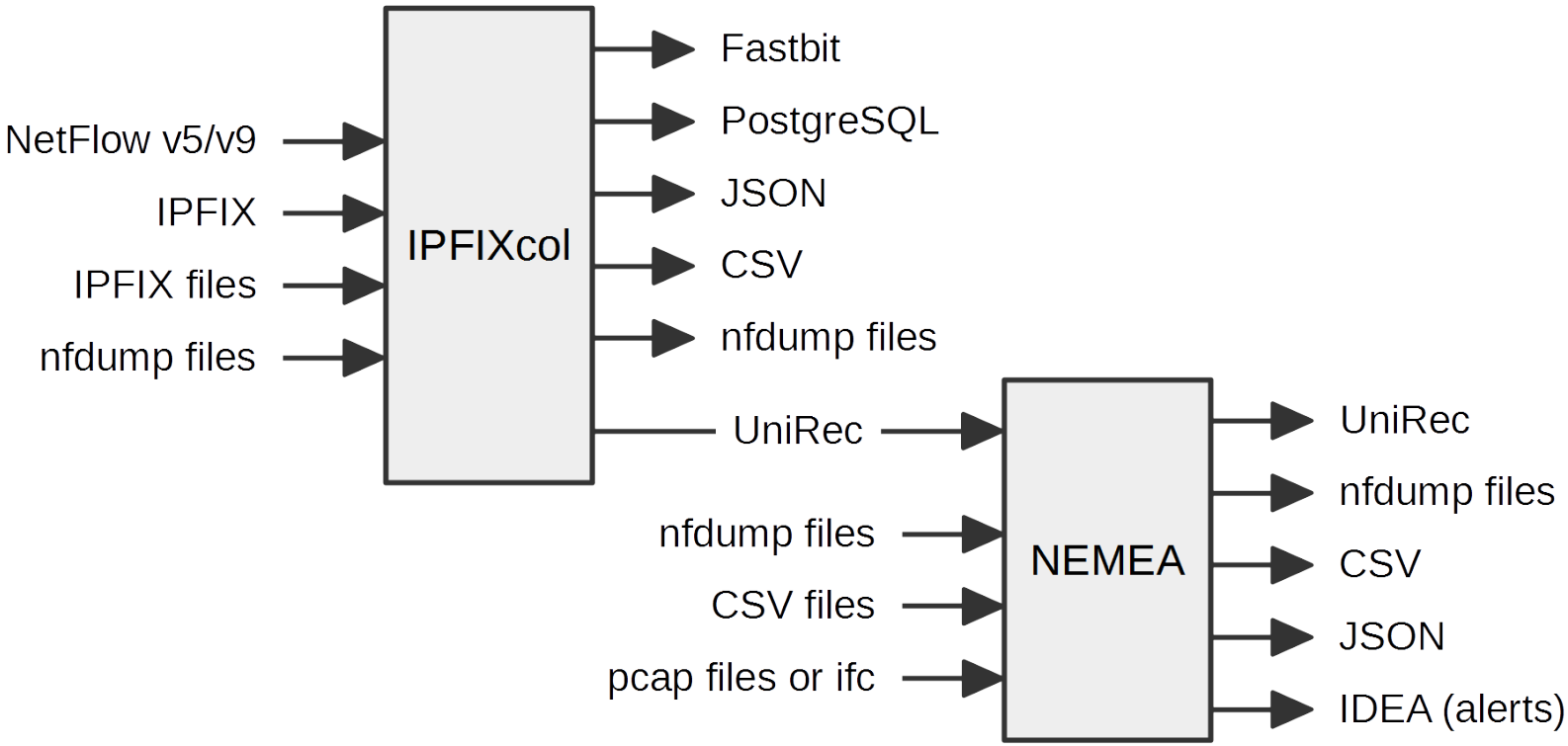
- `booterfilter.cron` URL of blacklist
- `blacklist.txt` downloaded blacklist
- `filter` data prepared for `unirecfilter`

Over 314 thousand observed flow records since Nov 2016.
Containing many pingbacks by WordPress.

Part IX

Compatibility with other systems

Supported input and output data formats



Part X

OpenWrt and EduroamAP

OpenWrt

Brief information:

- Operating system for embedded devices especially wireless routers
- Based on linux kernel
- Developed as a framework containing a complete toolchain for cross-compiling
- Fully customized firmware extendable by binary packages.
- Open source and free

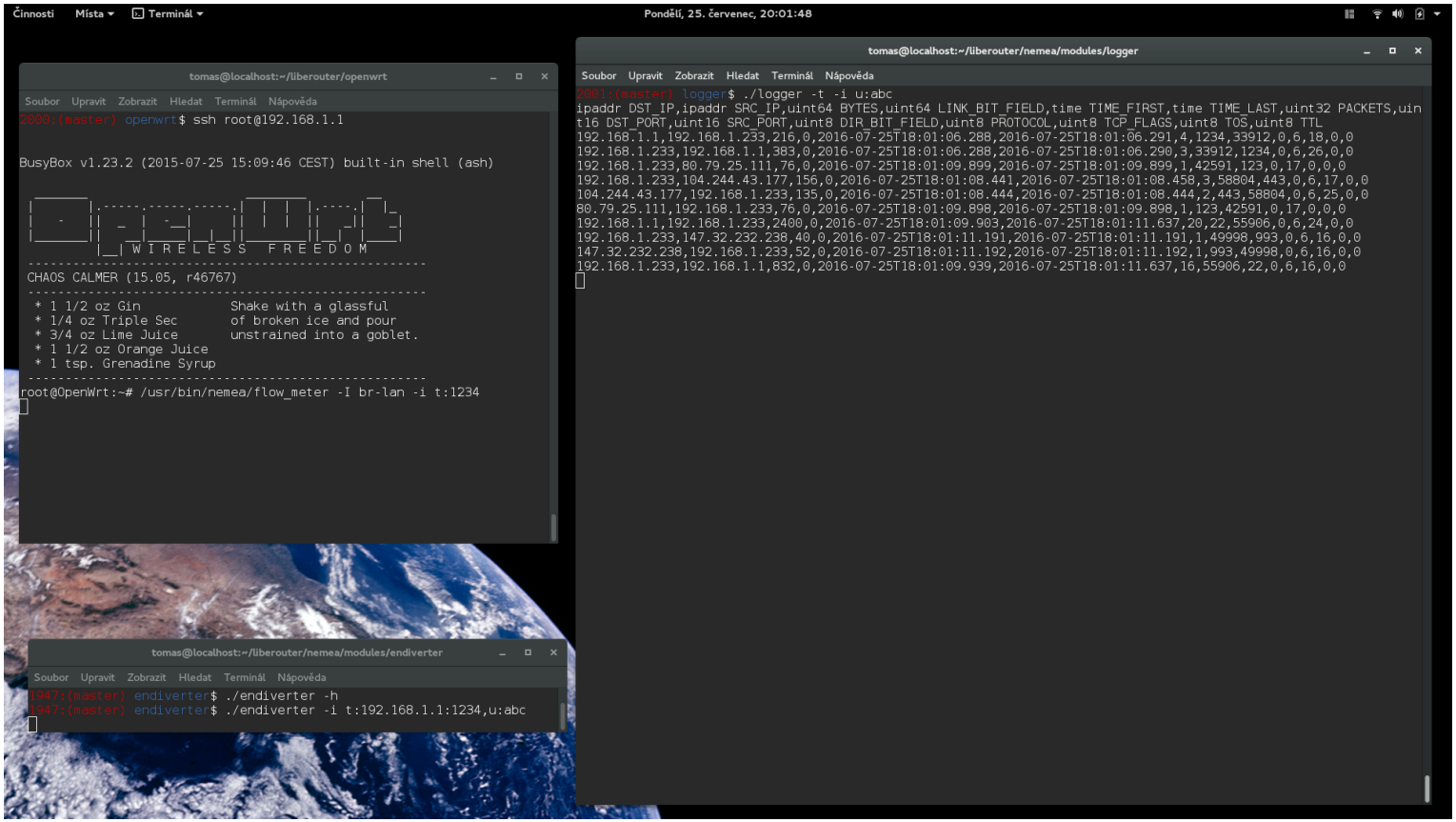
`http://openwrt.org`

OpenWrt and NEMEA

We have created a feed: we can create packages with

- NEMEA framework
- some modules, mainly `flow_meter`

Screenshot


Činnosti
Místa
Terminál
Pondělí, 25. červenec, 20:01:48
tomas@localhost:~/liberouter/openwrt
Soubor Upravit Zobrazit Hledat Terminál Nápověda
2000:(master) openwrt$ ssh root@192.168.1.1
BusyBox v1.23.2 (2015-07-25 15:09:46 CEST) built-in shell (ash)
W I R E L E S S F R E E D O M
CHAOS CALMER (15.05, r46767)
* 1 1/2 oz Gin Shake with a glassful
* 1/4 oz Triple Sec of broken ice and pour
* 3/4 oz Lime Juice unstrained into a goblet.
* 1 1/2 oz Orange Juice
* 1 tsp. Grenadine Syrup
root@OpenWrt:~# /usr/bin/nemea/flow_meter -I br-lan -i t:1234
tomas@localhost:~/liberouter/nemea/modules/logger
Soubor Upravit Zobrazit Hledat Terminál Nápověda
2001:(master) logger$ ./logger -t -i u:abc
ipaddr DST_IP,ipaddr SRC_IP,uint64 BYTES,uint64 LINK_BIT_FIELD,time TIME_FIRST,time TIME_LAST,uint32 PACKETS,uint16 DST_PORT,uint16 SRC_PORT,uint8 DIR_BIT_FIELD,uint8 PROTOCOL,uint8 TCP_FLAGS,uint8 TOS,uint8 TTL
192.168.1.1,192.168.1.233,216,0,2016-07-25T18:01:06.288,2016-07-25T18:01:06.291,4,1234,33912,0,6,18,0,0
192.168.1.233,192.168.1.1,383,0,2016-07-25T18:01:06.288,2016-07-25T18:01:06.290,3,33912,1234,0,6,26,0,0
192.168.1.233,80.79.25.111,76,0,2016-07-25T18:01:09.899,2016-07-25T18:01:09.899,1,42591,123,0,17,0,0,0
192.168.1.233,104.244.43.177,156,0,2016-07-25T18:01:08.441,2016-07-25T18:01:08.458,3,58804,443,0,6,17,0,0
104.244.43.177,192.168.1.233,135,0,2016-07-25T18:01:08.444,2016-07-25T18:01:08.444,2,443,58804,0,6,25,0,0
80.79.25.111,192.168.1.233,76,0,2016-07-25T18:01:09.898,2016-07-25T18:01:09.898,1,123,42591,0,17,0,0,0
192.168.1.1,192.168.1.233,2400,0,2016-07-25T18:01:09.903,2016-07-25T18:01:11.637,20,22,55906,0,6,24,0,0
192.168.1.233,147.32.232.238,40,0,2016-07-25T18:01:11.191,2016-07-25T18:01:11.191,1,49998,993,0,6,16,0,0
147.32.232.238,192.168.1.233,52,0,2016-07-25T18:01:11.192,2016-07-25T18:01:11.192,1,993,49998,0,6,16,0,0
192.168.1.233,192.168.1.1,832,0,2016-07-25T18:01:09.939,2016-07-25T18:01:11.637,16,55906,22,0,6,16,0,0
tomas@localhost:~/liberouter/nemea/modules/endiverter
Soubor Upravit Zobrazit Hledat Terminál Nápověda
1947:(master) endiverter$ ./endiverter -h
1947:(master) endiverter$ ./endiverter -i t:192.168.1.1:1234,u:abc

EduroamAP

Brief info:

- Device powered by OpenWrt, SW assembled and prepared by CESNET
- OpenWrt system with openvpn detects internet connection
- VPN tunnel for provisioning and infrastructure monitoring
- solved certificates distribution
- **eduroam** access point out-of-box
- Contact: jan.tomasek@cesnet.cz

Working prototypes

Working prototypes