D4 Project

Open and collaborative network monitoring

Team CIRCL
https://www.d4-project.org/

2019/07/03

Problem statement

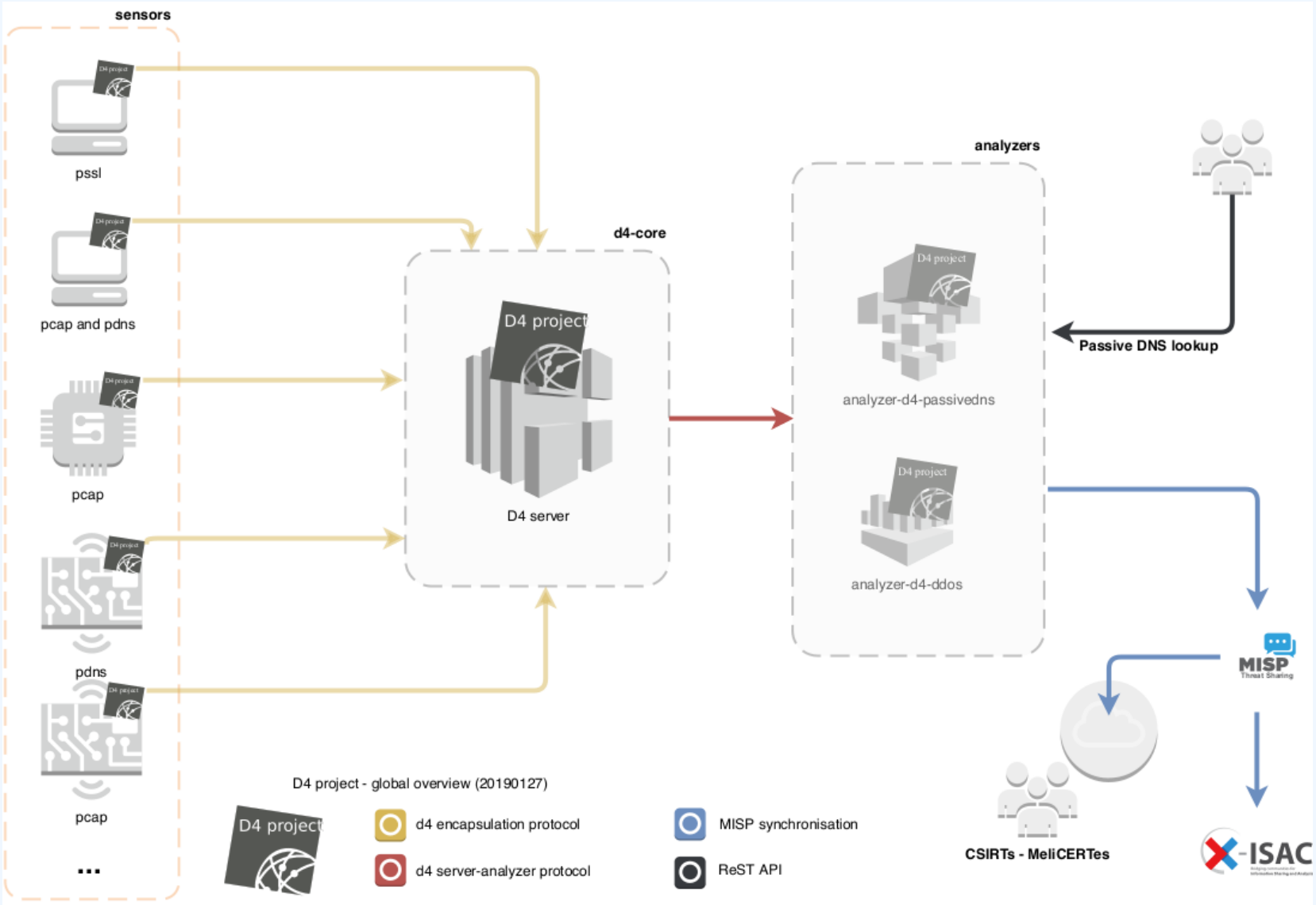
- CSIRTs (or private organisations) build their **own honeypot, honeynet or blackhole monitoring network**
- Designing, managing and operating such infrastructure is a tedious and resource intensive task
- **Automatic sharing** between monitoring networks from different organisations is missing
- Sensors and processing are often seen as blackbox or difficult to audit

Objective

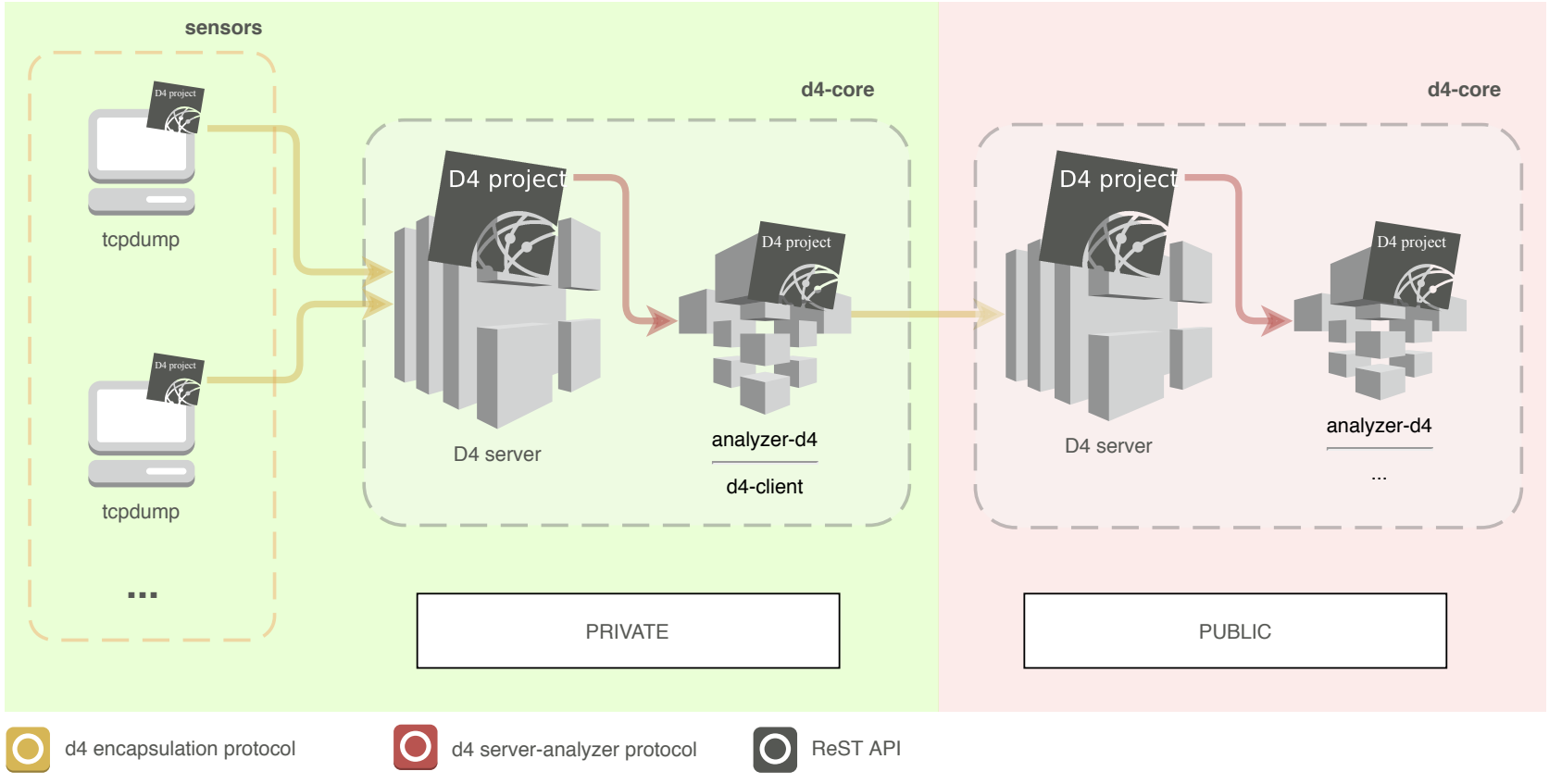
- Based on our experience with MISP[1] where sharing played an important role, we transpose the model in D4 project
- Keeping the protocol and code base **simple and minimal**
- Allowing every organisation to **control and audit their own sensor network**
- Extending D4 or **encapsulating legacy monitoring protocols** must be as simple as possible
- Ensuring that the sensor server has **no control on the sensor** (unidirectional streaming)
- Don't force users to use dedicated sensors and allow **flexibility of sensor support** (software, hardware, virtual)

[1] https://github.com/MISP/MISP

D4 Overview


sensors
pssl
pcap and pdns
pcap
pdns
pcap
...
d4-core
D4 project
D4 server
analyzers
analyzer-d4-passivedns
analyzer-d4-ddos
Passive DNS lookup
MISP
Threat Sharing
CSIRTs - MeliCERTes
-ISAC
D4 project - global overview (20190127)
D4 project
d4 encapsulation protocol
d4 server-analyzer protocol
MISP synchronisation
ReST API

D4 Overview


sensors
D4 project
tcpdump
D4 project
tcpdump
...
d4-core
D4 project
D4 server
D4 project
analyzer-d4
d4-client
PRIVATE
d4-core
D4 project
D4 server
D4 project
analyzer-d4
...
PUBLIC
d4 encapsulation protocol
d4 server-analyzer protocol
ReST API

(short) History

- D4 Project (co-funded under INEA CEF EU program) started - **1st November 2018**
- D4 encapsulation protocol version 1 published - **1st December 2018**
- v0.1 release of the D4 core[2] including a server and simple D4 C client - **21st January 2019**
- First version of a golang D4 client[3] running on ARM, MIPS, PPC and x86 - **14th February 2019**

[2] https://www.github.com/D4-project/d4-core
[3] https://www.github.com/D4-project/d4-goclient/

(short) History

Release	Date
analyzer-d4-passivedns-v0.1	Apr. 5, 2019
analyzer-d4-passivessl-0.1	Apr. 25, 2019
analyzer-d4-pibs-v0.1	Apr. 8, 2019
BGP-Ranking-1.0	Apr. 25, 2019
d4-core-v0.1	Jan. 25, 2019
d4-core-v0.2	Feb. 14, 2019
d4-core-v0.3	Apr. 8, 2019
d4-goclient-v0.1	Feb. 14, 2019
d4-goclient-v0.2	Apr. 8, 2019
d4-server-packer-0.1	Apr. 25, 2019
IPASN-History-1.0	Apr. 25, 2019
sensor-d4-tls-fingerprinting-0.1	Apr. 25, 2019

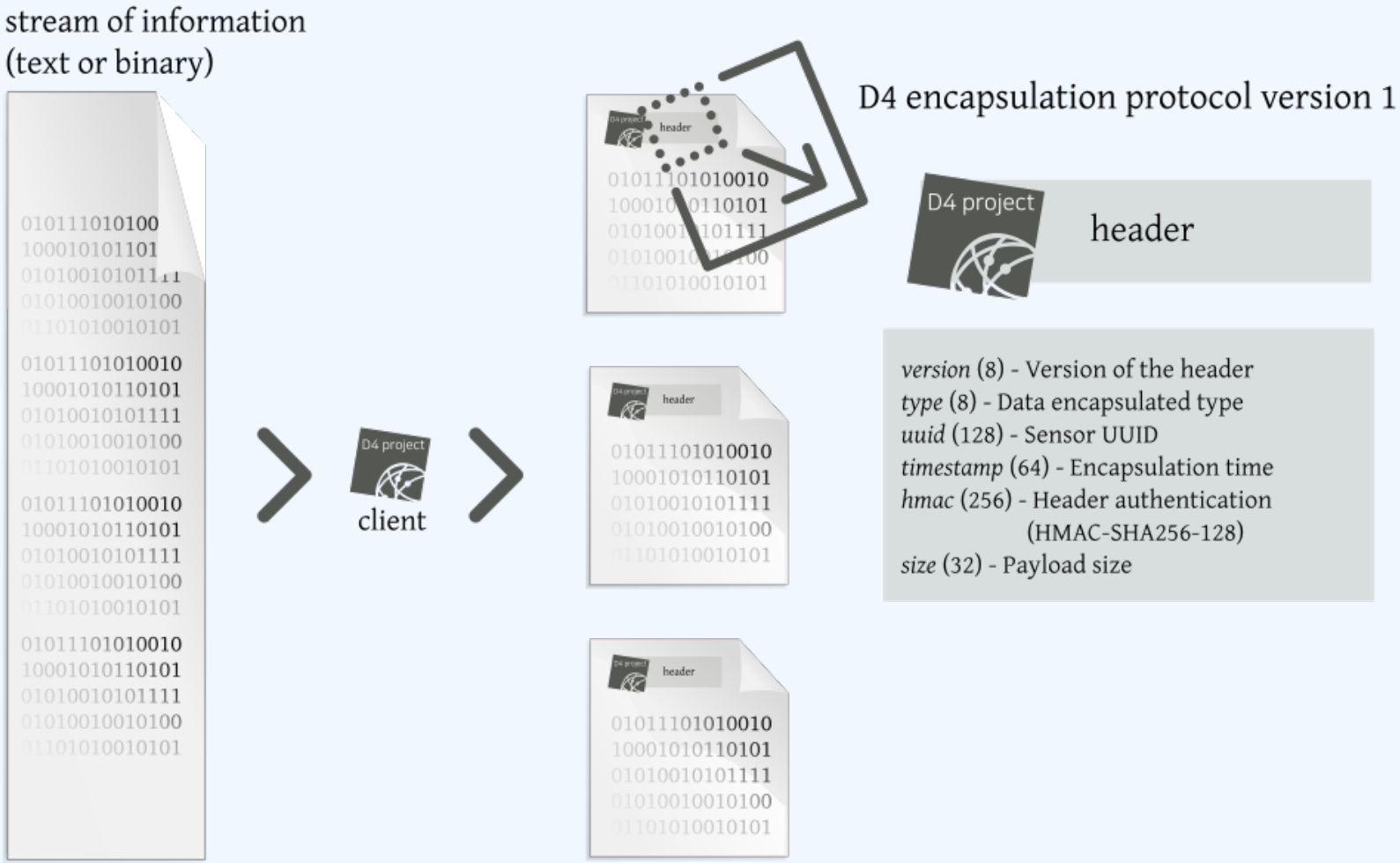
see `https://github.com/D4-Project`

Roadmap - output

CIRCL will host a server instance for organisations willing to contribute to a public dataset without running their own D4 server:

- ✓ Blackhole DDoS
- ✓ Passive DNS
- ✓ Passive SSL
- BGP mapping
- egress filtering mapping
- Radio-Spectrum monitoring: 802.11, BLE, ~~GSM~~, etc.
- ...

D4 encapsulation protocol



https://www.d4-project.org

D4 Header

Name	bit size	Description
version	uint 8	Version of the header
type	uint 8	Data encapsulated type
uuid	uint 128	Sensor UUID
timestamp	uint 64	Encapsulation time
hmac	uint 256	Authentication header (HMAC-SHA-256-128)
size	uint 32	Payload size

D4 Header

Type	Description
0	Reserved
1	pcap (libpcap 2.4)
2	meta header (JSON)
3	generic log line
4	dnscap output
5	pcapng (diagnostic)
6	generic NDJSON or JSON Lines
7	generic YAF (Yet Another Flowmeter)
8	passivedns CSV stream
254	type defined by meta header (type 2)

D4 meta header

D4 header includes an easy way to **extend the protocol** (via type 2) without altering the format. Within a D4 session, the initial D4 packet(s) type 2 defines the custom headers and then the following packets with type 254 is the custom data encapsulated.

```
{
  "type": "ja3-jl",
  "encoding": "utf-8",
  "tags": [
    "tlp:white"
  ],
  "misp:org": "5b642239-4db4-4580-adf4-4ebd950d210f"
}
```

D4 server

- D4 core server[4] is a complete server to handle clients (sensors) including the decapsulation of the D4 protocol, control of sensor registrations, management of decoding protocols and dispatching to adequate decoders/analysers.
- D4 server is written in Python 3.6 and runs on standard GNU/Linux distribution.

[4] https://github.com/D4-project/d4-core

D4 server handling

D4 server reconstructs the encapsulated stream from the D4 sensor and saves it in a Redis stream.

- Support TLS connection
- Unpack D4 header
- Verify client secret key (HMAC)
- check blocklist
- Filter by types (Only accept one connection by type-UUID - except: type 254)
- Discard incorrect data
- Save data in a Redis Stream (unique for each session)

D4 server - worker handler

After the stream is processed depending of the type using dedicated worker.

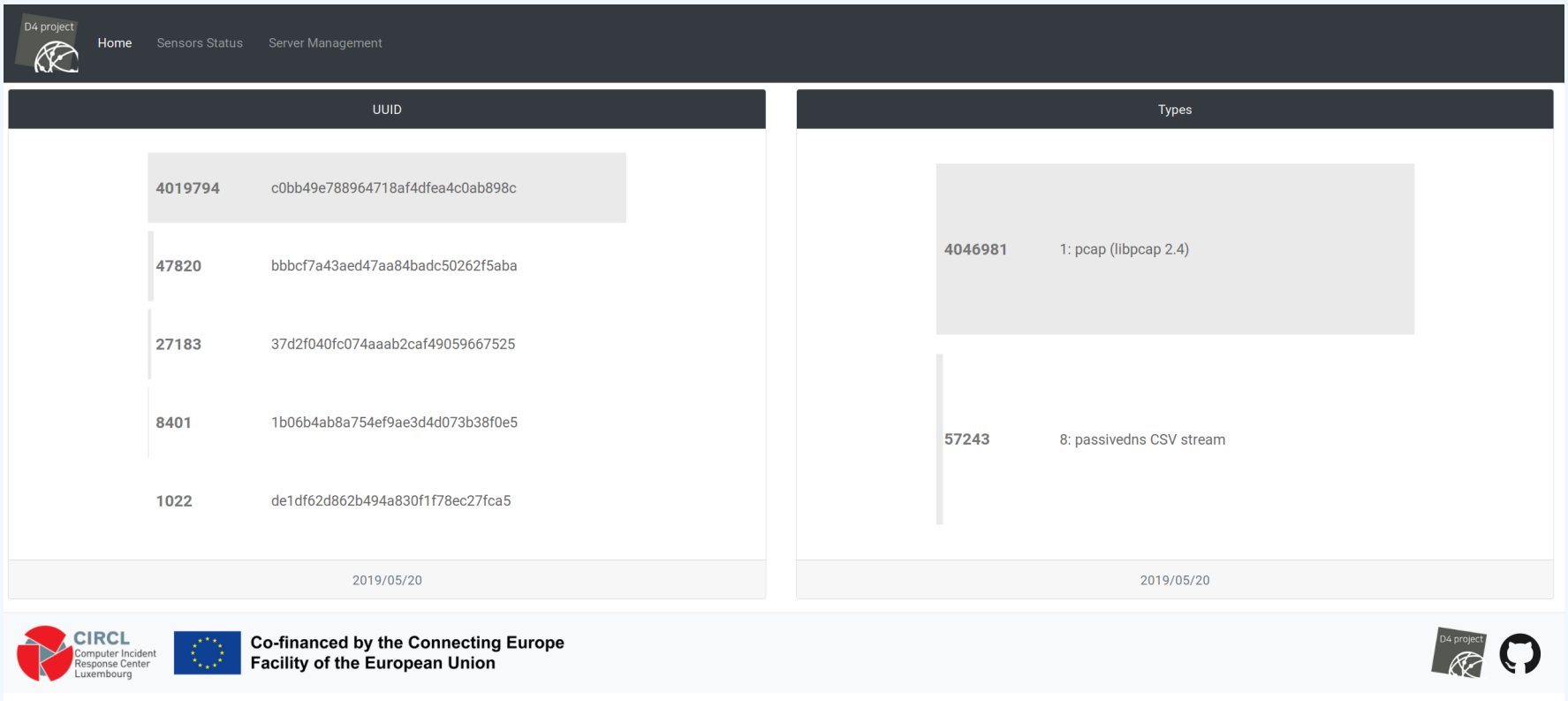
- Worker Manager (one by type)
  - Check if a new session is created and valid data are saved in a Redis stream
  - Launch a new Worker for each session
- Worker
  - Get data from a stream
  - Reconstruct data
  - Save data on disk (with file rotation)
  - Save data in Redis. Create a queue for D4 Analyzer(s)

D4 server - management interface

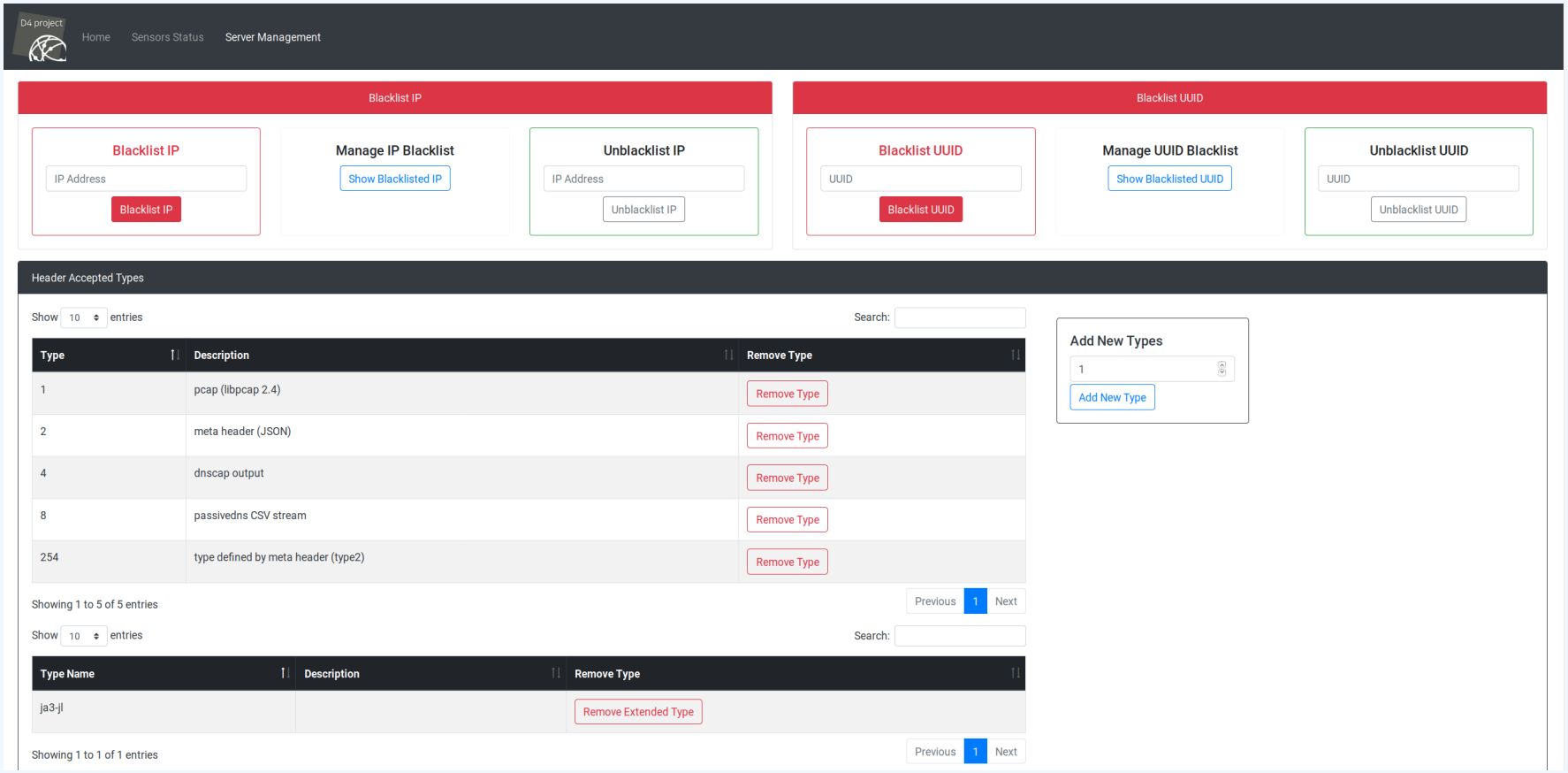
The D4 server provides a **web interface** to manage D4 sensors, sessions and analyzer.

- Get Sensors status, errors and statistics
- Get all connected sensors
- Manage Sensors (stream size limit, secret key, ...)
- Manage Accepted types
- UUID/IP blocklist
- Create Analyzer Queues

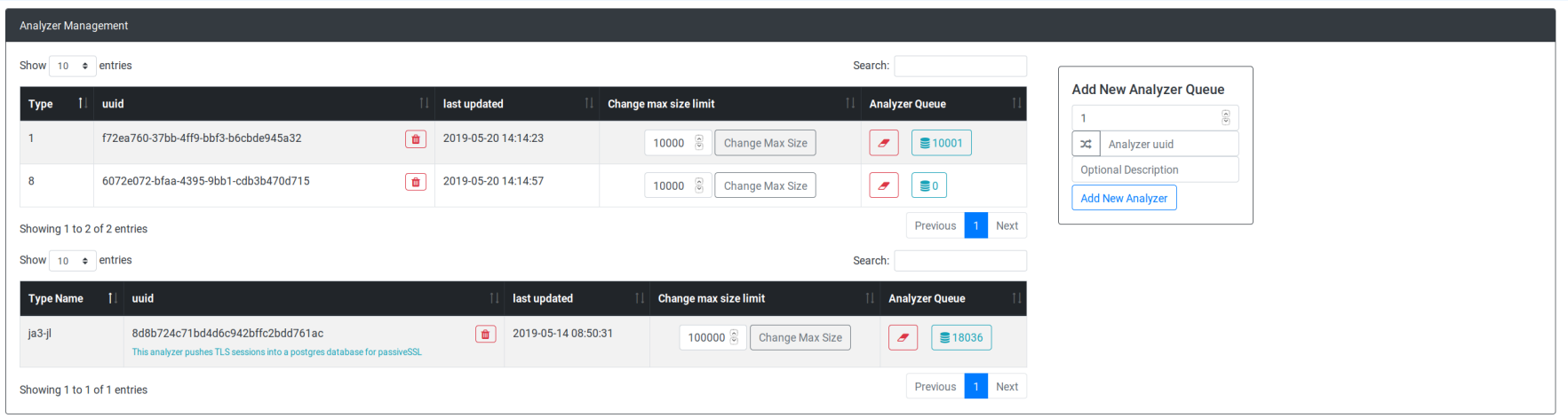
D4 SERVER - MAIN INTERFACE



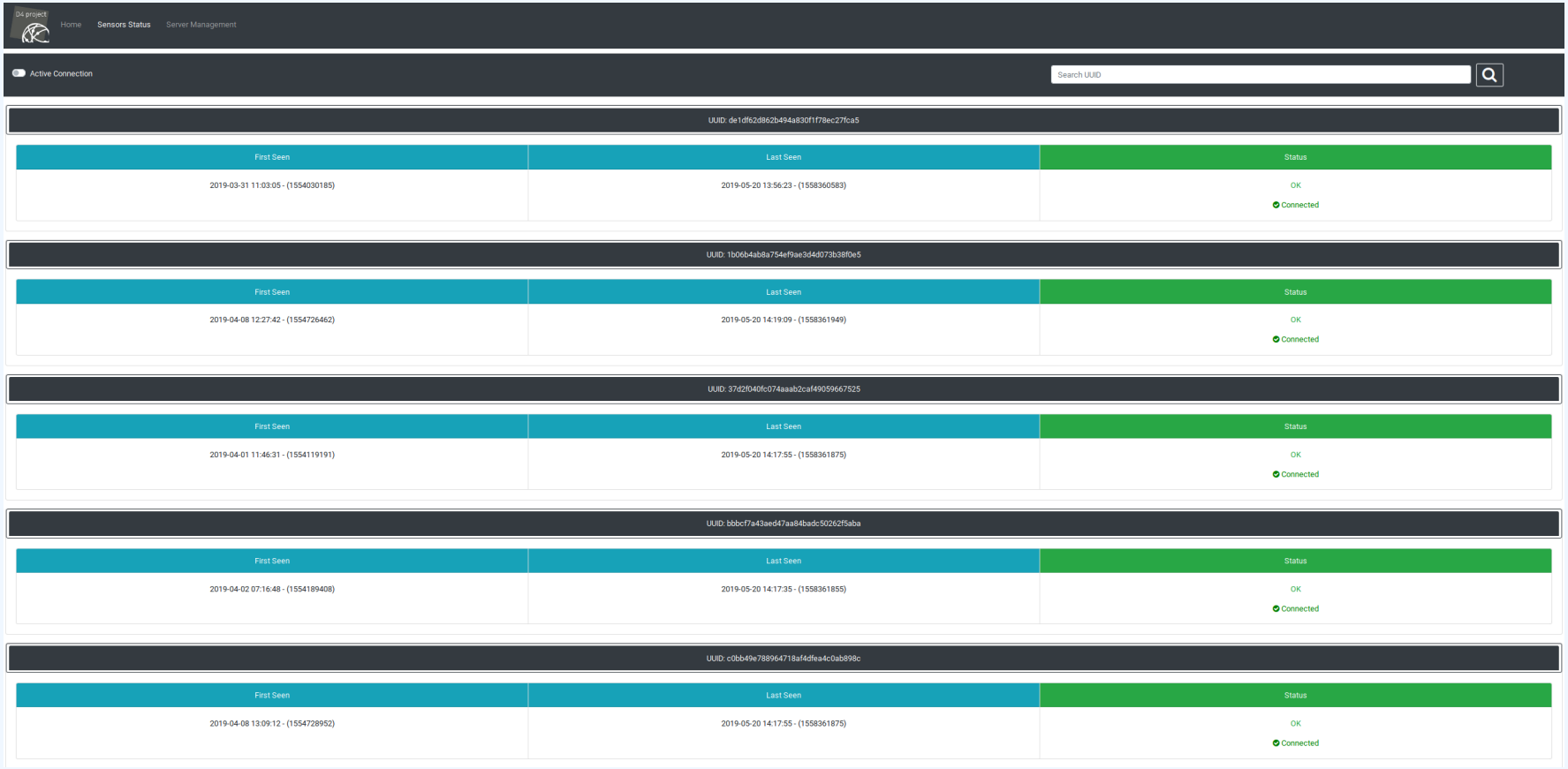
D4 SERVER - SERVER MANAGEMENT



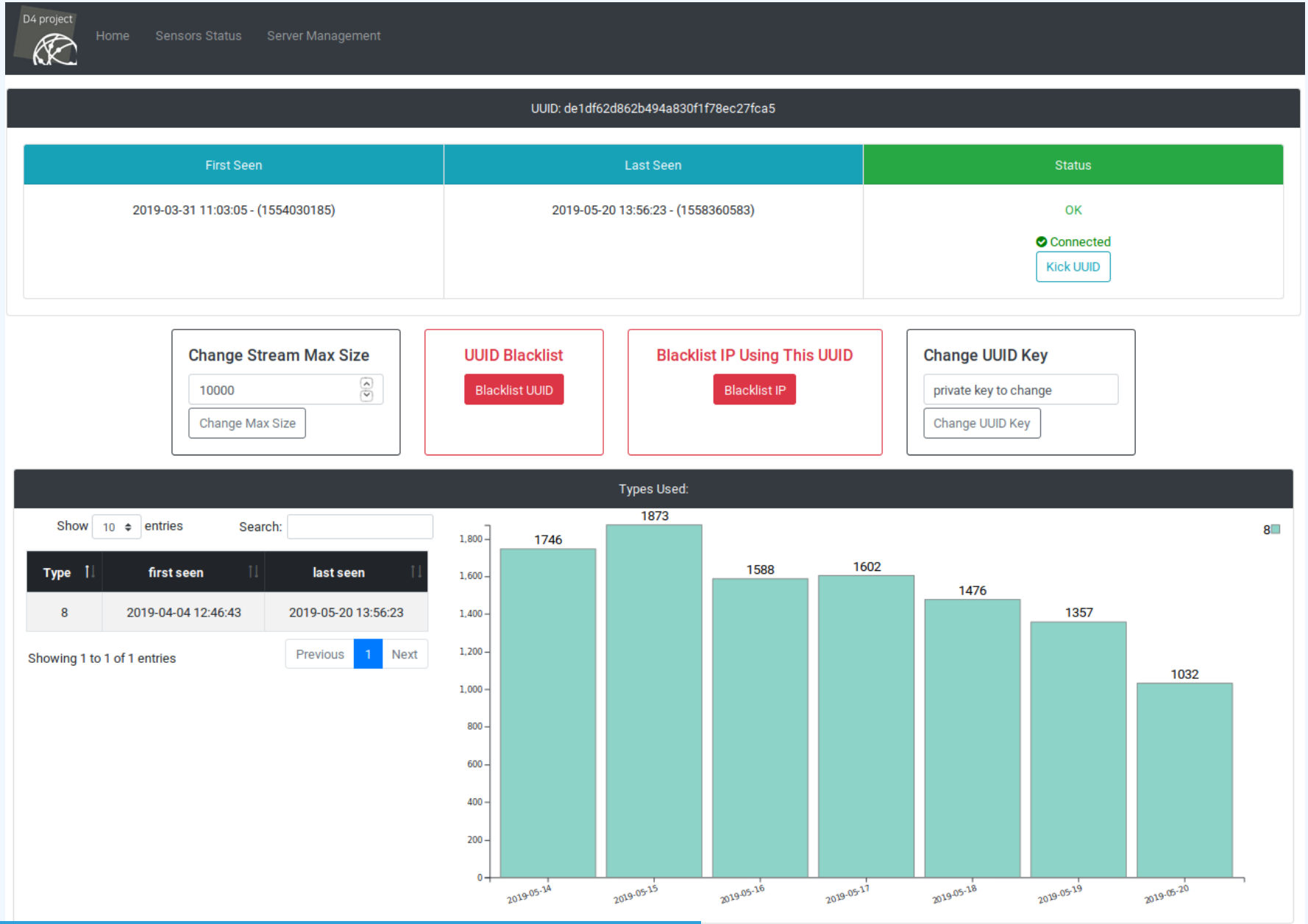
D4 SERVER - SERVER MANAGEMENT



D4 server - sensor overview



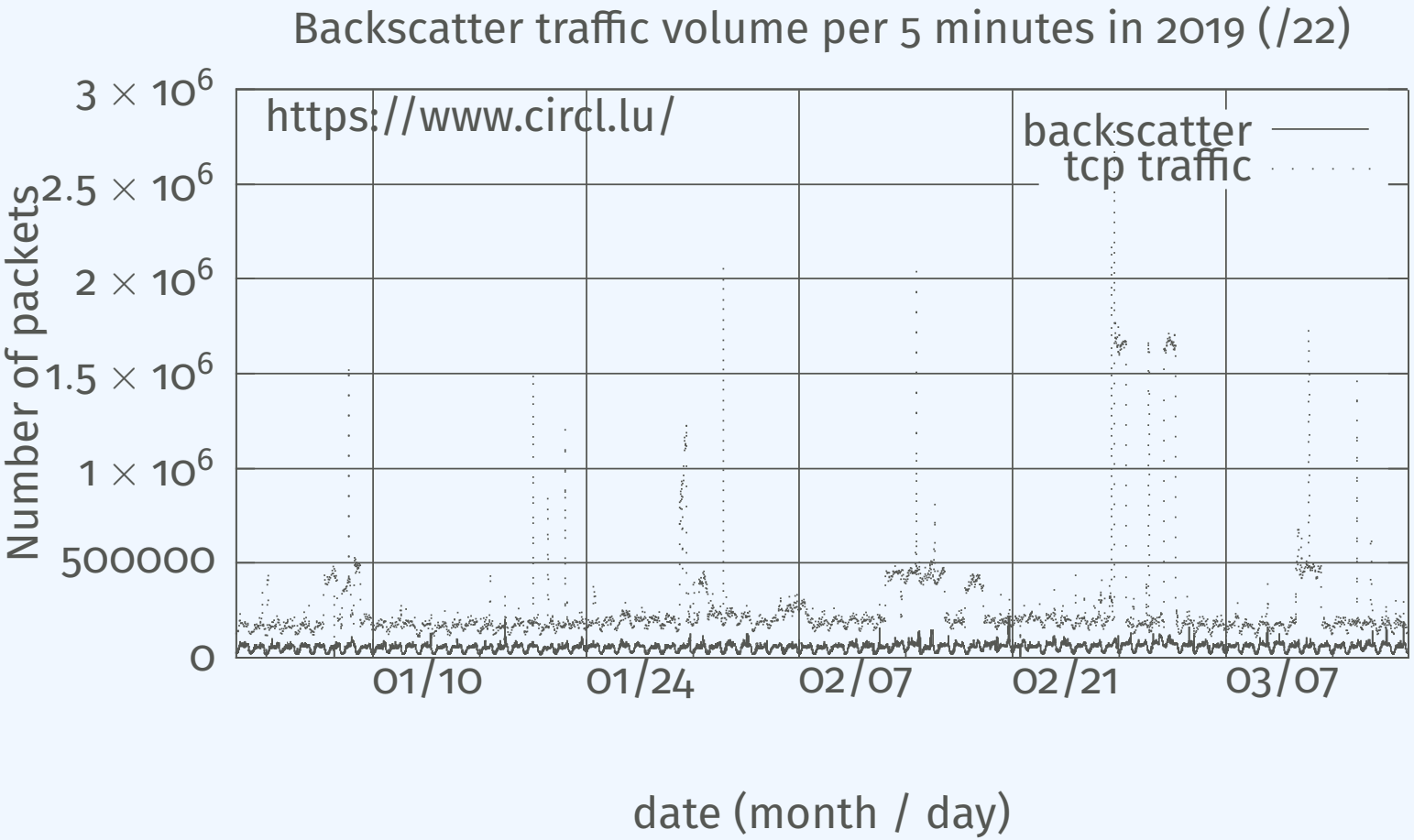
D4 server - sensor management



A distributed Network telescope to observe DDoS attacks

Motivation

DDoS Attacks produce an observable side-effect:



What can be derived from backscatter traffic?

- External point of view on ongoing Denial of Service attacks:
  - **Confirm** if there is a DDoS attack
  - **Recover** time line of attacked targets
  - **Confirm** which services (DNS, webserver, . . .)
  - **Observe** Infrastructure changes
- **Assess the state of an infrastructure under denial of service attack**
  - **Detect** failure/addition of intermediate network equipments, firewalls, proxy servers etc
  - **Detect** DDoS mitigation devices
- **Create** models of DoS/DDoS attacks

D4 in this setting

D4 - for data collection and processing:

- **provide** various points of observation in non contiguous address space,
- **aggregate** and **mix** backscatter traffic collected from D4 sensors,
- **perform** analysis on big amount of data.

D4 - from a end-user perspective:

- **provide** backscatter analysis results,
- **provide** daily updates,
- **provide** additional relevant (or pivotal) information (DNS, BGP, etc.),
- **provide** an API and search capabilities.

First release

- ✓ analyzer-d4-pibs[5], an analyzer for a D4 network sensor:
  - **processes** data produced by D4 sensors (pcaps),
  - **displays** potential backscatter traffic on standard output,
  - **focuses** on TCP SYN flood in this first release.

[5] https://github.com/D4-project/analyzer-d4-pibs

Passive DNS

Problem statement

- CIRCL (and other CSIRTs) have their own passive DNS[6] collection mechanisms
- Current **collection models** are affected with DoH[7] and centralised DNS services
- DNS answers collection is a tedious process
- **Sharing Passive DNS stream** between organisation is challenging due to privacy

[6] https://www.circl.lu/services/passive-dns/
[7] DNS over HTTPS

Potential Strategy

- Improve **Passive DNS collection diversity** by being closer to the source and limit impact of DoH (e.g. at the OS resolver level)
- Increasing diversity and **mixing models** before sharing/storing Passive DNS records
- Simplify process and tools to install for **Passive DNS collection by relying on D4 sensors** instead of custom mechanisms
- Provide a distributed infrastructure for mixing streams and filtering out the sharing to the validated partners

First release

- ✓ analyzer-d4-passivedns[8], an analyzer for a D4 network sensor:
  - **processes** data produced by D4 sensors (in passivedns CSV format[9]),
  - **ingests** these into a **Passive DNS server** which can be queried later to search for the Passive DNS records,
  - **provides** a lookup server (using on redis-compatible backend) that is a Passive DNS REST server compliant to the Common Output Format[10].

[8] https://github.com/D4-project/analyzer-d4-passivedns
[9] https://github.com/gamelinux/passivedns
[10] https://tools.ietf.org/html/draft-dulaunoy-dnsop-passive-dns-cof-04

Passive SSL revamping

A passive SSL fingerprinter

CSIRT's rationale for collecting TLS handshakes:

- **pivot** on additional data points,
- **find** owners of IP addresses,
- **detect** usage of CIDR blocks,
- **detect** vulnerable systems,
- **detect** compromised services,
- **detect** key material reuse,
- **detect** weak keys.

Objectives - TLS Fingerprinting

Keeping a log of links between:

- x509 certificates,
- ports,
- IP address,
- client (ja3),
- server (ja3s),

> *"JA3 is a method for creating SSL/TLS client fingerprints that should be easy to produce on any platform and can be easily shared for threat intelligence."*[11]

[11] https://github.com/salesforce/ja3

Objectives - Mind your Ps and Qs

Collect and **store** x509 certificates and TLS sessions:

- Public keys type and size,
- moduli and exponents,
- curves parameters.

Detect anti patterns in crypto:

- Shared Public Keys,
- Moduli that share one prime factor,
- Moduli that share both prime factor,
- Small factors,
- Nonces reuse / common preffix or suffix, etc.

First release

- ✓ sensor-d4-tls-fingerprinting [12]: **Extracts** and **fingerprints** certificates, and **computes** TLSH fuzzy hash.
- ✓ analyzer-d4-passivessl [13]: **Stores** Certificates / PK details in a PostgreSQL DB.
- lookup-d4-passivessl [14]: **Exposes** the DB through a public REST API.

[12] github.com/D4-project/sensor-d4-tls-fingerprinting
[13] github.com/D4-project/analyzer-d4-passivessl
[14] github.com/D4-project/lookup-d4-passivessl

Future

- **Sensitive information sanitization** by specialized analyzers
- **Previewing datasets** collected in D4 sensor network and providing **open data stream** (if contributor agrees to share under specific conditions)
- **Leverage MISP sharing communities** to augment Threat Intelligence, and provide accurate metrology.

Use it

- **Manage** your own sensors and servers, **find** shameful bugs and **fill** in github issues
- Even better, **send** Pull Requests!
- **Share** data to public servers to improve the datasets (and detection, response, etc.)
- **Feed** your MISP instances with D4's findings - **Share** yours
- **Leech** data, **write** your own analyzers, **do** research

Get in touch if you want to join the project, host a sensor or contribute

- Collaboration can include research partnership, sharing of collected streams or improving the software.
- Contact: info@circl.lu
- `https://github.com/D4-Project`
- `https://twitter.com/d4_project`
- `https://d4-project.org`
  - Passive DNS tutorial
  - Data sharing tutorial