

D4 Project

Open and collaborative network monitoring

Team CIRCL

<https://www.d4-project.org/>

20190207



Alexandre Dulaunoy - Sami Mokaddem

- 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

- Based on our experience with MISP¹ 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>

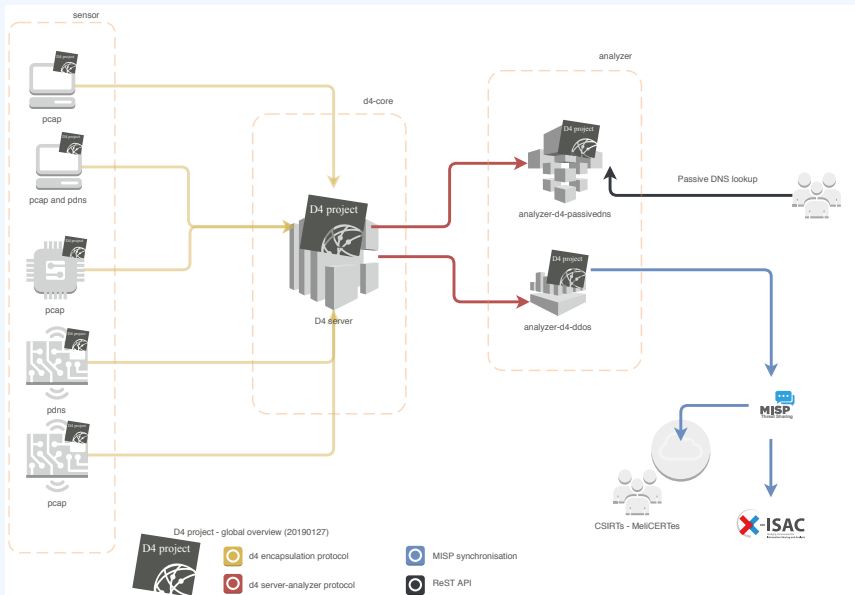
(SHORT) HISTORY

- D4 Project (co-funded under INEA CEF EU program) started - 1st November 2018
- D4 encapsulation protocol version 1 published - 1st December 2018
- vo.1 release of the D4 core² including a server and simple D4 C client - 21st January 2018
- First version of a golang D4 client³ running on ARM, MIPS, PPC and x86 - January 2018

²<https://www.github.com/D4-project/d4-core>

³<https://www.github.com/D4-project/d4-goclient/>

D4 OVERVIEW



ROADMAP (NEXT 2 MONTHS)

- Passive DNS analyzer (alpha version released)
- Passive SSL collector and analyzer
- Backscatter DDoS traffic analyzer
- **Default server** (blackhole monitoring or Passive DNS collector) at CIRCL for organisations willing to contribute without running their own D4 server

D4 ENCAPSULATION PROTOCOL

stream of information
(text or binary)

```
010111010100
100010101101
010100101011
010100100100
011010100101

01011101010010
10001010110101
01010010101111
01010010010100
01101010010101

01011101010010
10001010110101
01010010101111
01010010010100
01101010010101
```



D4 encapsulation protocol version 1



version (8) - Version of the header
type (8) - Data encapsulated type
uuid (128) - Sensor UUID
timestamp (64) - Encapsulation time
hmac (256) - Header authentication
(HMAC-SHA256-128)
size (32) - Payload size



<https://www.d4-project.org>

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

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 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 core server⁴ 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>

Use-case: migrating a legacy network capture model into a D4 network sensor

REMOTE NETWORK CAPTURE

CIRCL operated honeybot for multiple years using a simple model of remote network capture.

Definition (Principle)

- KISS (Keep it simple stupid) - Unix-like
- Linux & OpenBSD operating systems

Sensor

```
tcpdump -l -s 65535 -n -i vro -w - '( not port $PORT and not host $HOST )' | socat - OPENSsl  
-CONNECT:$COLLECTOR:$PORT,cert=/etc/openssl/  
client.pem,cafile=/etc/openssl/ca.crt,verify=1
```

Limitations

- Scalability → one port per client
- Identification and registration of the client
- Integrity of the data

Multiplexing streams in D4

- Inspired by the unix command tee
- Read from standard input
- Add the d4 header
- Write it on standard output

USING D4 NATIVE CLIENT

```
tcpdump -n -s0 -w - | ./d4 -c ./conf | socat -  
  OPENSLL-CONNECT:$D4-SERVER-IP-ADDRESS:$PORT,  
  verify=1
```

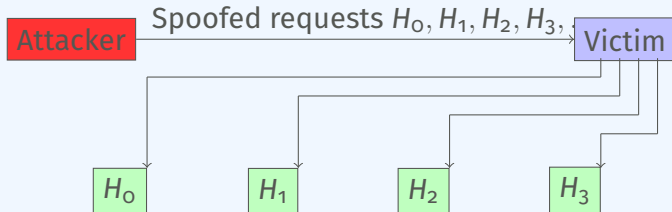
Configuration directory

Parameter	Explanation
type	see D4 Header slide
source	standard input
key	HMAC key
uuid	Identifier of the sensor
version	version of the sensor
destination	standard output
snaplen	length of data being read & written

Use-case: D4 analyzer to detect DDoS attacks in backscatter traffic

OBSERVING SYN FLOODS ATTACKS IN BACKSCATTER TRAFFIC

Attack description



Connections
H_0
H_1
H_2
H_3

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 are a target (DNS, webserver, ...)
- Infrastructure changes or updates
- 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 or services
- Create probabilistic models of denial of service attacks

CONFIRM IF THERE IS/WAS A DDOS ATTACK

Problem

- Distinguish between compromised infrastructure and backscatter
- Look at TCP flags → filter out single SYN flags
- Focus on ACK, SYN/ACK, ...
- Do not limit to SYN/ACK or ACK → ECE (ECN Echo)⁵

```
tshark -n -r capture-20170916110006.cap.gz -T  
fields -e frame.time_epoch -e ip.src -e tcp.  
flags
```

```
1505552542.807286000 x.45.177.71 0x000000010  
1505552547.514922000 x.45.177.71 0x000000010
```

⁵<https://tools.ietf.org/html/rfc3168>

PASSIVE IDENTIFICATION OF BACKSCATTER (WIP)

```
./pibs -b -r pcap_file.cap
```

Early version is available of PIBS⁶ with a focus on TCP traffic.

Options	Explanations
-r	read pcap file
-b	display IPs under DDoS on standard output

Dependencies

libwiretap-dev

libhiredis-dev

libwsutil-dev

⁶<https://github.com/D4-project/analyzer-d4-pibs>