

D4 Project

Open and collaborative network monitoring

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

2019/09/23

TEAM CIRCL



- 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>

- 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 2019**
- First version of a golang D4 client³ running on ARM, MIPS, PPC and x86 - **January 2019**
- First Analyzers - **Spring 2019**
- Client Generator - **Summer 2019**

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

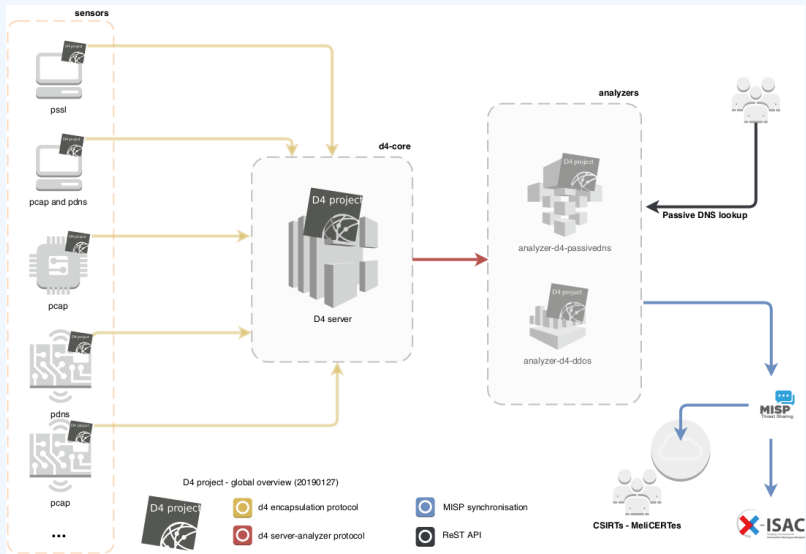
³<https://www.github.com/D4-project/d4-goclient/>

(SHORT) HISTORY

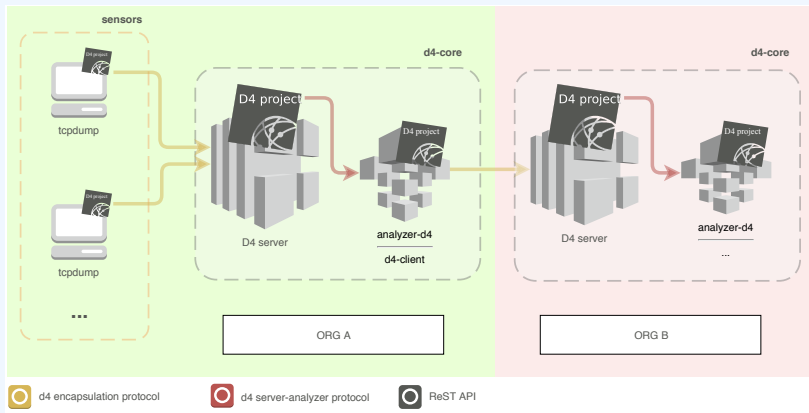
Release	Date
AIL-framework-v1.5	Apr. 26, 2019
...	
AIL-framework-v2.1	Aug. 14, 2019
analyzer-d4-balboa-vo.1	Aug. 19, 2019
analyzer-d4-passivedns-vo.1	Apr. 5, 2019
analyzer-d4-passivessl-o.1	Apr. 25, 2019
analyzer-d4-pibs-vo.1	Apr. 8, 2019
BGP-Ranking-1.0	Apr. 25, 2019
BGP-Ranking-1.1	Aug. 19, 2019
d4-core-vo.1	Jan. 25, 2019
d4-core-vo.2	Feb. 14, 2019
d4-core-vo.3	Apr. 8, 2019
d4-goclient-vo.1	Feb. 14, 2019
d4-goclient-vo.2	Apr. 8, 2019
d4-sensor-generator-vo.1	Aug. 22, 2019
d4-server-packer-o.1	Apr. 25, 2019
IPASN-History-1.0	Apr. 25, 2019
IPASN-History-1.1	Aug. 19, 2019
sensor-d4-tls-fingerprinting-o.1	Apr. 25, 2019

see <https://github.com/D4-Project>

D4 OVERVIEW



D4 OVERVIEW - CONNECTING SENSOR NETWORKS



<https://d4-project.org/2019/06/17/sharing-between-D4-sensors.html>

- Passive DNS collection
- Passive SSL collection
- AIL collection
- Correlations, CTI
- DDoS Detection

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

- ✓ Blackhole DDoS
- ✓ Passive DNS
- ✓ Passive SSL
- Gene⁴ / WHIDS⁵ (sysmon)
- BGP mapping
- egress filtering mapping
- Radio-Spectrum monitoring: 802.11, BLE, GSM, etc.

⁴<https://github.com/oxrawsec/gene>

⁵<https://github.com/oxrawsec/whids>

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 server⁶ 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.

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

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 blacklist
- 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)

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)

- Worker custom type (called Worker 2)
 - ▶ Get type 2 data from a stream
 - ▶ Reconstruct Json
 - ▶ Extract extended type name
 - ▶ Use default type or special extended handler
 - ▶ Save Json on disk
 - ▶ Get type 254 data from a stream
 - ▶ Reconstruct type 254
 - ▶ Save data in Redis. Create a queue for D4 Analyzer(s)

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 blacklist
- Create Analyzer Queues

D4 SERVER - MAIN INTERFACE


The screenshot displays the D4 Server Main Interface. At the top, there is a navigation bar with the D4 project logo and links for Home, Sensors Status, and Server Management. The main content is divided into two panels: 'UUID' and 'Types'. The 'UUID' panel shows a list of five entries, each with a numerical ID and a corresponding UUID string. The 'Types' panel shows two entries, each with a numerical ID and a descriptive type name. The date 2019/05/20 is displayed at the bottom of both panels.


UUID	
4019794	c0bb49e788964718af4dfea4c0ab898c
47820	bbbcf7a43aed47aa84badc50262f5aba
27183	37d2f040fc074aaab2caf49059667525
8401	1b06b4ab8a754ef9ae3dd4d073b38f0e5
1022	de1df62d862b494a830f1f78ec27fca5



Types	
4046981	1: pcap (libpcap 2.4)
57243	8: passivedns CSV stream

2019/05/20

2019/05/20

 CIRCL
Computer Incident
Response Center
Luxembourg

 Co-financed by the Connecting Europe
Facility of the European Union

D4 SERVER - SERVER MANAGEMENT

The screenshot displays the 'Server Management' section of the D4 Server interface. It is divided into two main columns: 'Blacklist IP' and 'Blacklist UUID'.

Blacklist IP Section:

- Blacklist IP:** A form with an 'IP Address' input field and a 'Blacklist IP' button.
- Manage IP Blacklist:** A button labeled 'Show Blacklisted IP'.
- Unblacklist IP:** A form with an 'IP Address' input field and an 'Unblacklist IP' button.

Blacklist UUID Section:

- Blacklist UUID:** A form with a 'UUID' input field and a 'Blacklist UUID' button.
- Manage UUID Blacklist:** A button labeled 'Show Blacklisted UUID'.
- Unblacklist UUID:** A form with a 'UUID' input field and an 'Unblacklist UUID' button.

Header Accepted Types Section:

This section contains two tables. The top table is titled 'Header Accepted Types' and shows a list of types with their descriptions and a 'Remove Type' button for each.

Type	Description	Remove Type
1	pcap (libcap 2.4)	Remove Type
2	meta header (JSON)	Remove Type
4	dnscap output	Remove Type
8	passivedns CSV stream	Remove Type
254	type defined by meta header (type2)	Remove Type

Navigation: Showing 1 to 5 of 5 entries. Previous 1 Next

Add New Types: A form with a text input containing '1' and an 'Add New Type' button.

The bottom table is titled 'Extended Types' and shows a single entry:

Type Name	Description	Remove Type
ja3-f		Remove Extended Type

Navigation: Showing 1 to 1 of 1 entries. Previous 1 Next

D4 SERVER - SERVER MANAGEMENT

Analyzer Management

Show 10 ▾ entries Search

Type	uuid	last updated	Change max size limit	Analyzer Queue
1	f72ea760-370b-4f99-bb93-b6c6e645a32	2019-05-20 14:14:23	10000 <input type="text"/> Change Max Size	<input type="text" value="10001"/>
8	4072e072-bfaa-4395-9bb1-ccb3b470d715	2019-05-20 14:14:57	10000 <input type="text"/> Change Max Size	<input type="text" value="0"/>

Showing 1 to 2 of 2 entries Previous 1 Next

Show 10 ▾ entries Search

Type Name	uuid	last updated	Change max size limit	Analyzer Queue
jdk-f	8d8b724c71b64d6c942bffc2b6d761ac <small>This analyzer pushes TLS sessions into a postgres database for passiveSSL.</small>	2019-05-14 08:50:31	100000 <input type="text"/> Change Max Size	<input type="text" value="18036"/>

Showing 1 to 1 of 1 entries Previous 1 Next

Add New Analyzer Queue

1

Optional Description

D4 SERVER - SENSOR OVERVIEW

201905 Home Sensors Status Server Management

Active Connection

UID: 04180306237445207796037604		
First Seen	Last Seen	Status
2019-09-01 13:05:05 (-150408190)	2019-09-20 13:56:23 (-15080056)	OK Connected

UID: 10060606076476060607000604		
First Seen	Last Seen	Status
2019-04-08 12:27:42 (-15047040)	2019-05-20 14:19:08 (-15080194)	OK Connected

UID: 010200001000000000000000		
First Seen	Last Seen	Status
2019-04-01 11:46:31 (-15041191)	2019-05-20 14:17:35 (-15080187)	OK Connected

UID: 000070000000000000000000		
First Seen	Last Seen	Status
2019-04-02 07:16:49 (-15040940)	2019-05-20 14:17:35 (-15080187)	OK Connected

UID: 000000000000000000000000		
First Seen	Last Seen	Status
2019-04-08 10:06:12 (-15042092)	2019-05-20 14:17:35 (-15080187)	OK Connected

D4 SERVER - SENSOR MANAGEMENT

D4 project



[Home](#)

[Sensors Status](#)

[Server Management](#)

UUID: de1df62d862b494a830f178ec27fca5

First Seen	Last Seen	Status
2019-03-31 11:03:05 - (1554030185)	2019-05-20 13:56:23 - (1558360583)	OK <input checked="" type="checkbox"/> Connected Kick UUID

Change Stream Max Size

10000

[Change Max Size](#)

UUID Blacklist

[Blacklist UUID](#)

Blacklist IP Using This UUID

[Blacklist IP](#)

Change UUID Key

private key to change

[Change UUID Key](#)

Types Used:

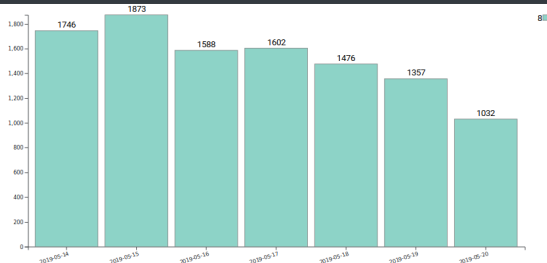
Show 10 entries

Search:

Type	first seen	last seen
8	2019-04-04 12:46:43	2019-05-20 13:56:23

Showing 1 to 1 of 1 entries

[Previous](#) [1](#) [Next](#)



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 vrr0 -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

Encapsulating 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 -  
  OPENSsl-CONNECT:$D4-SERVER-IP-ADDRESS:$PORT,  
  verify=1
```

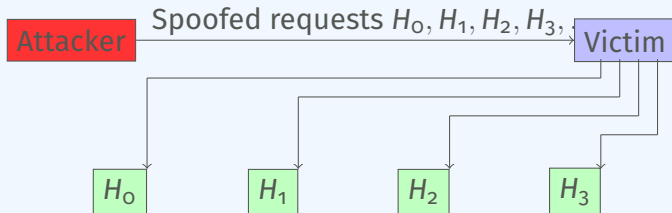
Configuration directory

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)⁷

```
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
```

⁷<https://tools.ietf.org/html/rfc3168>


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

Early version is available of PIBS⁸ 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

⁸<https://github.com/D4-project/analyzer-d4-pibs>

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