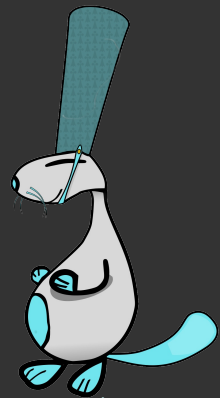


The Future of Protocol Reversing and Simulation Applied on ZeroAccess

29C3, Hamburg

December 29 2012

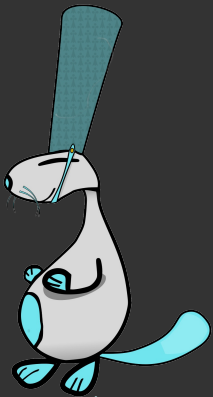
@Netzob



Authors...

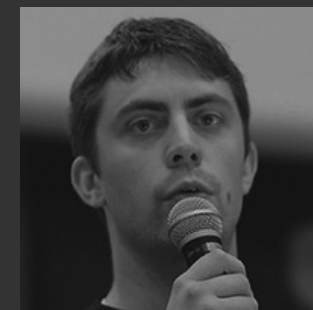
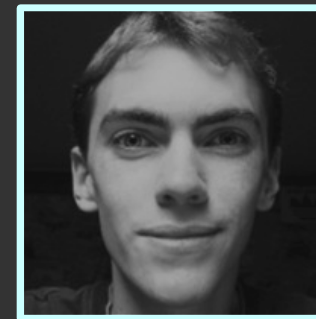
« You talkin' to me? »

Travis Bickle





Yes, we're French !



Frédéric GUIHÉRY (@_sygus)

- IT security engineer
 - Reverse engineering
 - System analysis and hardening
 - Trusted Computing



Georges BOSSERT (@Lapeluche)

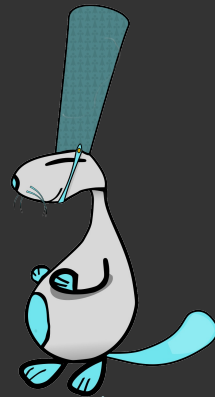
- ▶ PhD student
 - ▶ Intrusion Detection
 - ▶ Botnet simulation
 - ▶ Protocol learning

Supelec CIDre
Research team



Advisers :

- ▶ *Guillaume Hiet*
- ▶ *Ludovic Mé*

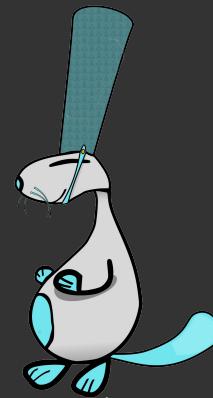


netzob.org



AMOSSYS, France

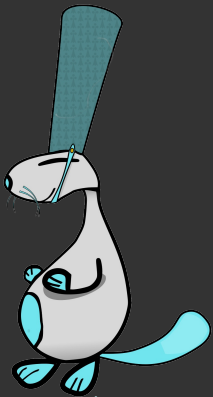
- ▶ Audit and evaluation
 - ▶ ITSEF lab (Common Criteria, CSPNs, ...)
 - ▶ Pentest lab
- ▶ R&D

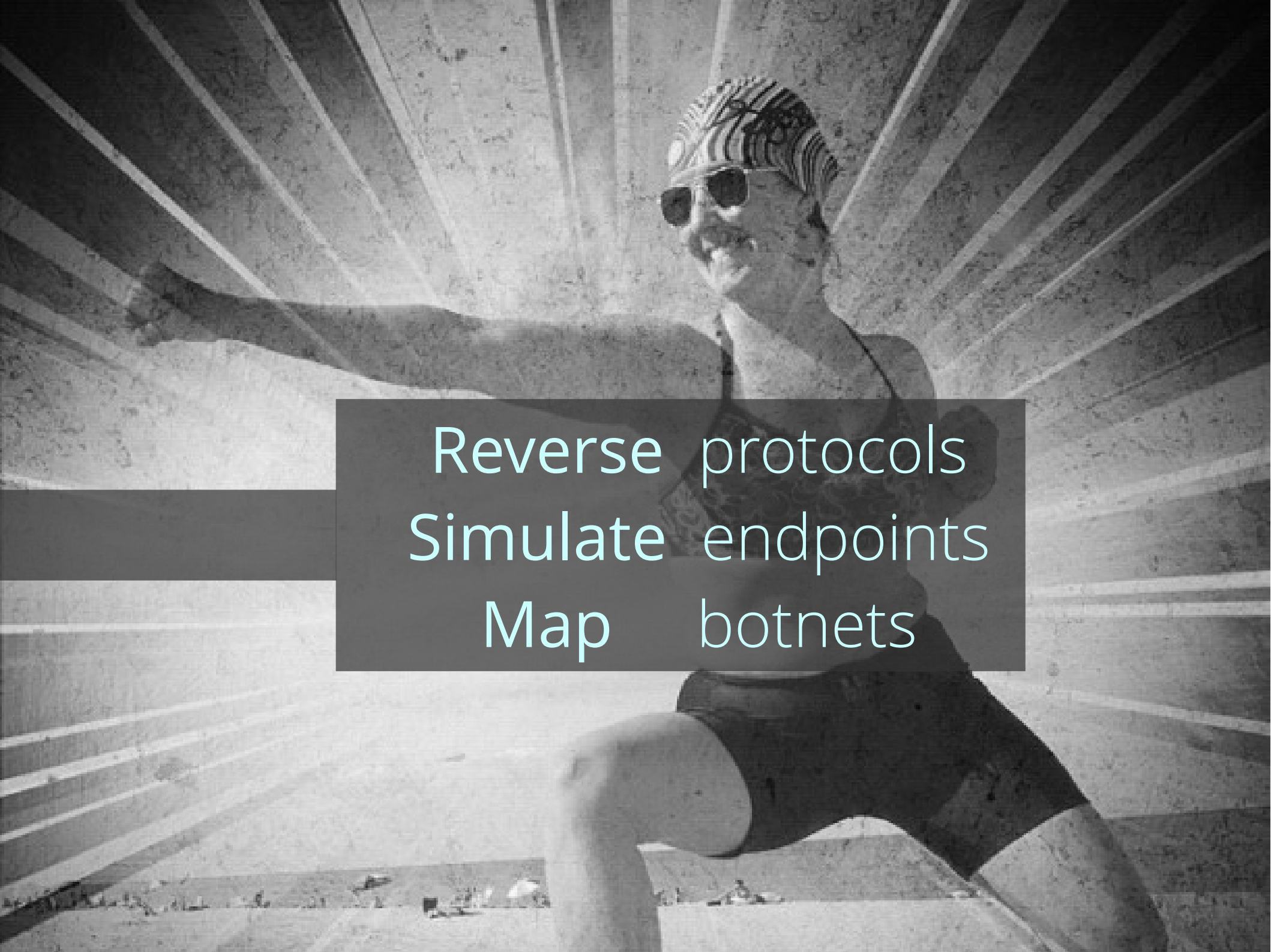


Topics...

« Go ahead, make my day »

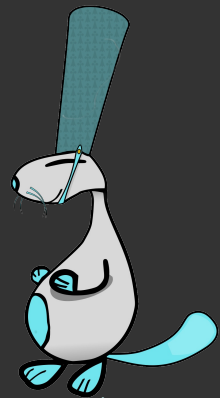
Harry Callahan

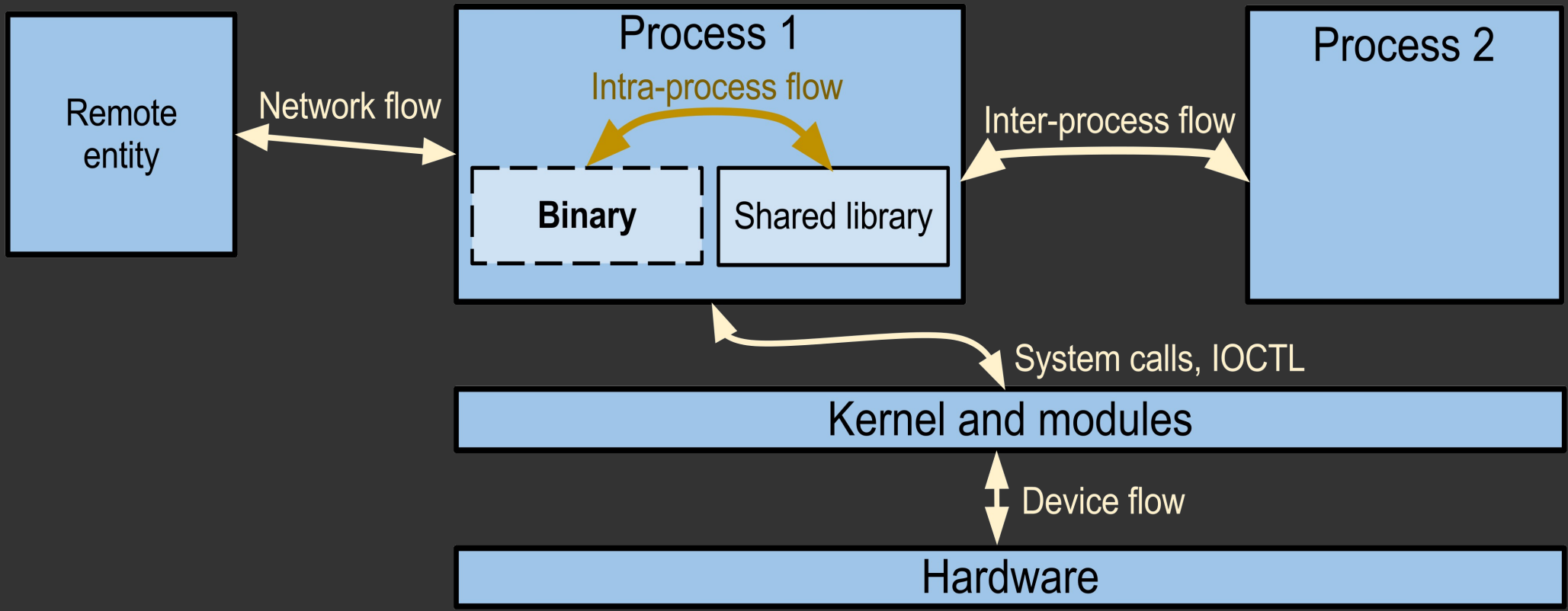


A black and white photograph of a woman in athletic wear running on a track. She is wearing a patterned swim cap, sunglasses, a dark tank top, and shorts. Her right arm is extended forward. The background shows the curved lanes of a running track. A semi-transparent dark grey rectangular box is overlaid in the center of the image, containing white text.

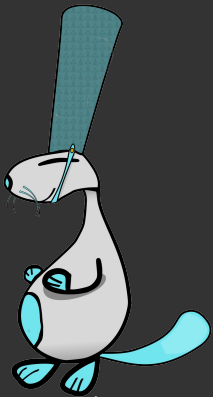
Reverse protocols
Simulate endpoints
Map botnets

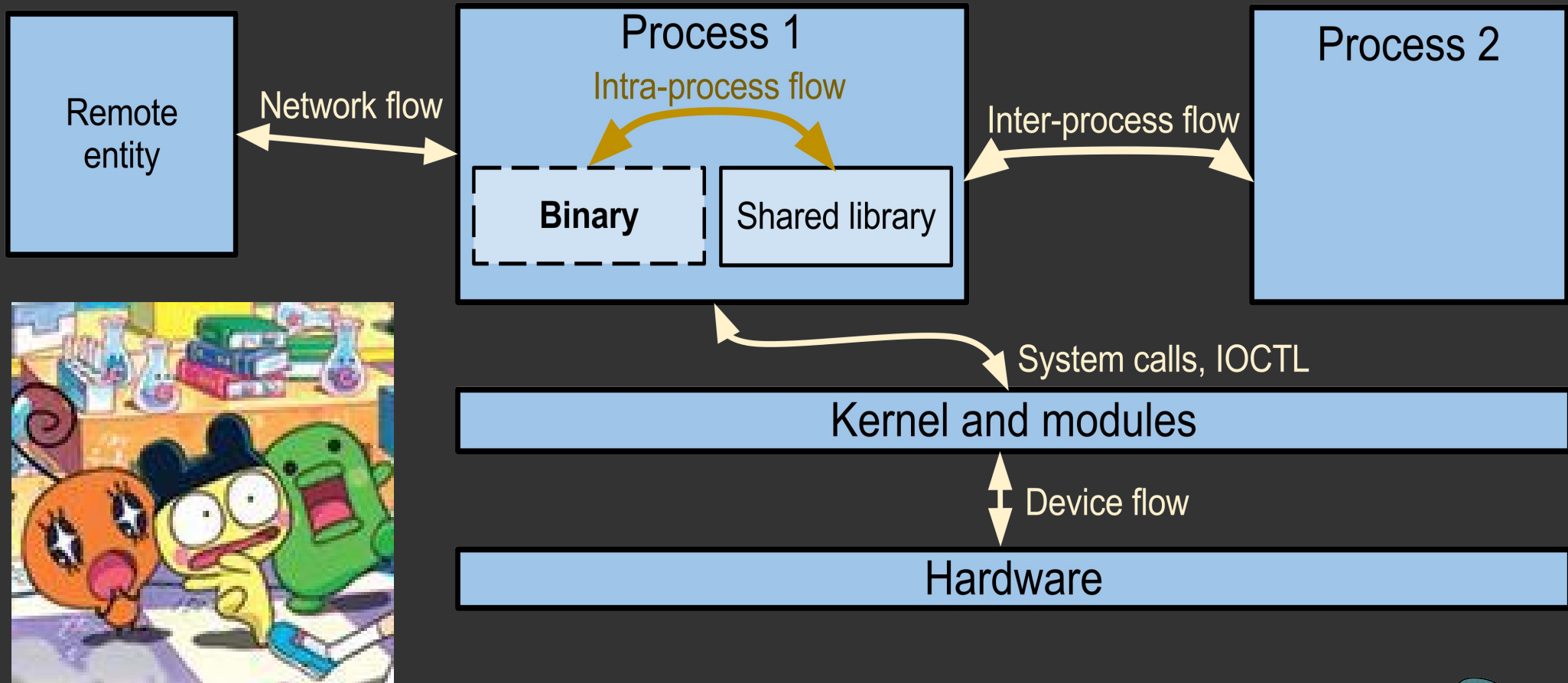
Why reverse engineering of protocols ?





Protocols are everywhere

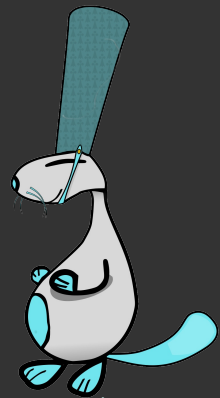




Protocols are everywhere
 (even within Tamagotchis)

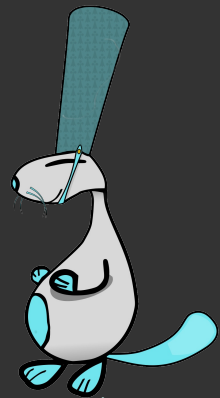
Assess the robustness of implementations

- ▶ **Ex** : Fuzz the control API of a centrifuge
- ▶ 29C3 Ex :
 - « Many Tamagotchis Were Harmed in the Making of this Presentation »
 - « EXSi Beast »



Assess the robustness of implementations

- ▶ Ex : Fuzz the control API of a centrifuge
- ▶ **29C3 Ex :**
 - « Many Tamagotchis Were Harmed in the Making of this Presentation »
 - « EXSi Beast »



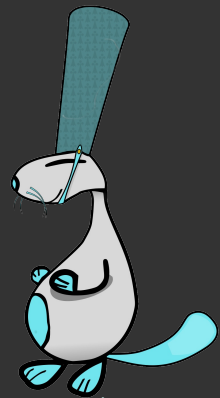
Analyze traffic and identify potential data leakage

- ▶ **Ex :** Are you sure your « IP Reputation Appliance » doesn't leak your emails ?



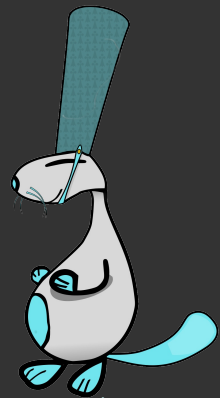
Compare the implementation of a protocol
with its official specifications

- ▶ **Ex** : CC evaluations of crypto products

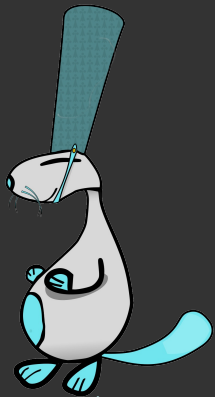


To develop a free version of a proprietary implementation

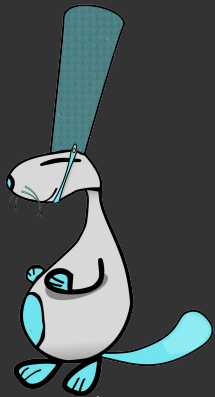
- ▶ **Ex** : Drew Fisher's talk @ 28C3 on Kinect RE



Current reverse engineering approach...

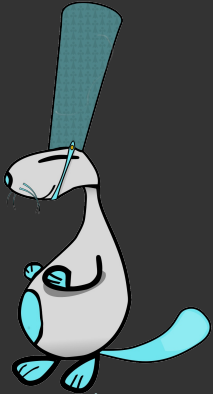


Have you ever (tried to) RE a protocol ?



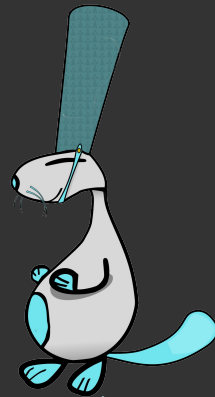
Did it looked like this ?

```
4f945a3...427c99812598f17...9e0bb1c2722165b3497f4c607deb34efb5ef1878aea36bce6ae7d9be
89546ef16a59c959f592ea5297385abf71c297a...8f66ff7ca8f11e7f23b441fb4a88bc2b851c14c...b
a2078fcd3b19c25348399bec164aad64f84f6868d4495bd50d332340276904ef00620800f7ffa2f2
70c5c3c66a3edde50b8347fae58ed2bcb2d287401c86c4e1600e9ff...3956ce847ea3138ef800...35272
09416f06d6c3b42a4fa7dad6c2aa7f0c3da7caf5d2311dd6d0e8640c298b3d6bd613400cbff19589b
c1200ec...49355a7d4ab3e40859d76878af9391c6014700b8ff3ed10cf93d2565eb53c1c8728ac0
b42f37be5199f4aa33400cbff1188df51ce08ce8e78e...b604...c1907678b16f1f1b66ed3023929acb2
4ccb894ea3d548f030eba364100beff776f888bec5edb91448b780cd1d382c5a95a0411acd8850927
f5f4b9eb3bf83ff10656f2b734e9f3340f75ba9a62db0f09707adf62feeda7b887e0d66311e46a9f
479a2ca40c82a61a948cb4346b891d7888d39e057b521ccb5a759888bb568d2bbe4a406e41fc3028
32af01d017e2a8ed00c7501b75b9c0c3a...509530...0cfff49015a7447fac2cfba6b836d8e7dbe2497
bbbb0df3b088703f8e8984200bdf9359b4c09979744fff393639023b62190cecac64aeede5d757ac
bffdff7db1004effbc99859183ba69a30f4749400ab2401de222723...cb77db211c99d23d5778abe...f9c
16a58d51d5ba10377d469ff8ae1e0ff8c6afad206c20110ca24f171475905acb3...3180...ad2bad258d7
119488f8ade92c00d3fff508ca9cec38...7776fecc000a77e7e1140062e474ab25fa80c39fe14882
0a691e8a3fe3c9423d986427ba81200edff842d1c115254e573d4a2730927d3e4d03e693800ccffdd
4313000cfff872cc756b40aa3e01490ebcea54cf3def3fc4f91edb5674840d9446204b5667613dad9
4178d0aade5039d0fe3d4662dfbf1043c642d6d34581337eedb736287d9bd35249911a426914e038d
a78b3fc0cca831f8f57bbb4f0dc216bb67c354149ad639add4bd026cefa0110cefa02101800e7ffcc
ebb3990a8ed5e9daf6ff6d1c01663c0f305cld39110098a0b...ca3c5e3495c5b5b10eb0...9582339d88
ff46e530e92b21bf1fe07e6ed1fe942f0a177a1ba802ab20527317dd132c40220090df7ff8deac3
ee58d858d6dae222d1dd32f00d0ff3127e2003add173565570f127c3f00760ccea7ba57a680750832
79aec8bad48b3929e603c2731480321869fec0df02d45c3a95d3f036c356c984ee031992cd2d53e50
d668979894b8...a61b00e4ff523f8b...e7bb03d57f99445edf68846ff5504454f8774581955b67243f00
7385557232c7f492ead9f9fdbecd27f34...200cdf02d5422ffa685f3ec0452554f6c403df9d9...39655
333f3e210446d6b43674100beff256f0...71e7047ba8af6d2a131f587baf61b9e286f0d5c6c24...579a3
01fa5b7c5f5469b7af29e972ccb4800b7...fadf844279d51bdd46df1c5b1e84ee78d1d75a54d2...df372
41800e7ff4871180f123aef0a820fd9805556c4495106b0f6370b5adc1b00e4ffa32993f0c8743f8a
aeacf90a0d7cdca9e9c0a56e6176cbf1bd402d913cceb2216c6b1d97321e19e93368df6e7706728d
888a9c9f56f879976a0bd34c00a00f5ffc05ef8f1118fe604c0800f7ffcfeacacbdb8a02621800
9e8b0c038aad085d41100eeff233a767ce75b801493af803e57a76683e31400ebff8baed3dfedc446
```



Did it looked like this ?

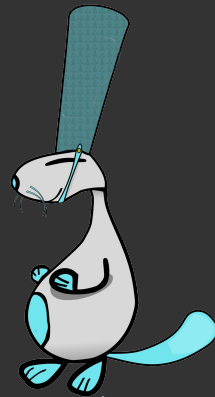
- ▶ Complex
 - ▶ Time-consuming
- ▶ Mostly Manual




Did it looked like this ?

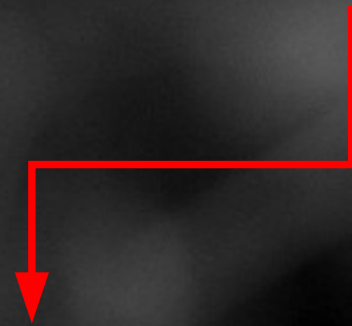
- ▶ Complex
 - ▶ Time-consuming
- ▶ Mostly Manual

MOSTLY VISUAL



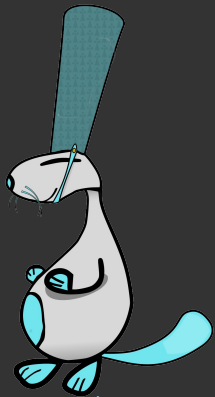


No available tool to **reverse**
a **proprietary** protocol...



Should we create one?

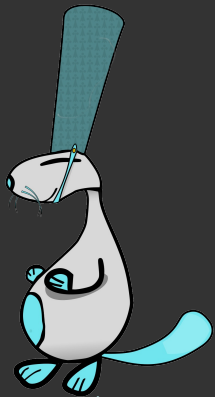
Let's see if we can **automate**
some RE tasks...



Some reminders about **protocols**

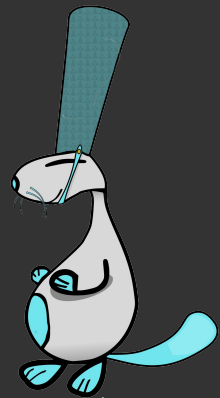
SECURITY
WILL BE
RIGHT BACK

Let's examine the TCP protocol



Different **types** of messages

- ▶ SYN message
- ▶ ACK message
- ▶ PUSH message
- ▶ FIN message
- ▶ RST message
- ▶ ...



Concept of encapsulation layers

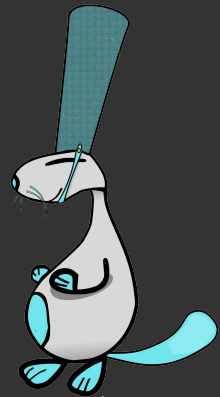
Ethernet

IPv4

TCP

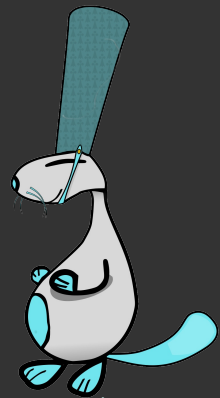
HTTP

HTTP Body

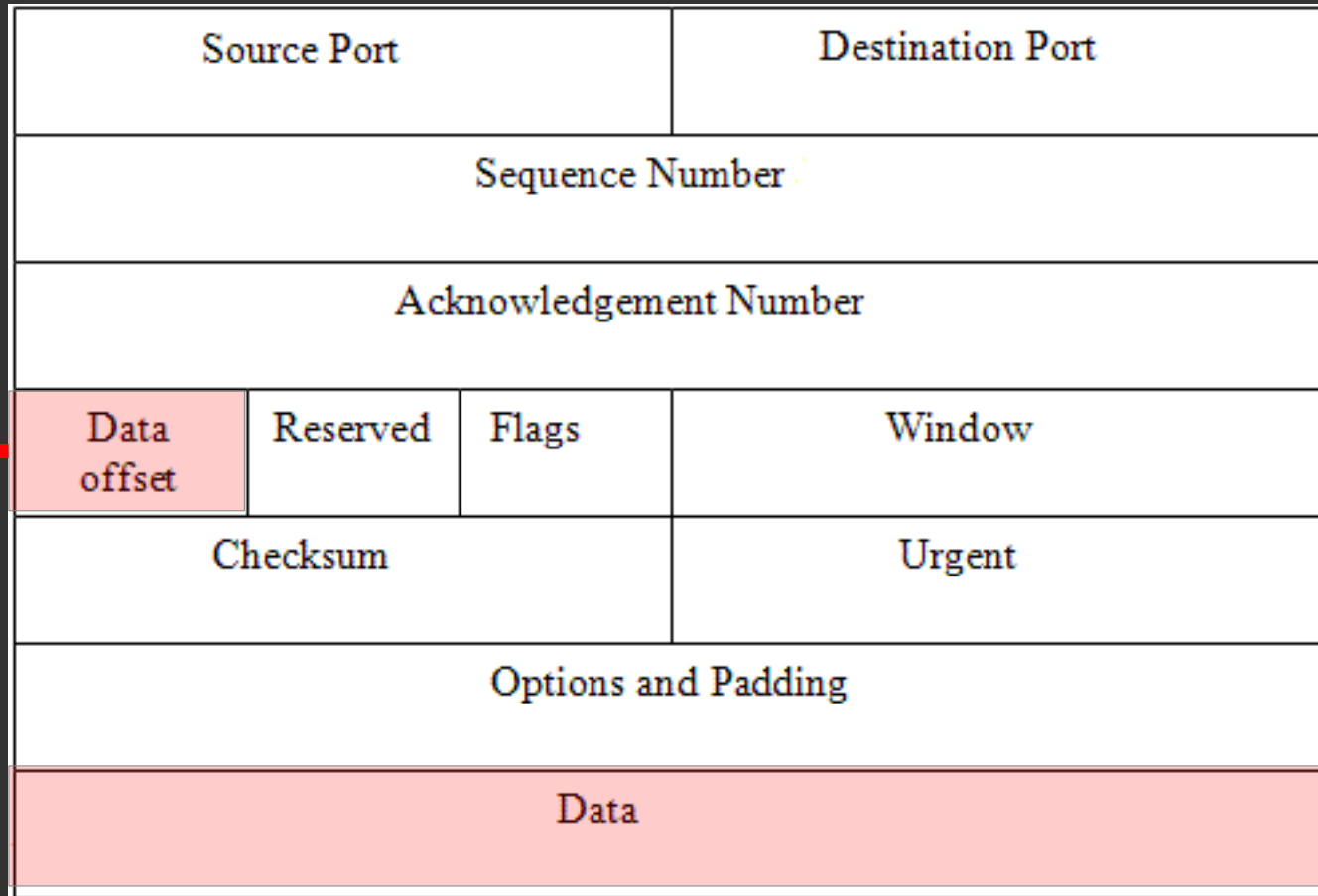


Fields partitioning

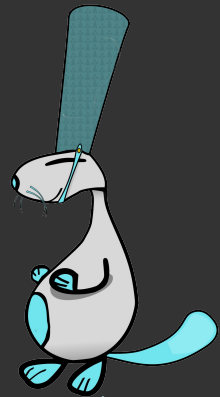
Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
Data offset	Reserved	Flags	Window
Checksum		Urgent	
Options and Padding			
Data			



Relations



Intra-message relations



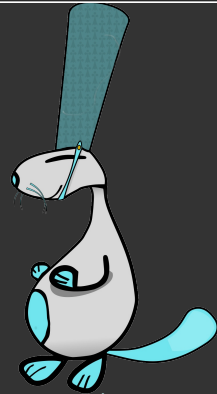
Relations

Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
Data offset	Reserved	Flags	Window
Checksum		Urgent	
Options and Padding			
Data			



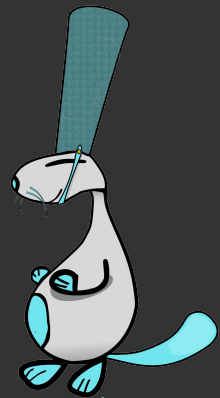
Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
Data offset	Reserved	Flags	Window
Checksum		Urgent	
Options and Padding			
Data			

Inter-message relations



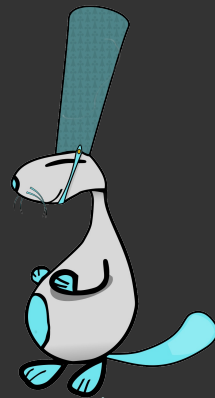
Contextual values

Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
Data offset	Reserved	Flags	Window
Checksum		Urgent	
Options and Padding			
Data			

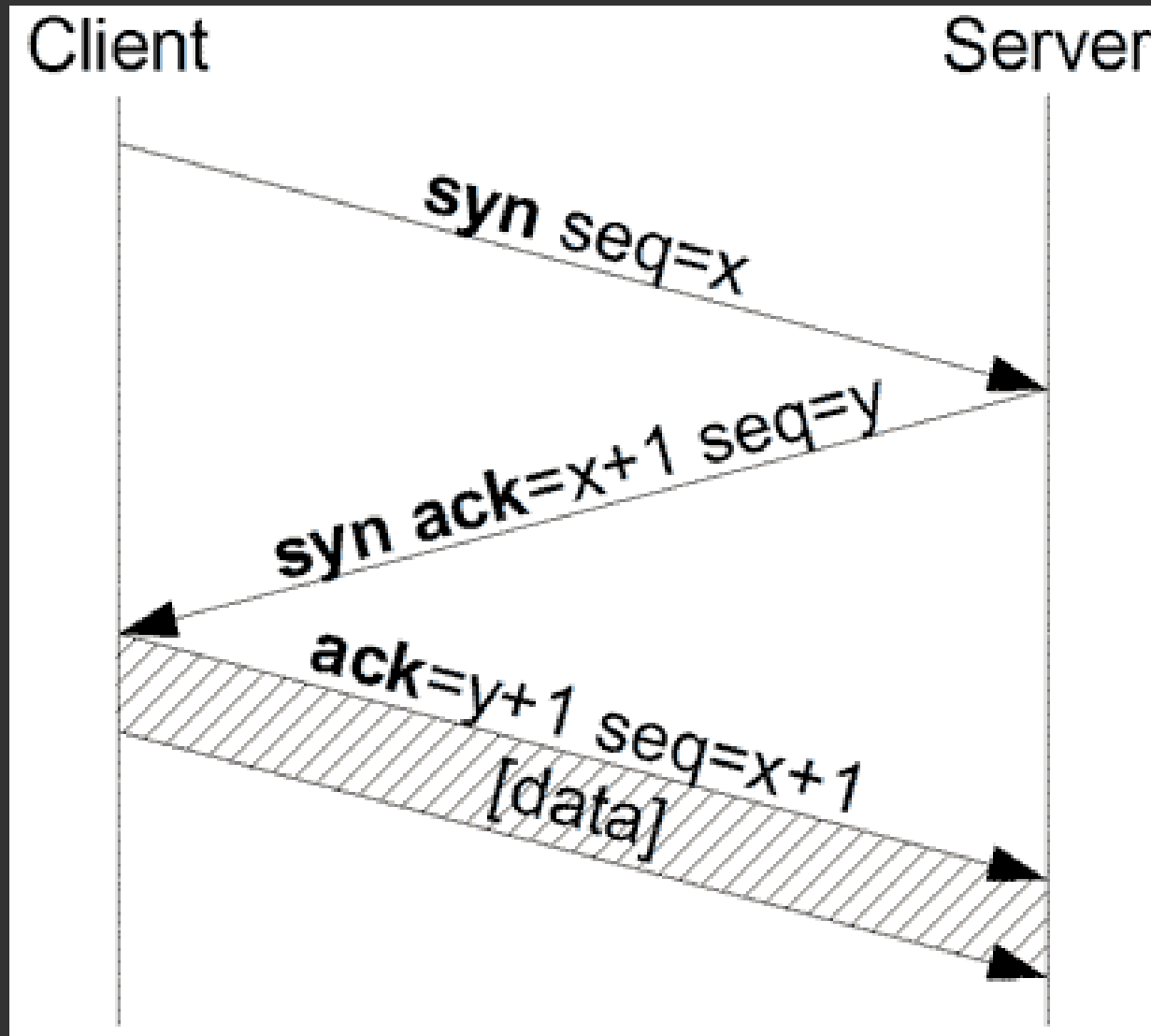


Application-level values

Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
Data offset	Reserved	Flags	Window
Checksum		Urgent	
Options and Padding			
Data			



Sequence of valid messages

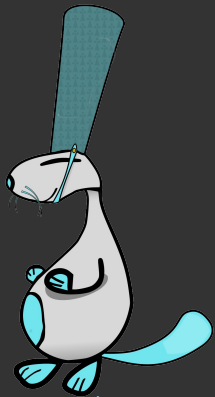




Let's find a **model** that covers protocol attributes

Academics are very good with * models :)

* *sometimes useless*



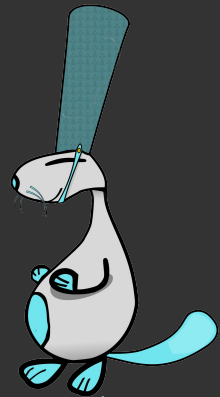
netzob.org



« *Design and Validation of Computer Protocols* »
by G. Holzmann

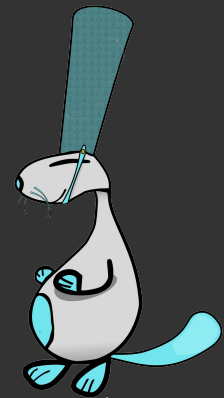


A Communication Protocol is made of
5 distinct parts ...

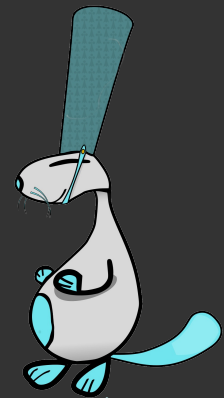


netzob.org

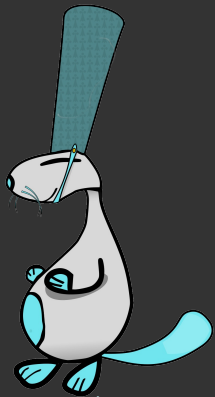
a service (1/5)



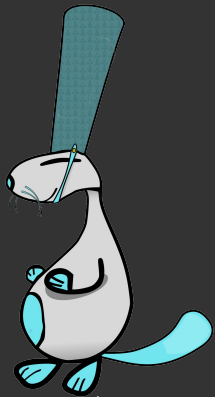
a service (1/5)



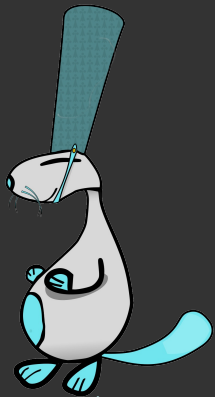
some **assumptions** about the
environment (2/5)



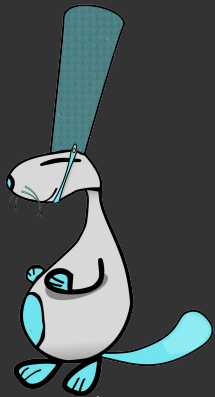
a vocabulary of messages (3/5)



the **encoding** (format) of each message
(4/5)



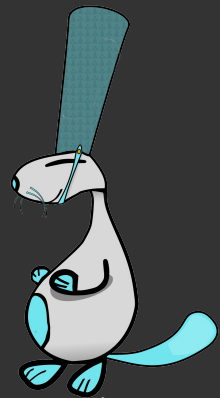
the procedure rules (5/5)



Yes, that was an academic model

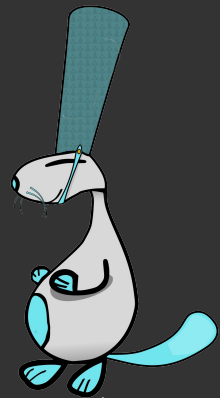


SMHP'S



Reduced model for a Protocol

- ▶ a vocabulary → a list of **Message Format**
- ▶ a grammar → **State Machine**





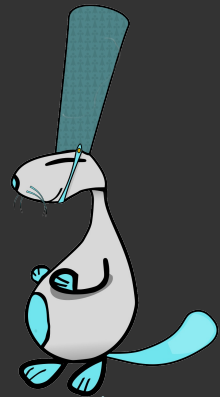
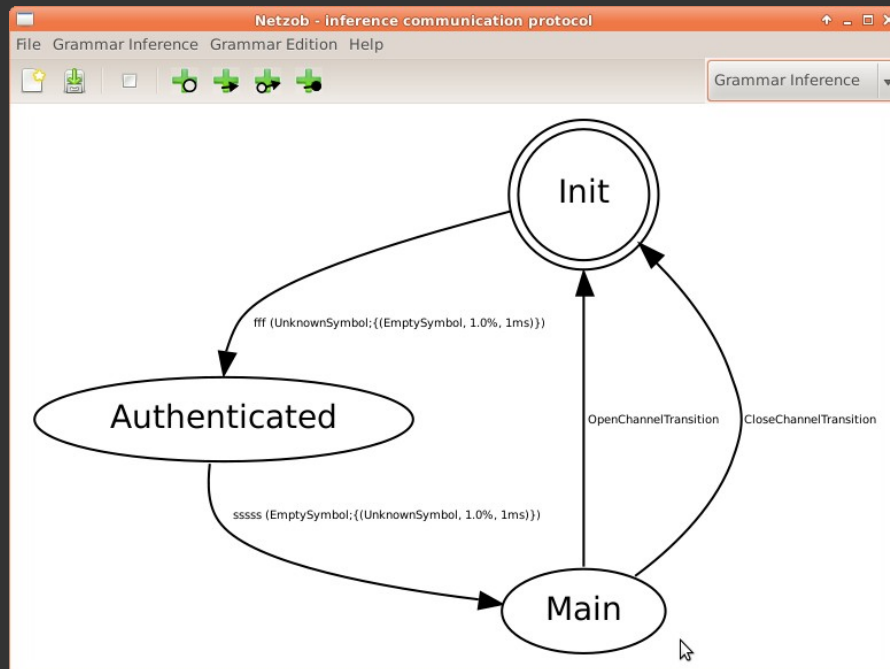
Introducing Netzob ...

Goals of Netzob

- ▶ Infer unknown (proprietary) protocols

Simulate actors of a communication

Smart-Fuzz targeted implementations



Goals of Netzob

Infer unknown (proprietary) protocols

- ▶ Simulate actors of a communication

Smart-Fuzz targeted implementations

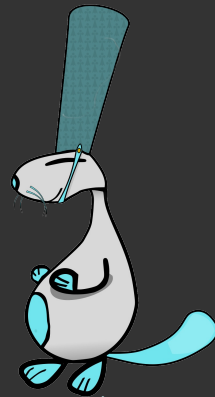
The screenshot displays the Netzob application window titled "Netzob - inference communication protocol". The interface is divided into several sections:

- Actors:** A table with columns "Status" and "Name". One actor named "client" is listed with a red status icon.
- Properties:** A table with columns "Property" and "Value".

Property	Value
ID	73b8bef6-b107-4093-9649-b643e6ead8fd
Name	client
Initiator	Yes
Protocol	UDP
Bind IP	127.0.0.1
Bind Port	10000
Target IP	127.0.0.1
Target Port	4242
- Grammar:** A state transition diagram with three states: "aaa", "a", and "sa". Transitions are labeled with symbols like "a" and "sa".
- Communication Channel:** A table with columns "Time", "Symbol", and "Message".

Time	Symbol	Message
17:18:22	Symbol 3	0x63653d2d2069643d226d6f7665
- Memory Accesses:** A table with columns "OP", "Time", "Variable", and "Data".

Buttons for "Propriétés", "Supprimer", and "Arrêter" are visible below the properties table. A "Simulateur" dropdown menu and "Arrêter" and "Lire" buttons are located in the top right corner.

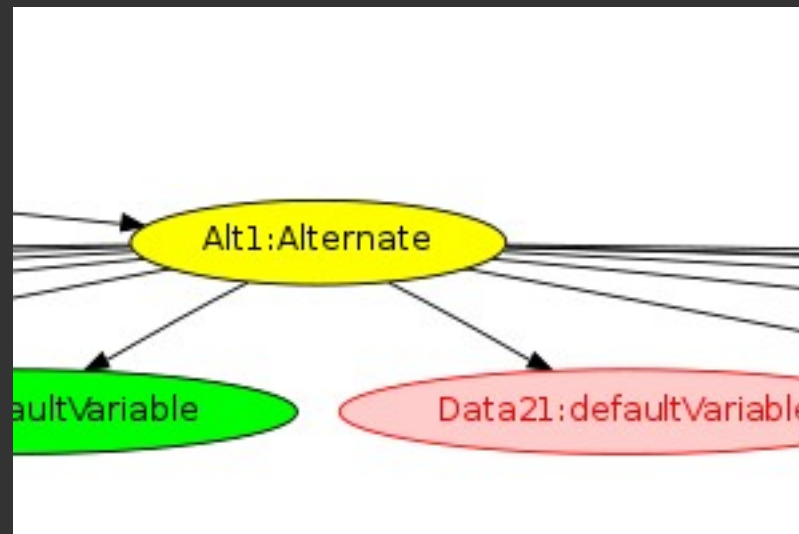


Goals of Netzob

Infer unknown (proprietary) protocols

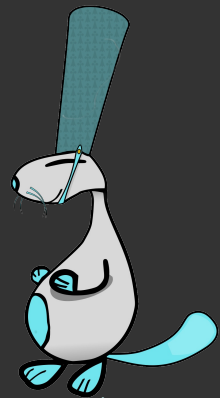
Simulate actors of a communication

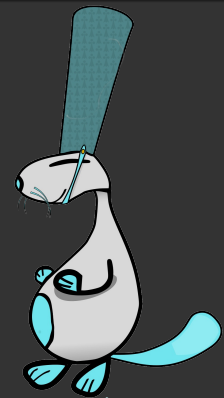
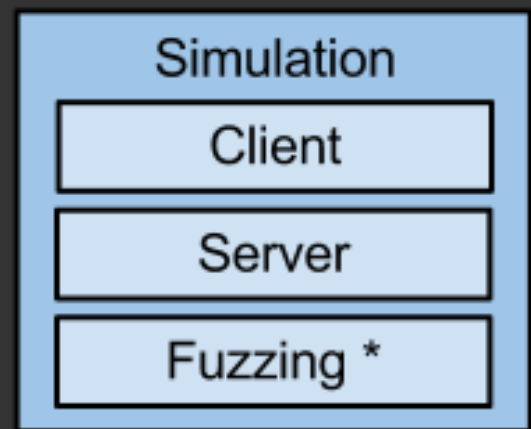
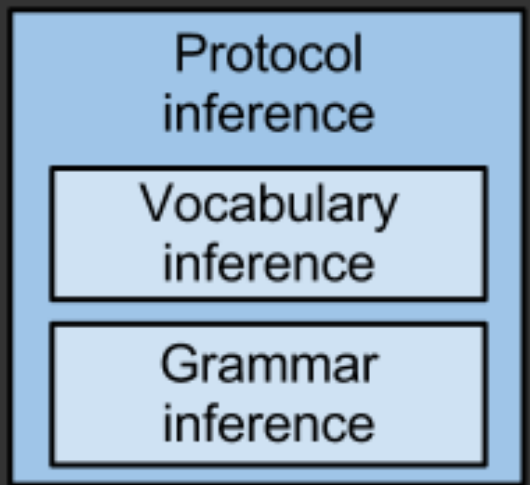
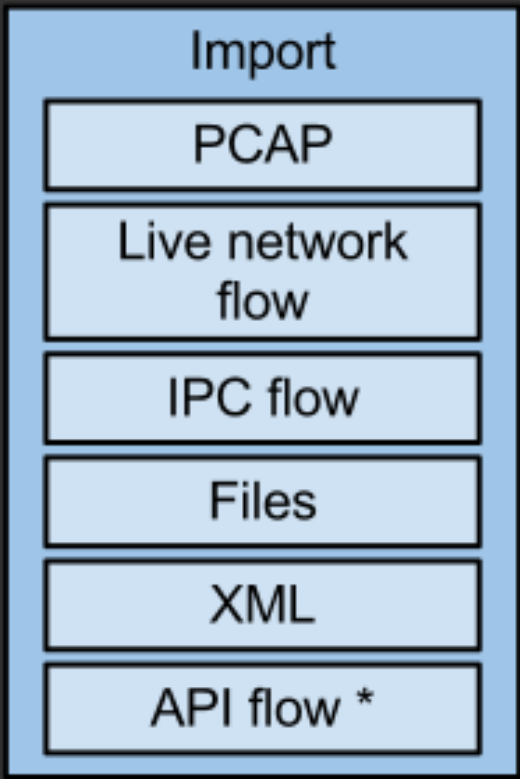
- ▶ Smart-fuzz targeted implementations

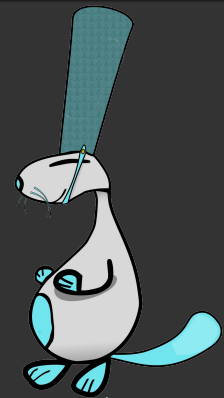
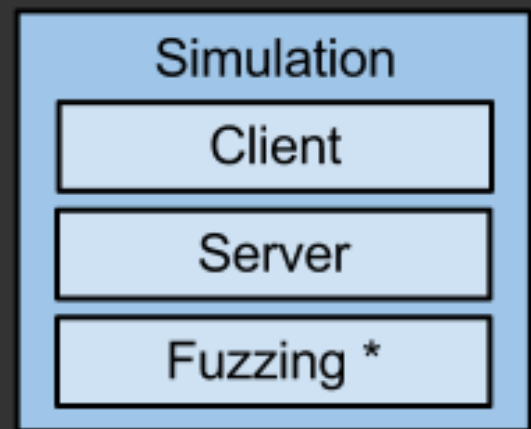
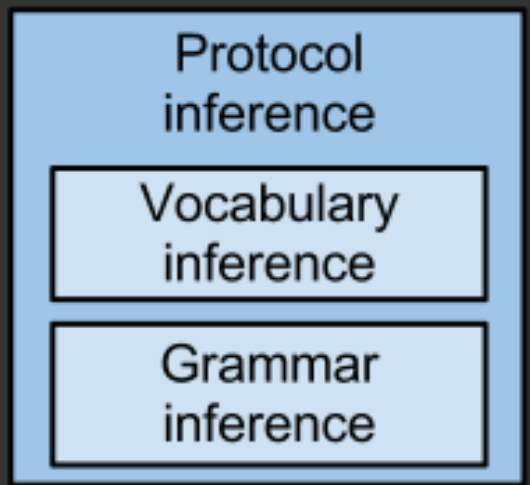
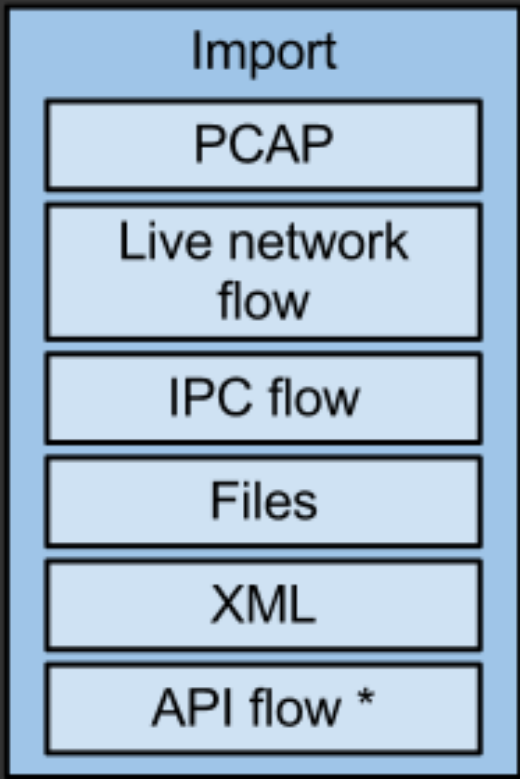


Approach taken by Netzob

- **Passive** and **active** inference
- Semi-Automatic Approach
- **No binary manipulation**

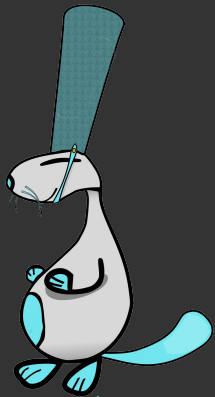






Netzob implementation

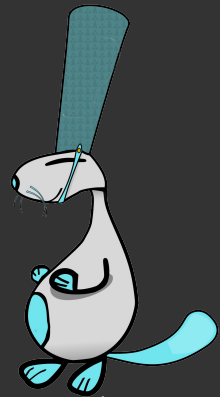
- Graphical interface (GTK3)
- Mostly written in **Python** and C (46 000 LOC)
- Plugin architecture
- Available through
 - **GIT Repositories**
 - Python package
 - Per-Os packages (Debian, Ubuntu, Gento, ...)



Netzob team (chronologically ordered)

- Georges Bossert
- Frédéric Guihéry
- Guillaume Hiet
- Olivier Tétard
- Maxime Olivier
- Alexandre Pigné
- Goulven Guiheux
- Frank Roland
- Fabien André
- Quentin Heyler
- Benjamin Dufour
- Giuseppe massaro

Netzob's Sponsors



netzob.org

« State of the art » boundaries



Fuzzing

Language Theory

Netzob

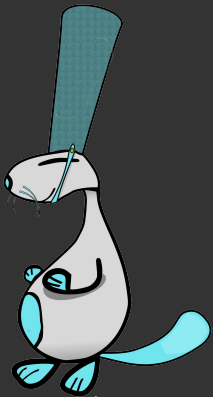
Reverse Engineering

Grammar Inference

Botnet Behavioural Analysis

Sum of human knowledge

The unknown



netzob.org

NEW « State of the art » boundaries



Netzob

Fuzzing

Language Theory

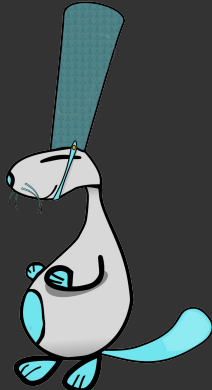
Reverse Engineering

Grammar Inference

Botnet Behavioural Analysis

New sum of human knowledge

The unknown



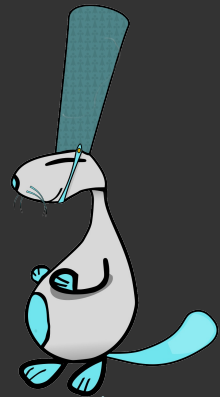
A black and white photograph of a server room. A person is in the center, looking down at a laptop. The room is filled with rows of server racks on both sides, creating a perspective that leads to a bright light at the end of the aisle. The lighting is dramatic, with strong highlights and deep shadows.

RE Zero Access C&C protocol with Netzob

Zero Access (aka Sirefef)

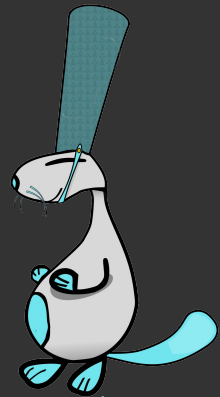
- ▶ Recent botnet (Sept. 2011)
 - ▶ still in activity
- ▶ +/- 1 million zombies (9 millions installed)
- ▶ **Click fraud and bitcoin miner**
- ▶ At least 2 versions of the rootkit
 - ▶ Upgraded **P2P** protocol

Based on Sophos and Kindsight Reports

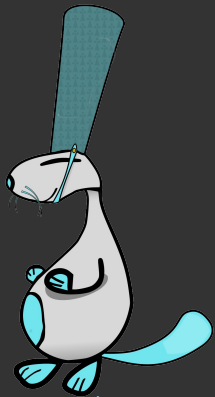


Zero Access (aka Sirefef)

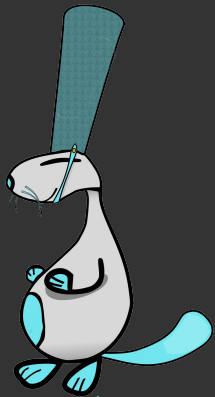
- ▶ Multiple **P2P management messages**
 - ▶ Peers directory retrieval
 - ▶ Files directory retrieval
- ▶ UDP & TCP connections
 - ▶ UDP for messages (udp:16464)
 - ▶ TCP for data
 - ▶ **Hard coded Bootstrap Peers**
 - ▶ Ex : 68.51.108.245, (...), 216.211.181.226



Let's play with its P2P protocol



Requirements



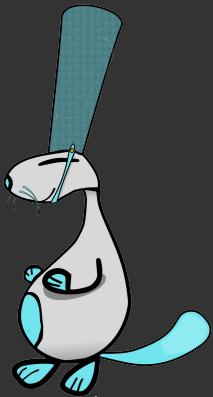
Few **real** communication traces

ZAccess : some traces were provided by

Kevin McNamee

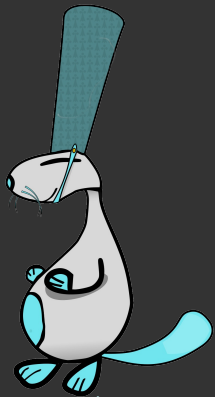
an **infected machine**

Contagio (<http://contagiodump.blogspot.com>)



A **confined** environment and **the binary**

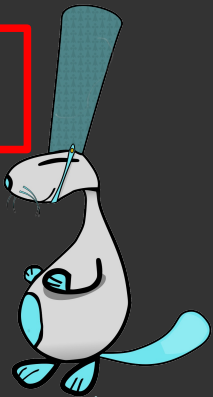
Adapted Virtual Machines + Firewalls + Torify + management
system



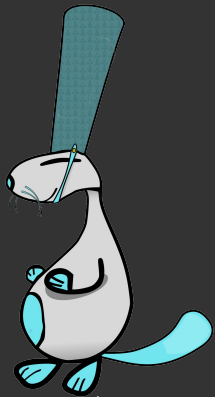
A **confined** environment and **the binary**

Adapted Virtual Machines + Firewalls + Torify + management system

Warning : Consider legal issues before dealing with this !

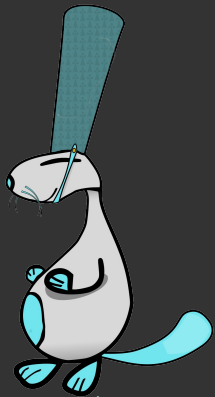


Step 1 : Get messages



Capture **dataflows**

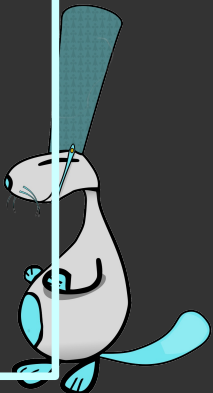
(Network, USB, IPC, API Hooking, Raw files, ...)



Capture dataflows

(Network, USB, IPC, API Hooking, Raw files, ...)

```
?..b(.....@.p..aU(.....3.Mi.L...".f3.8:+.6..d...x,'3...>4...L.....:3.8;..q..d...Q..3.....  
L.QsY.8:3..m.0...d.E...3.Zj."...L.F...8:3->ad...s  
.....3%.GgM...|.C.8:...}...o.....(t0...iW.....oj..`?...~!.c.p./e..z.....%gl.T.s.mE..+-.;  
R.{eoDz.}At..d.QP3.SX=):.JU.7`.t...g.V..5.2.....OW..f2p.X8(.....r..7.....S.&8...!  
'p.....<.  
..g3...C....&(P.T...y....5..2.....v...&.....iM..e  
....].LP...N..X.SV.JA.M.....#Y{...;XVptd....t..  
....=O.?... Qpga("...;...^:....LU.>!.i.1=...ui.  
L8.7c....@.....}1iN.7...25t.G.)53,.....S.....9!.L5.J.  
?..b(.....@.p.gN.(.....3V..L.....f3.8:?e..d.....3...x.F.L...u.:3.8;Wu..d.|....3.._'!..L.[Z.  
8:3.h.4...d.b...3.....L..k.8:3}.e.d.....3%.GgM...|.C.8:...}...o.....(t0...iW.....oj..  
`?...~!.c.p./e..z.....%gl.T.s.mE..+-.;R.{eoDz.}At..d.QP3.SX=):.JU.7`.t...g.V..5.2.....  
OW..f2p.X8(.....r..7.....S.&8...!  
'p.....<.  
..g3...C....&(P.T...y....5..2.....v...&.....iM..e  
....].LP...N..X.SV.JA.M.....#Y{...;XVptd....t..  
....=O.?... Qpga("...;...^:....LU.>!.i.1=...ui.  
L8.7c....@.....}1iN.7...25t.G.)53,.....S.....9!.L5.J.
```



Split dataflows in messages

(sub protocol knowledge, time based, delimiter...)

?.b(.....@.p...aU(.....3.Mi.L..."..f3.8:+.6..d..x,'3...>4...L.....:3.8;..q..d..Q.3.....

Message 1

L...E...3.Zj."...L.F..8:3->ad...s

...8:...}...o.....(t0...iW.....oj..`?...~!.c.p./e..z.....%gl.T.s.mE..+-.;

R.{eoDz.}At..d.QP3.SX=)::JU.7`.t...g.V.5

Message 2

'p.....<.

..g3...C....&(P.T...y....5..2.....v...&.....iM..e

....].LP...N..X.SV.JA.M.....#Y.{...;XVptd....t..

....=.O..?... Qpga("...;...^:....LU..>!.i.1=...ui.

L8.7c....@.....}1iN.7...25t.G.)53,.....S.....9!.L5.J.

?.b(.....@.p.gN..(.....3V..L.....f3.8:?e..d.....3...x.F.L...u...:3.8;Wu.d.|....3.._'!..L.

Message 3

[Z.L..k.8:3}.e.d.....3%.GgM....|.C.8:...}...o.....(t0...iW.....oj..

`?...~!.c.p./e..z.....%gl.T.s.mE..+-.;R.{eoDz.}At..d.QP3.SX=)::JU.7`.t...g.V..5.2.....

OW..f2p.X8(.....r..7.....S..&8...!.

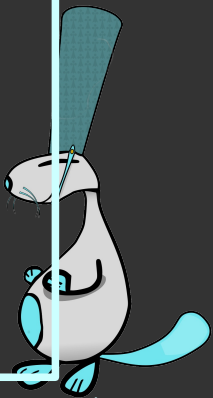
'p.....<..g3...C....&(P.T...y....5..2.....v...&.....iM..e

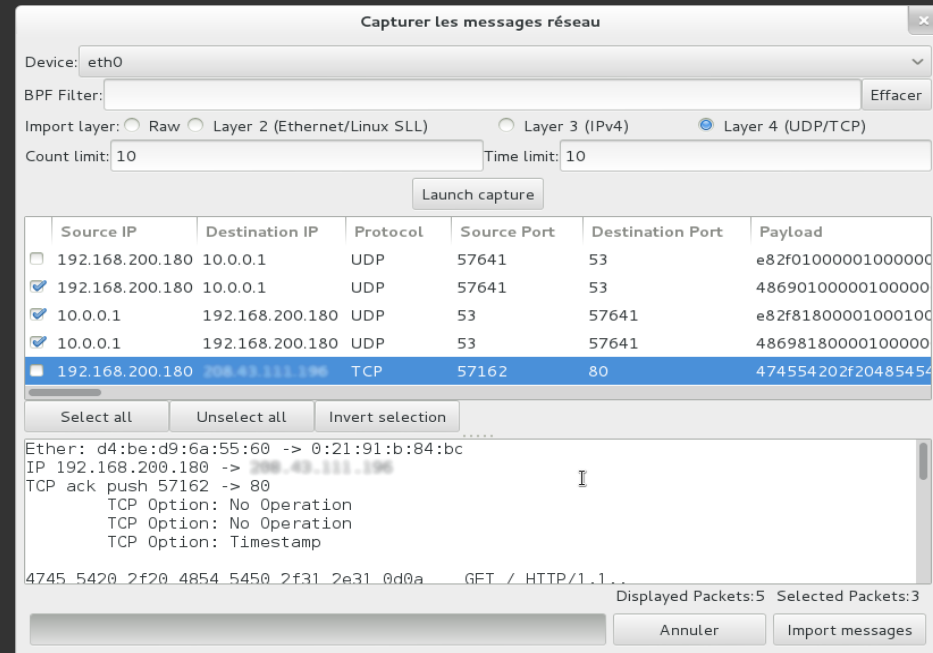
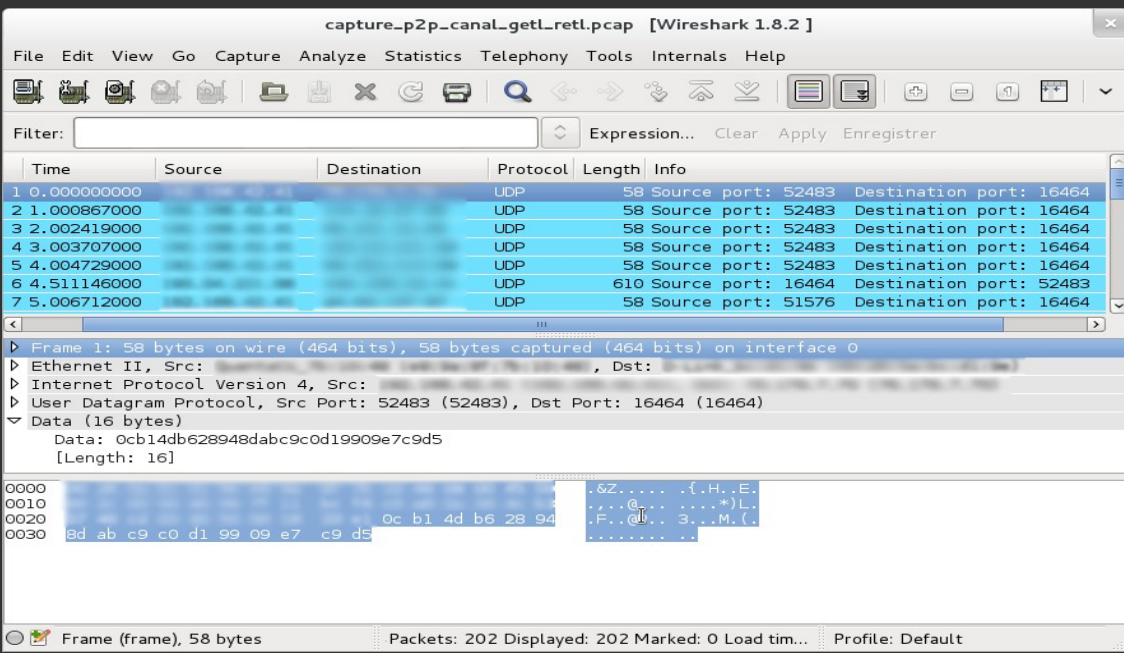
....].LP...N..X.SV.JA.M.....#Y.{...;XVptd....t..

....=.O..?... Qpga("...;...^:....LU..>!.i.1=...ui.

Message 4

L8.7c....@.....}1iN.7...25t.G.)53,.....S.....9!.L5.J.



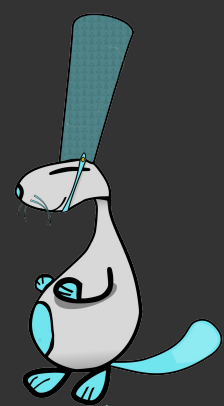


Import

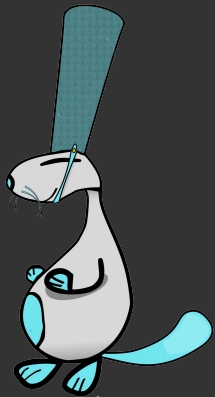
Capture

Netzob framework

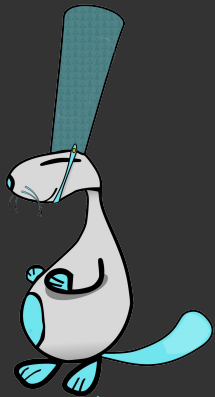
Filter imported messages
 Choose layer of import



Step 2 : RE vocabulary

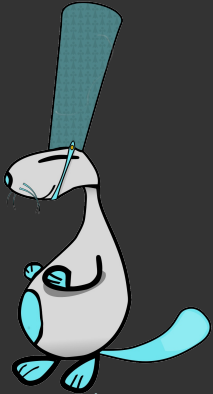


Abstract messages



1 message = a sorted **received or sent**
sequence of bits

010110101001000101010110100101011101010001010010



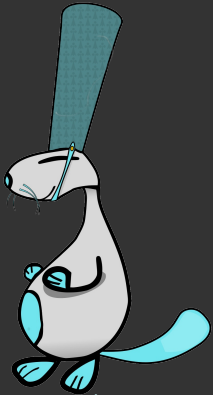
1 message = a sorted **received or sent**
sequence of bits



specific to a context

Emails, IPs, Timestamps, BID, AddID, ...

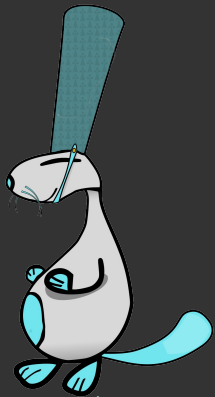
01011010**1001000101010110100101011101010001010010**



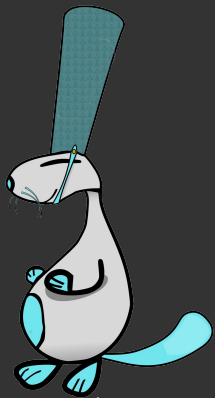
We have to **decontextualize** messages

The IDEA :

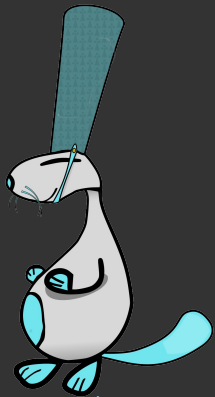
Regroup messages by **similarity** and find
contextual variations



We consider similar messages based on their
common partitioning



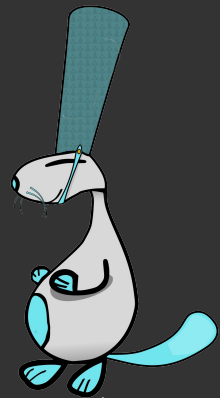
Messages are splitted in **Fields** using



Messages are splitted in **Fields** using

- ▶ Simple Alignment
- ▶ Delimiter-based Alignment
- ▶ Sequence Alignment

3e341eb5ce	4c068e	c2d5baed3a	331938	3b108271e8
dc18fcb8ce	4c068e	2da8f3e33a	331938	cf48cd8fe8
dc18fcb8ce	4c068e	2da8f3e33a	331938	cf48cd8fe8



Messages are splitted in **Fields** using

- Simple Alignment
- **Delimiter-based Alignment**
- Sequence Alignment

cfa0	4c	5519e43e2e27fa6916313dc89ac77569a00d	4c	38f
cfa0	4c	5519e43e2e27fa6916313dc89ac77569a00d	4c	38f
cfa0	4c	5519e43e2e27fa6916313dc89ac77569a00d	4c	38f

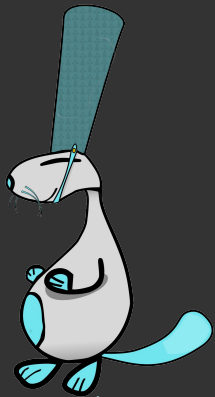


Messages are splitted in **Fields** using

- Simple Alignment
- Delimiter-based Alignment
- Sequence Alignment

Needleman & Wunsch

3a8	70	832f65bd867ad2	00	d9aeddc
3a8	70	832f65bd867ad2	00	d9aeddc
	70	c400	00	
	70	c400	00	



But WTF is Needleman & Wunsch ?

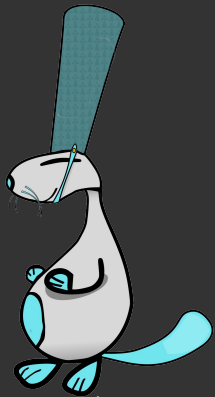


Sequence alignment with Needleman-Wunsh applied to RE of protocols (c.f. Marshall Bedoe)

We start with 2 messages

70 83 2f 65 bd 86 7a d2 00

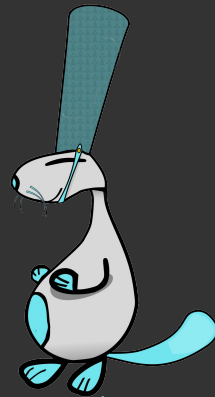
70 c4 00 00



Sequence alignment with Needleman-Wunsh

		70	83	2f	65	bd	86	7a	d2	00
70										
c4										
00										
00										

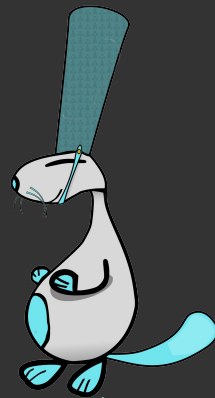
We build a distance matrix



Sequence alignment with Needleman-Wunsh

		70	83	2f	65	bd	86	7a	d2	00
	0	0	0	0	0	0	0	0	0	0
70	0									
c4	0									
00	0									
00	0									

We initialize the matrix



Sequence alignment with Needleman-Wunsh

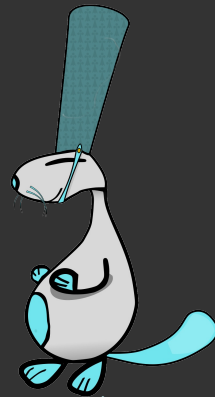
		70	83	2f	65	bd	86	7a	d2	00
	0	0	0	0	0	0	0	0	0	0
70	0	?								
c4	0									
00	0									
00	0									

We fill the matrix with the formula:

$$M(i,j) = \text{Max}(M(i-1, j-1) + S, M(i, j-1) + W, M(i-1, j) + W)$$

S: Match/Mismatch score (+/- 10)

W: Gap score (0)

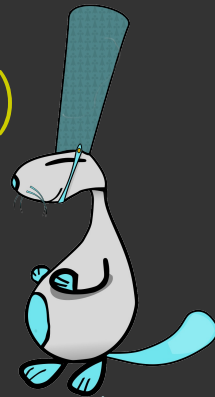


Sequence alignment with Needleman-Wunsh

		70	83	2f	65	bd	86	7a	d2	00
	0	0	0	0	0	0	0	0	0	0
70	0	?								
c4	0									
00	0									
00	0									

We fill the matrix with the formula:

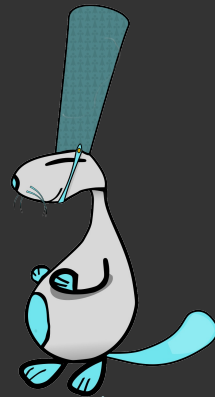
$$M(i,j) = \text{Max}(M(i-1, j-1) + S, M(i, j-1) + W, M(i-1, j) + W)$$



Sequence alignment with Needleman-Wunsh

		70	83	2f	65	bd	86	7a	d2	00
	0	0	0	0	0	0	0	0	0	0
70	0	10	10	10	10	10	10	10	10	10
c4	0	10	10	10	10	10	10	10	10	10
00	0	10	10	10	10	10	10	10	10	20
00	0	10	10	10	10	10	10	10	10	20

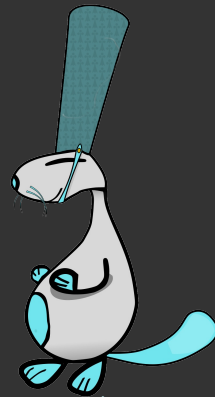
We fill the matrix



Sequence alignment with Needleman-Wunsh

		70	83	2f	65	bd	86	7a	d2	00
	0	0	0	0	0	0	0	0	0	0
70	0	10	10	10	10	10	10	10	10	10
c4	0	10	10	10	10	10	10	10	10	10
00	0	10	10	10	10	10	10	10	10	20
00	0	10	10	10	10	10	10	10	10	20

We do a traceback



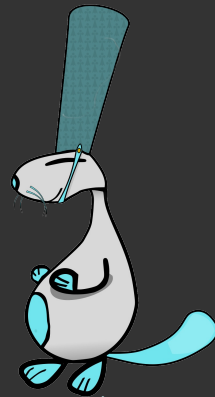
Sequence alignment with Needleman-Wunsh

		70	83	2f	65	bd	86	7a	d2	00
	0	0	0	0	0	0	0	0	0	0
70	0	10	10	10	10	10	10	10	10	10
c4	0	10	10	10	10	10	10	10	10	10
00	0	10	10	10	10	10	10	10	10	20
00	0	10	10	10	10	10	10	10	10	20

We compute the common pattern

70 83 2f 65 bd 86 7a d2 00

70 c4 00 -- -- -- -- 00



Sequence alignment with Needleman-Wunsh

We finally build a regex

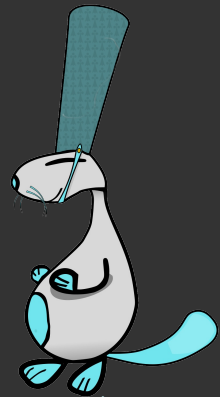
70 83 2f 65 bd 86 7a d2 00

70 c4 00 -- -- -- -- 00

(70)

(.*{2,7})

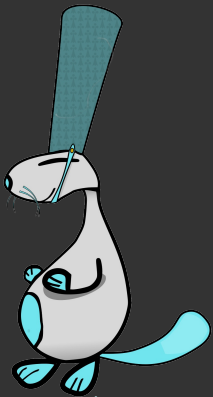
(00)



Static Fields

3a8	70	832f65bd867ad2	00	d9aedd
3a8	70	832f65bd867ad2	00	d9aedd
	70	c400	00	
	70	c400	00	

Symbol x → [..., 0x70, (.*){4,14}, 0x00, ...]

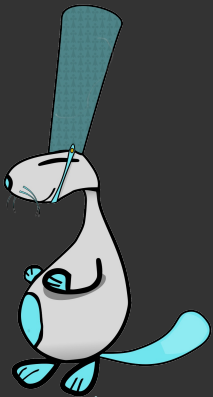


Static Fields

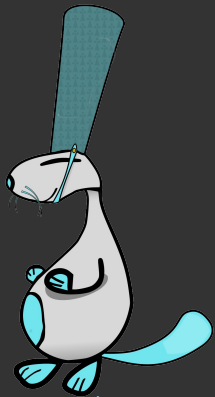
3a8	70	832f65bd867ad2	00	d9aeddc
3a8	70	832f65bd867ad2	00	d9aeddc
	70	c400	00	
	70	c400	00	

Dynamic Fields

Symbol x → [..., 0x70, (.*){4,14}, 0x00, ...]



How to measure **similarity** between two messages ?

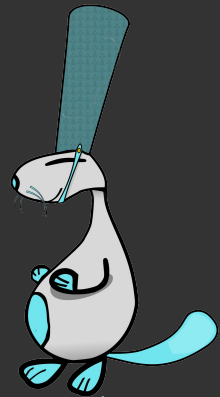


Measure the Quality of Fields

0 % < **Similarity Score** < 100 %

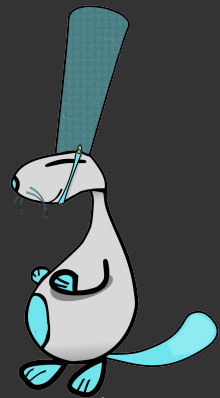
Messages have
Nothing in common

Messages are
identicals



Similarity scores between messages

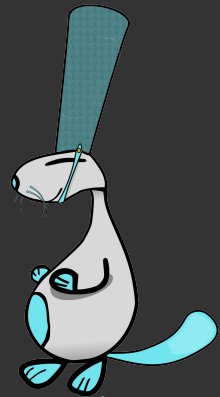
- ▶ S1: ratio of **dynamic fields / bytes**
- ▶ S2: ratio of **common dynamic bytes**



Similarity scores between messages

- ▶ S1: ratio of **dynamic fields / bytes**
- ▶ S2: ratio of **common dynamic bytes**

The design of Netzob allows for inclusion of future factors

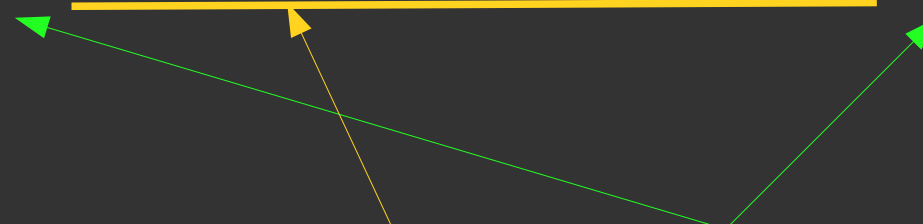


Similarity scores between messages

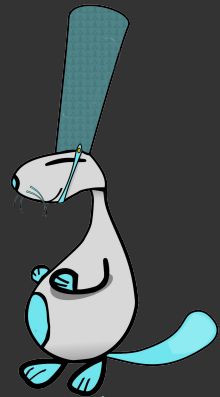
- ▶ S1: ratio of **dynamic fields / bytes**
- ▶ S2: ratio of common dynamic bytes

70 83 2f 65 bd 86 7a d2 00

70 c4 00 -- -- -- -- -- 00



$$S1 = 1 / (1 + 2)$$



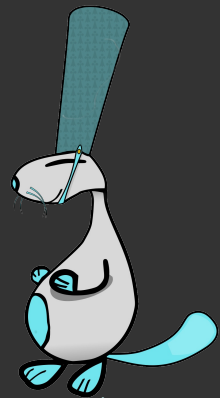
Similarity scores between messages

- ▶ S1: ratio of dynamic fields / bytes
- ▶ S2: ratio of **common dynamic bytes**

70 83 2f 65 bd 86 7a d2 00

70 c4 00 -- -- -- -- -- 00

$$S2 = 2 / 7$$

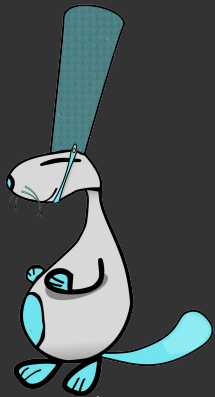


Similarity scores between messages

- ▶ S1: ratio of **dynamic fields / bytes**
- ▶ S2: ratio of **common dynamic bytes**



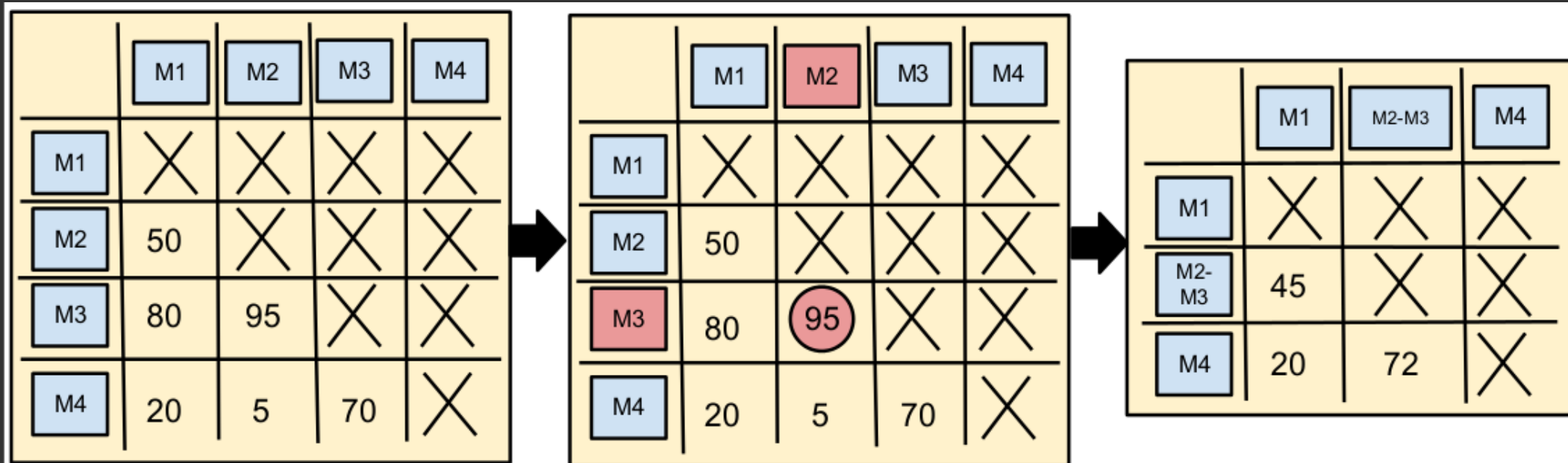
How to retrieve **groups** of similar messages ?



Hierarchical Clustering by similarities:

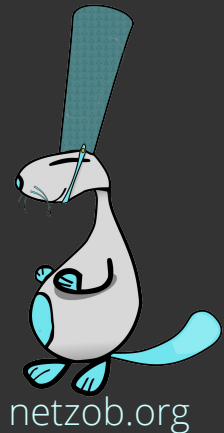
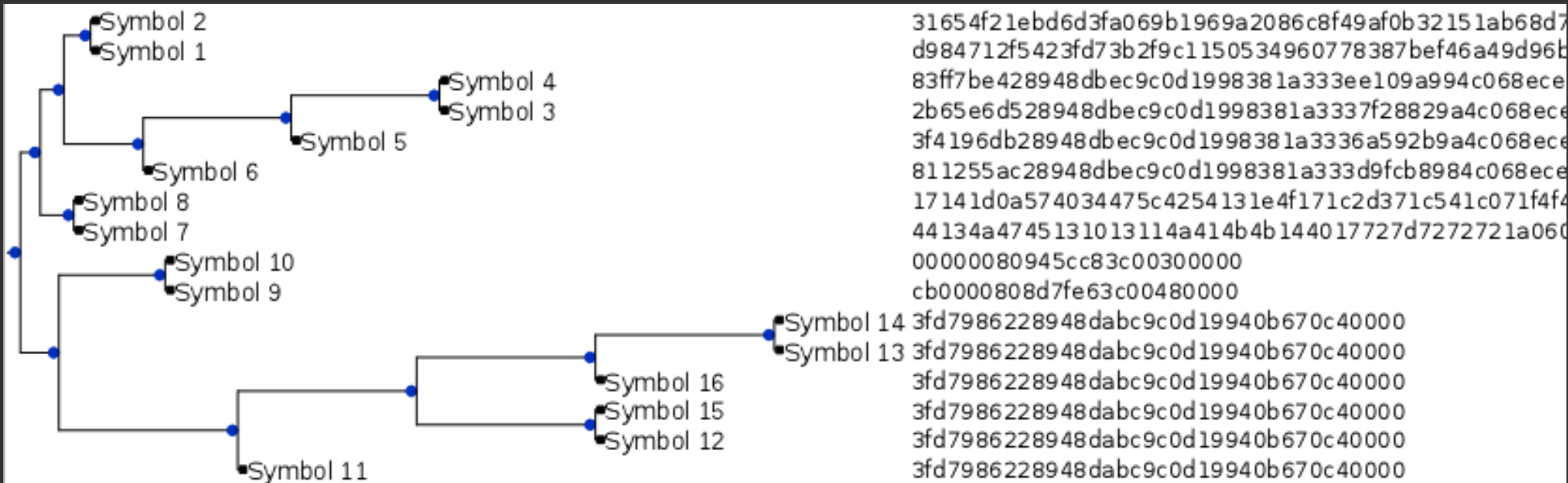
Similarity matrix

UPGMA

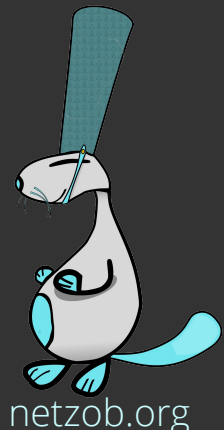
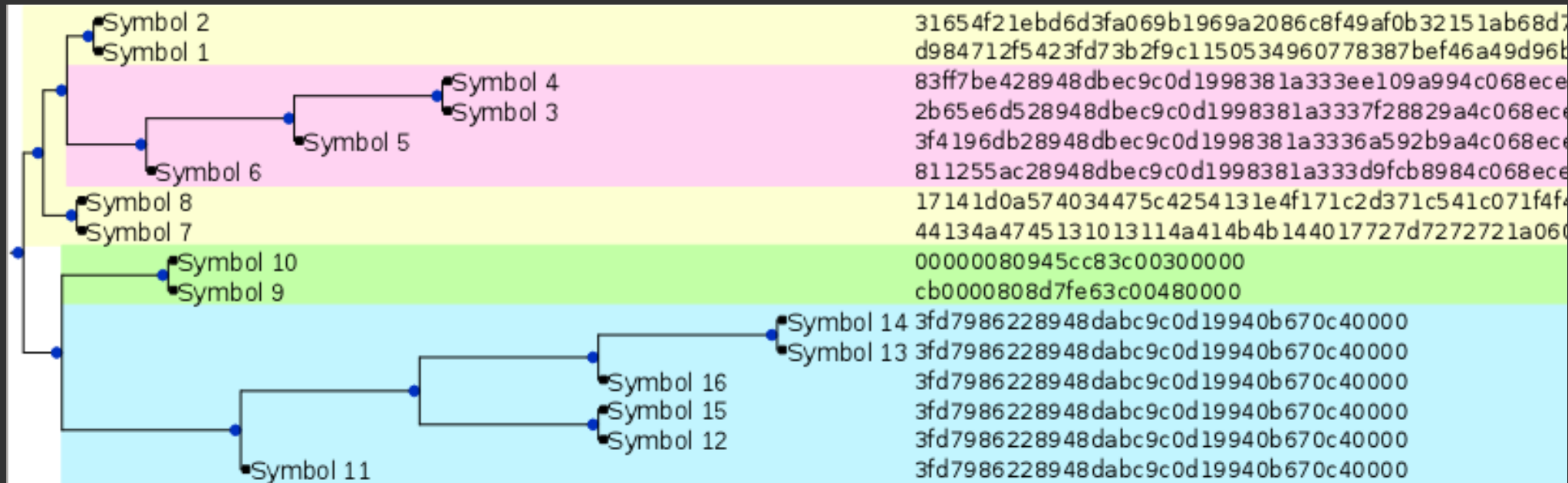


- ▶ Filling of a similarity matrix
- ▶ Iteratively merge the 2 most similar messages

UPGMA creates a similarity tree



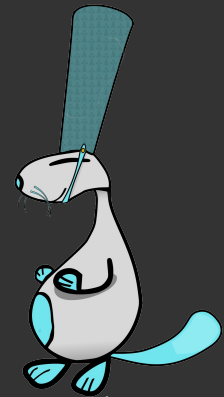
UPGMA creates a similarity tree and facilitates clustering



ZAccess Example

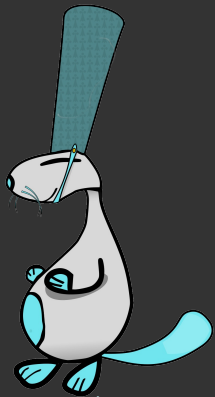
Results of Clustering and Sequence Alignment

Symboles			Symbol 5						
Nom	Message	Champ	Field 0 (.{8}) hex	Field 1 (4c74657200000000) hex	Field 2 (.{2}) hex	Field 3 (000000) hex	Field 4 (.{8}) hex	Field 5 (00000000) hex	Field 6
Symbol 10	5	3	01a76c9f	4c74657200000000	03	000000	bd584ae8	00000000	d2
Symbol 25	8	3	9782e4fb	4c74657200000000	06	000000	c725c5ca	00000000	b7
Symbol 23	10	5	46b9c0a9	4c74657200000000	05	000000	d0b4d6d1	00000000	be
Symbol 5	8	12	b7dafb61	4c74657200000000	0e	000000	deed851e	00000000	aa
			db34786e	4c74657200000000	0c	000000	e029f3c8	00000000	bd
			738a2cf6	4c74657200000000	07	000000	b6d28e36	00000000	8f
			badfa0e7	4c74657200000000	05	000000	b85b8fda	00000000	cd
			5d2676f0	4c74657200000000	04	000000	954cfe6f	00000000	6f

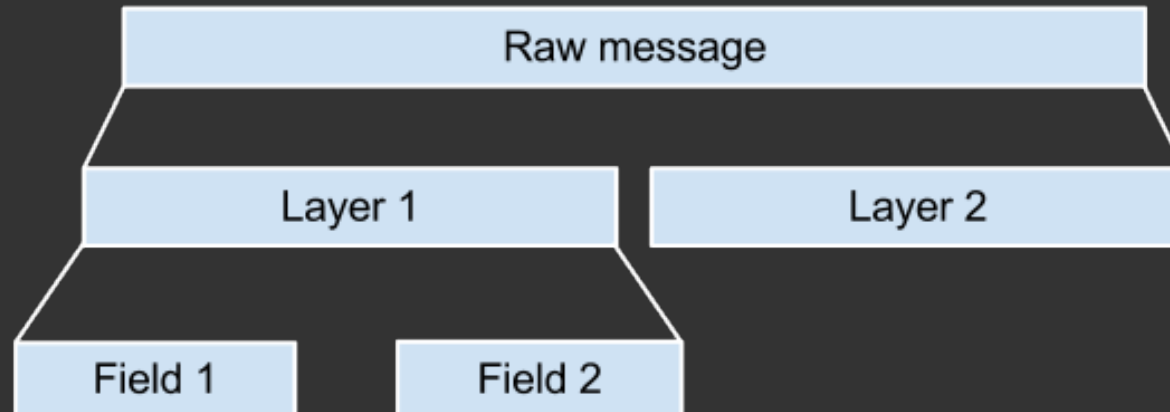


Abstract fields

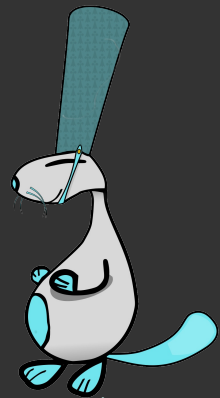
to decontextualize messages



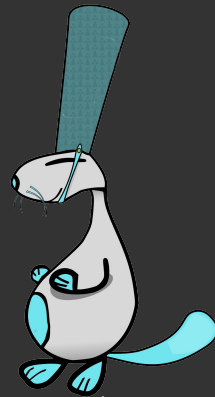
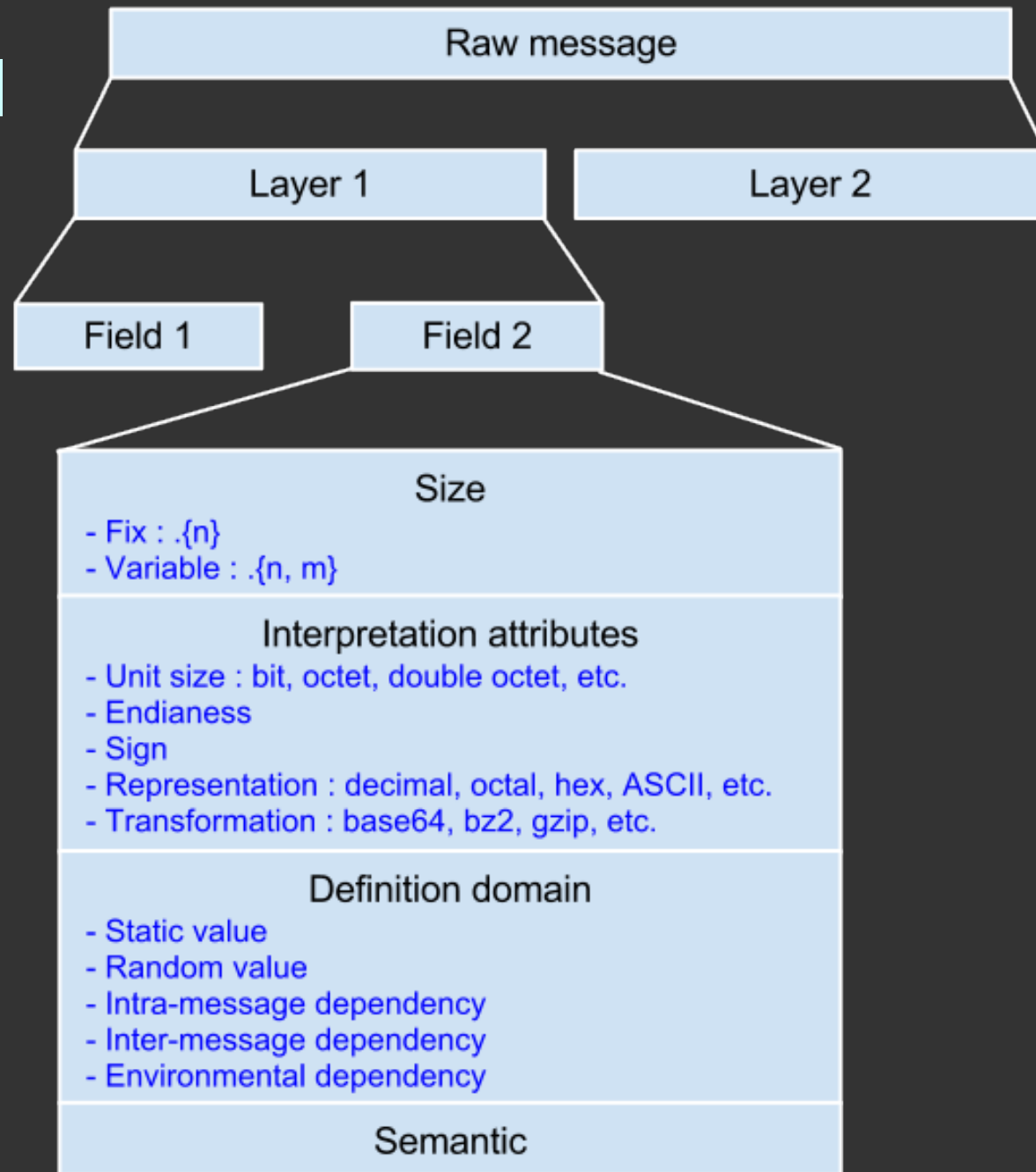
Message format model



Allows multiple partitionment strategies per symbols

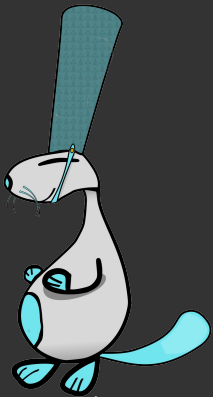


Full message format model



Interlude

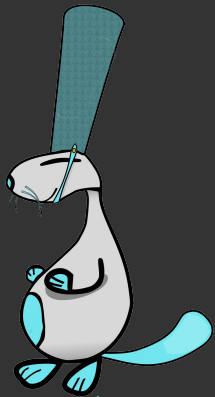
'cauze +/- 60 slides left



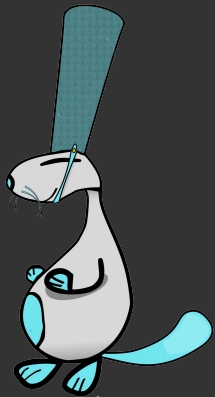
Lets's do 5 minutes of knitting



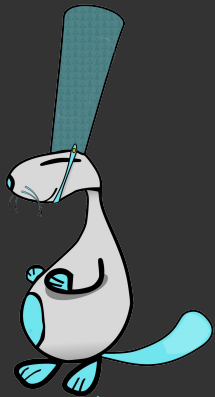
Transformations



How to handle
Encoded values
(XOR, ASN.1) ?

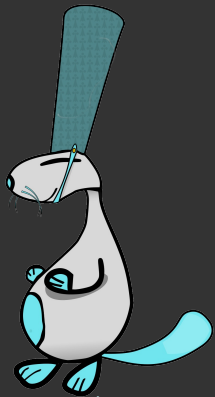


How to handle
Encrypted values
(symmetric, asymmetric, ...)?



Messages include **Transformed Fields**

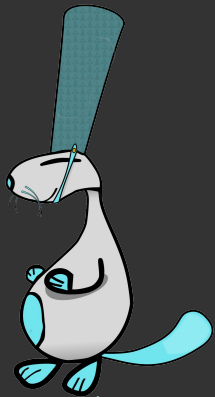
Let's use « **Transformation Functions** »



The idea

Transform raw bytes into application-level bytes

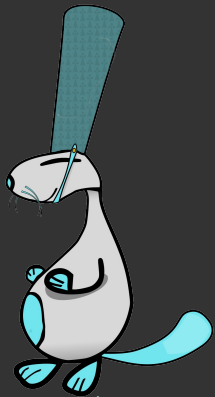
- ▶ Applied either on messages, layers or fields
- ▶ Provided functions (base64, gzip, bz2, ...)
- ▶ Allow custom transformation functions



The idea

Transform raw bytes into application-level bytes

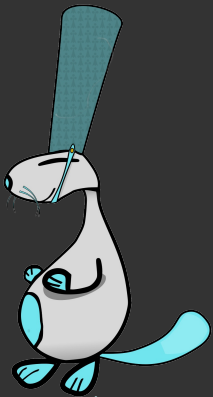
- ▶ Applied either on messages, **layers** or fields
- ▶ Provided functions (base64, gzip, bz2, ...)
- ▶ Allow custom transformation functions



The idea

Transform raw bytes into application-level bytes

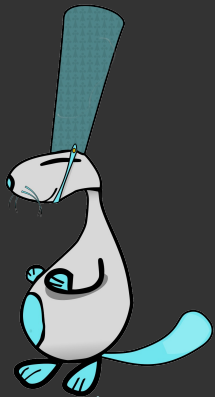
- ▶ Applied either on messages, **layers** or fields
- ▶ Provided functions (base64, gzip, bz2, ...)
- ▶ Allow custom transformation functions



The idea

Transform raw bytes into application-level bytes


- ▶ Applied either on messages, **layers** or fields
- ▶ Provided functions (base64, gzip, bz2, ...)
- ▶ Allow **custom** transformation functions



Adding a custom transformation function

Ex: ZeroAccess XOR-based obfuscation

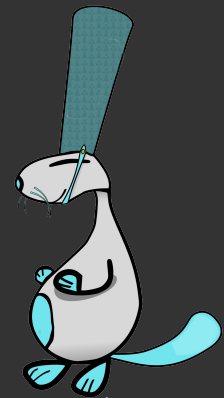
Create a Custom Tra



Custom Transformation Function

Name of the function

```
key=0x66747032
result = []
binMessage = binascii.a2b_hex(message)
for i in range(0, len(binMessage), 4):
    if len(binMessage[i:]) >= 5 :
        subData = struct.unpack("<l", binMessage[i:i+4])[0]
        xoredSubData = subData ^ key
        result.append(struct.pack("<l", xoredSubData))
        key = ((key << 1) & 0xffffffffL | key >> 31)
strMessage = ".join(result)
message = binascii.b2a_hex(strMessage)
```



Adding a custom transformation function

Ex: ZeroAccess XOR-based obfuscation

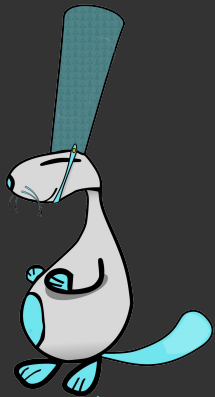
b59e6155	28948dbec9c0d1998381a333	ad4d699b	4c068ece
4adb017b	28948dbec9c0d1998381a333	9b72a59a	4c068ece
8c43c72c	28948dbec9c0d1998381a333	d9fcb898	4c068ece



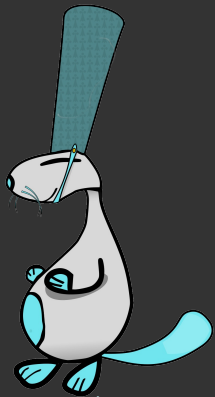
87ee1533	4c74657200000000010000000	8b4e2efc	00000000
78ab751d	4c74657200000000010000000	bd71e2fd	00000000
be33b34a	4c74657200000000010000000	ffffffff	00000000



Search for relations

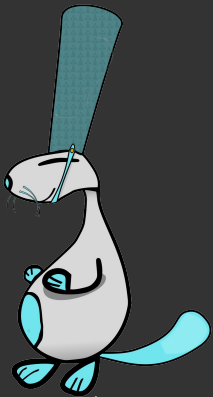


How to handle
when the **value depends** on something ?



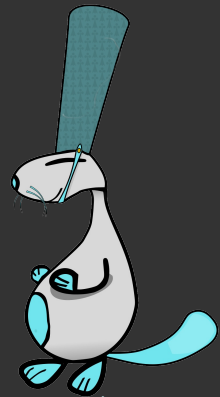
Binary ID, Affiliate ID,
Filenames, *etc.*

« **Inter-Symbol** » and « Intra-Symbol » relations



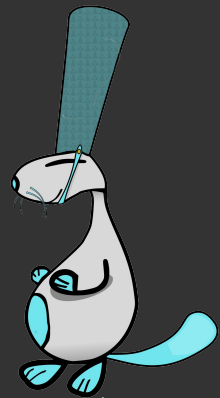
« Inter-Symbol » and « **Intra-Symbol** » relations

└─ Size Fields, CRCs, *etc.*



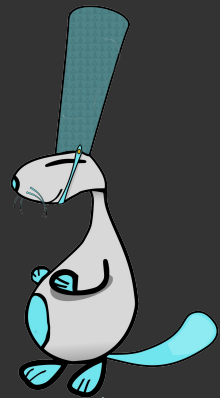
The idea

- ▶ **Correlate** field's size and values with the « Maximal Information Coefficient » (M.I.N.E.)
- ▶ Qualify correlated fields



The idea

- ▶ **Correlate** field's size and values with the « Maximal Information Coefficient » (M.I.N.E.)
- ▶ **Qualify** correlated fields



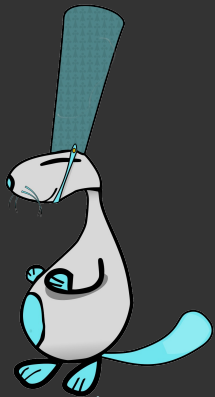
Generate Pairs of data for each field :

Simple way :

- ▶ **Value** of each field
- ▶ **Size** of each dynamic field

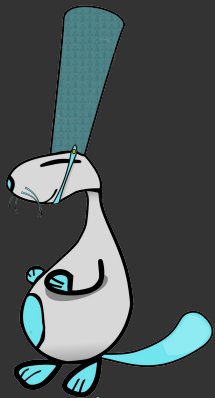
Add more :

- ▶ Concat fields
- ▶ Create n-grams (4bits, 8bits, ...)
- ▶ Consider CRCs, MD5, SHA1,



Search for closest pairs

- ▶ Measure **dependences** between each pairs
 - ▶ Support noisy datasets
- ▶ Rank pairs by their score

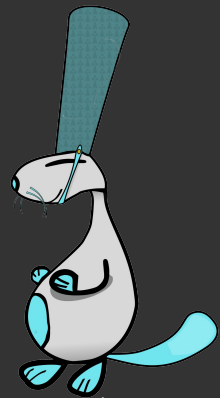


MINE(Value(F1), Size(F2)) = 1

→ Typical Size Field Relation

MINE(Value(F7), CRC32(Value(F1), Value(F2))) = 1

→ Typical CRC Relation

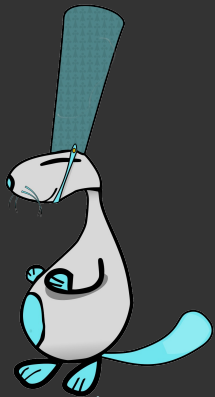


MINE(Value(F1), Size(F2)) = 1

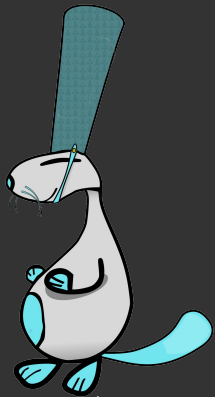
→ Typical Size Field Relation

MINE(Value(F7), CRC32(Value(F1), Value(F2))) = 1

→ Typical CRC Relation



« **Environmental** » dependencies

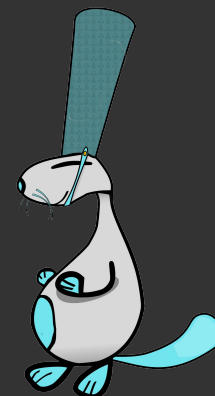


During packets capture:

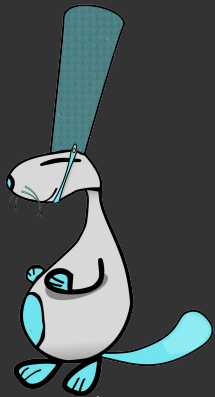
- ▶ Retrieve **contextual** data (IP, port, timestamp, etc.)
- ▶ Store them as meta-data

During vocabulary inference:

- ▶ **Search for meta-data** in messages

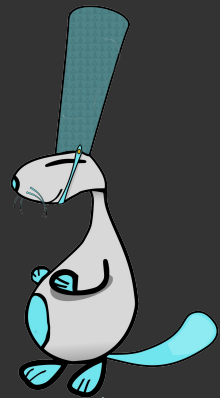


Step 3 : RE grammar



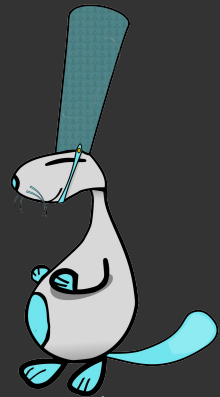
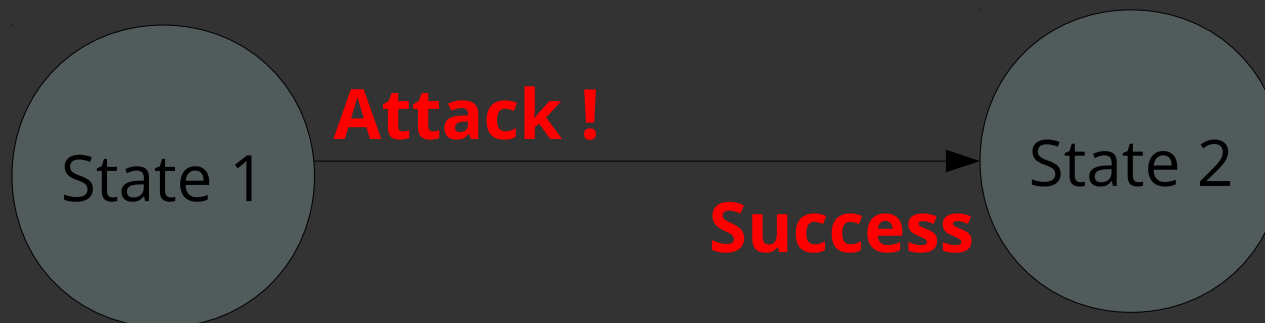
Sequence of valid exchanged symbols.

→ IO Automata

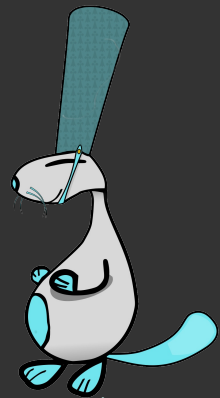


Sequence of valid exchanged symbols.

→ IO Automata

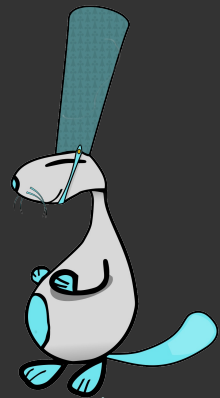
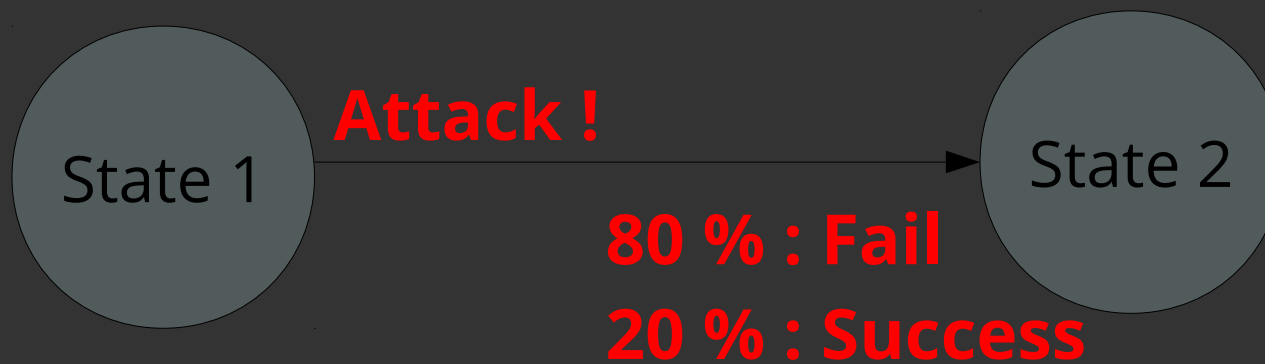


But answers depends on the environment

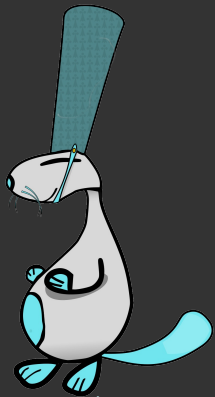
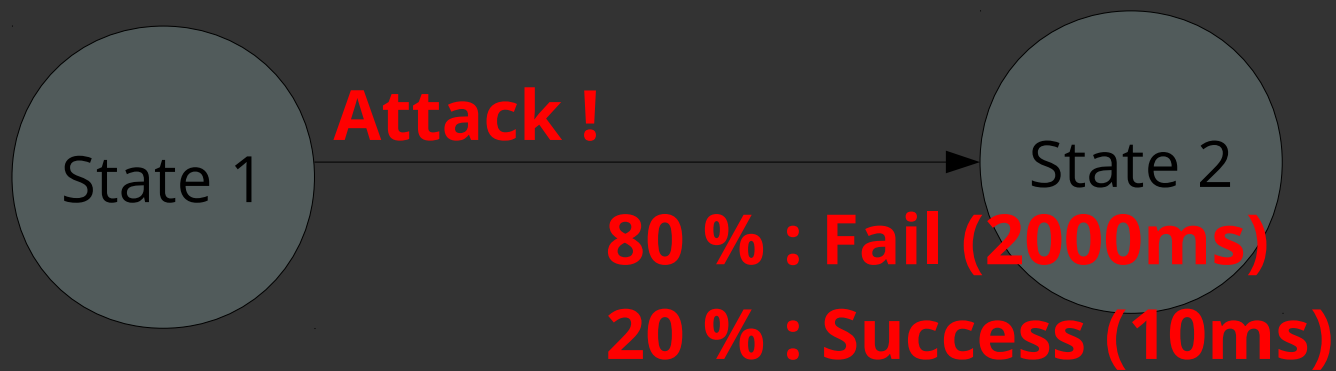


Our model (SMMDT)

→ Add probabilities on output messages

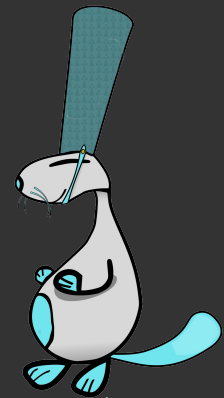
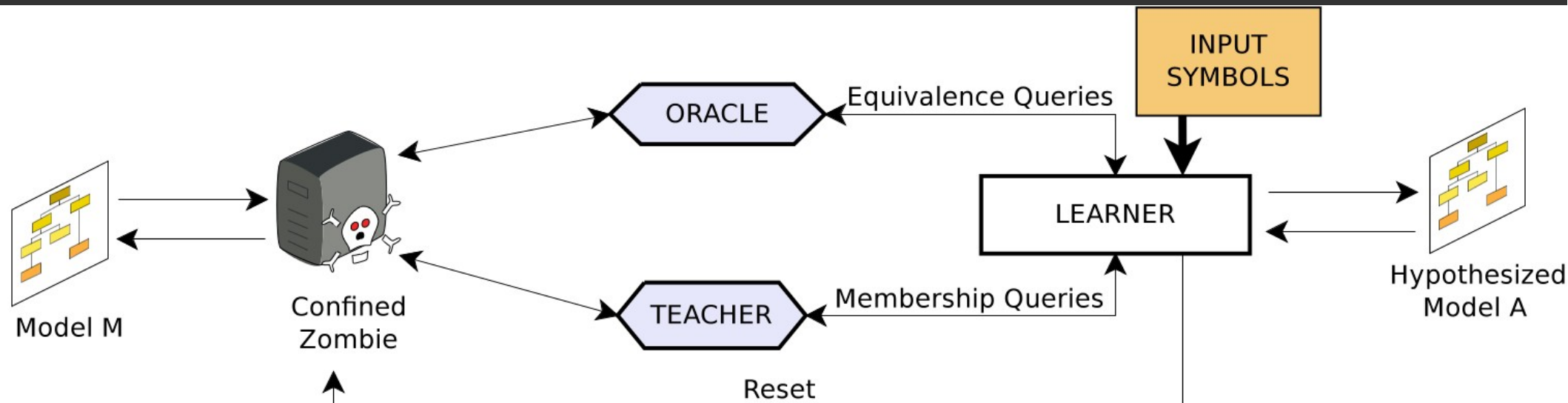


Our model (SMMDT)
→ Add the « reaction time »



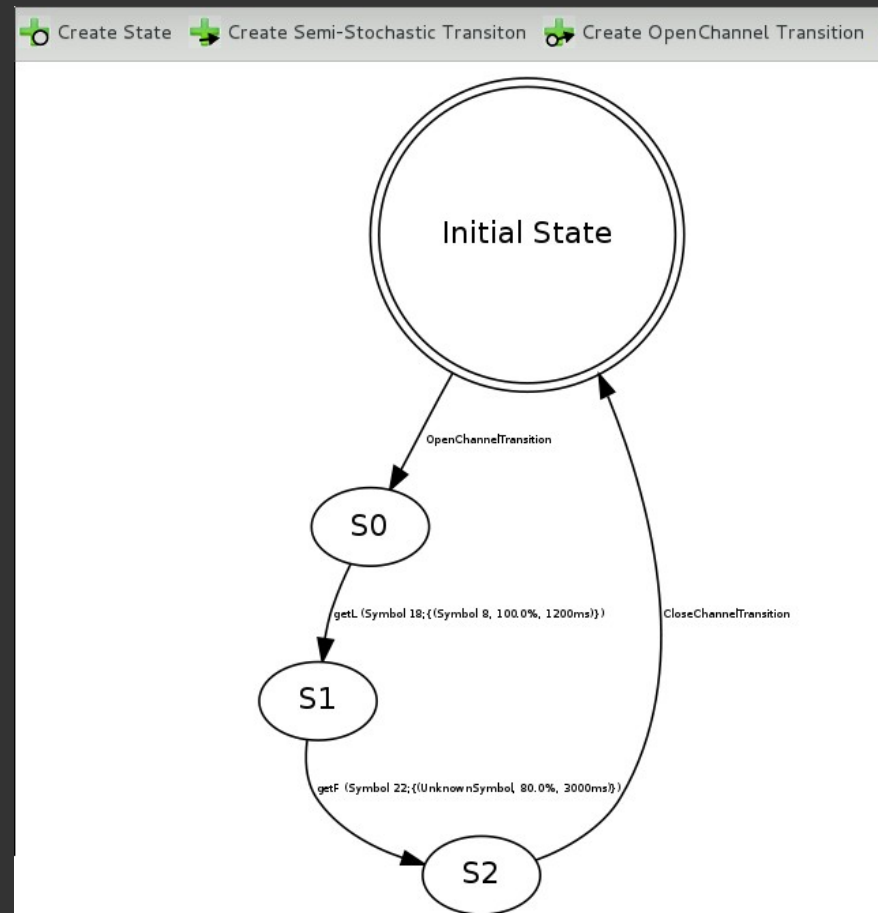
Active Grammatical Inference Process

→ Angluin L*a Algorithm



Active Grammatical Inference Process

→ **Angluin L*a Algorithm**

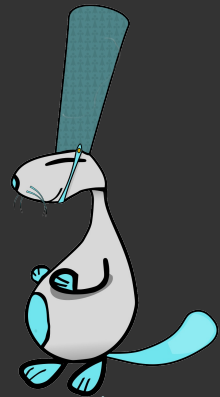




Generating traffic...

Netzob can generate traffic that:

- Follows the **inferred message format**
- Respects the **state machine**



Emulation of different kind of actors and flows

- Client ↔ Real server implementation
- Server ↔ Real client implementation
- **Both client(s) and server**

Create Network Actor

Create a Network Actor

General Informations

ID: d544669c-b30e-46b8-a67c-4d8b15b091

Name: zeroAccessBot

Initiator

Network Configuration

Type: CLIENT

Predefined Values



L4_PROTOCOL: UDP

BIND_IP: 192.168.42.41

BIND_PORT:

TARGET_IP: 115.22.87.69

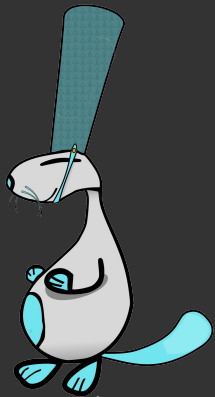
TARGET_PORT: 16464

Annuler  Appliquer 

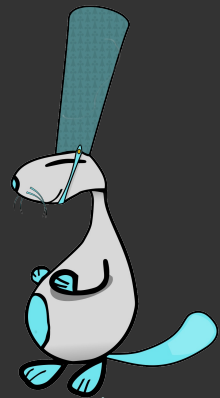
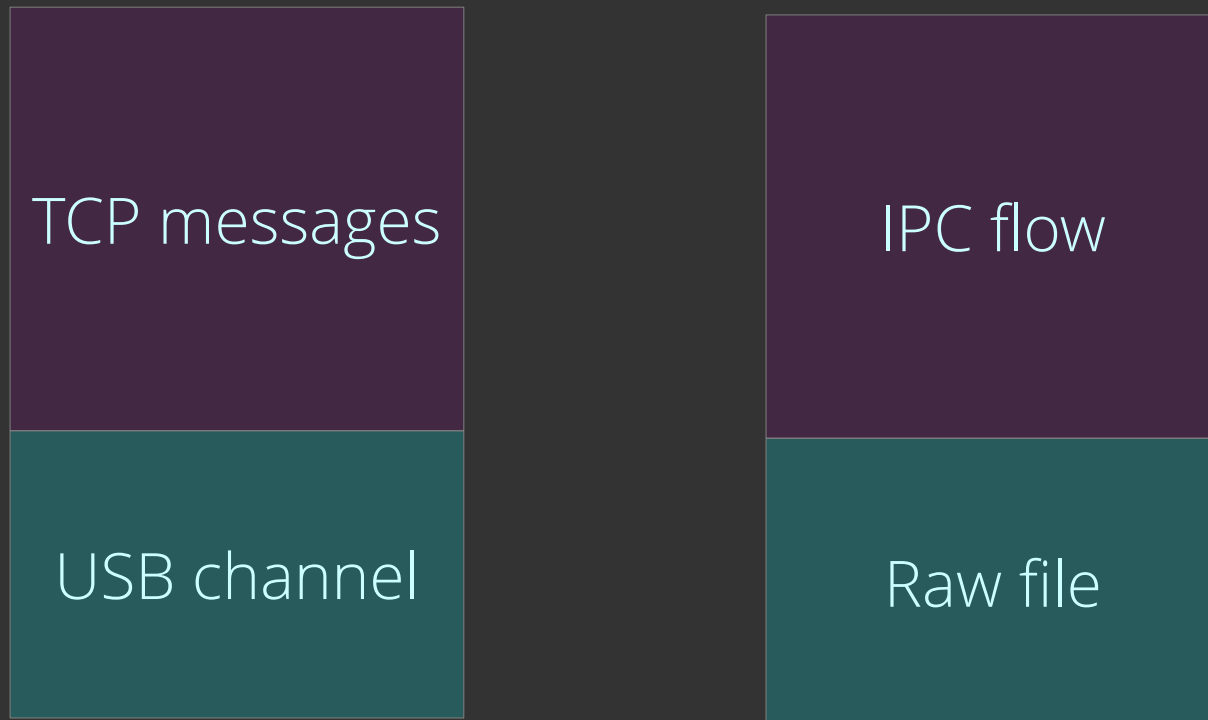
Distinction between

- Client / server
- Initiator / responder of the opening channel

*Ex : TLS with TCP session initiated
by the server*

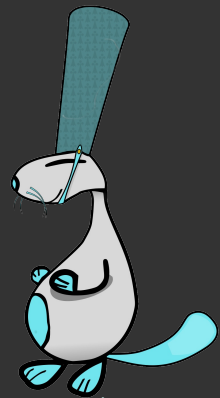


Abstraction from the communication channel



Memory mechanism

- Some received values are **memorized**...
- ...and **reinject**ed in future messages
- Also handles contextual values (IP, time, etc.)



Abstraction and contextualization principles

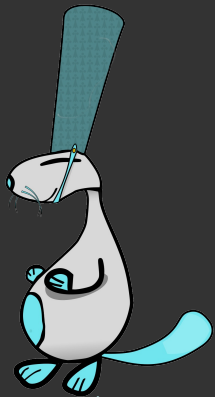
Input device

Input
flow

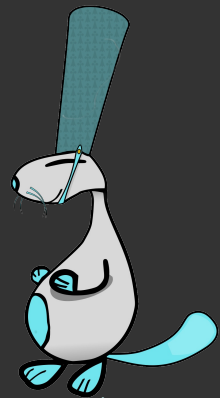
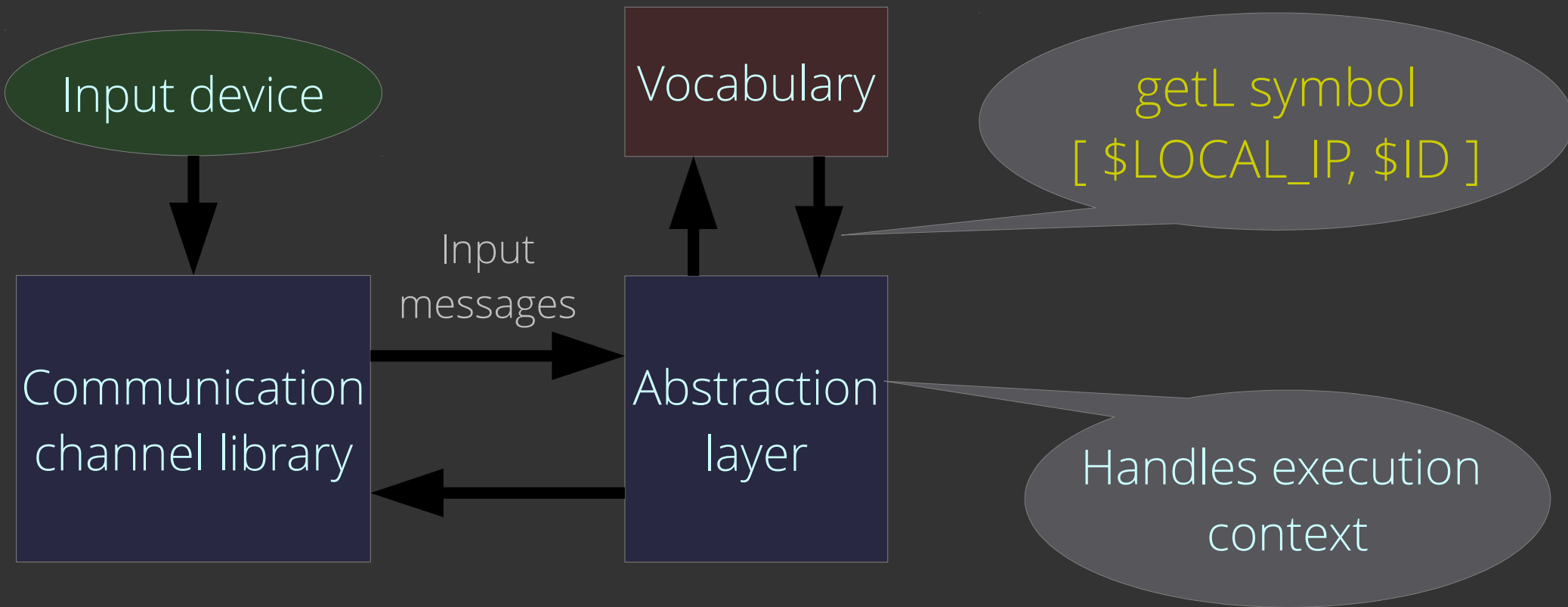


Communication
channel library

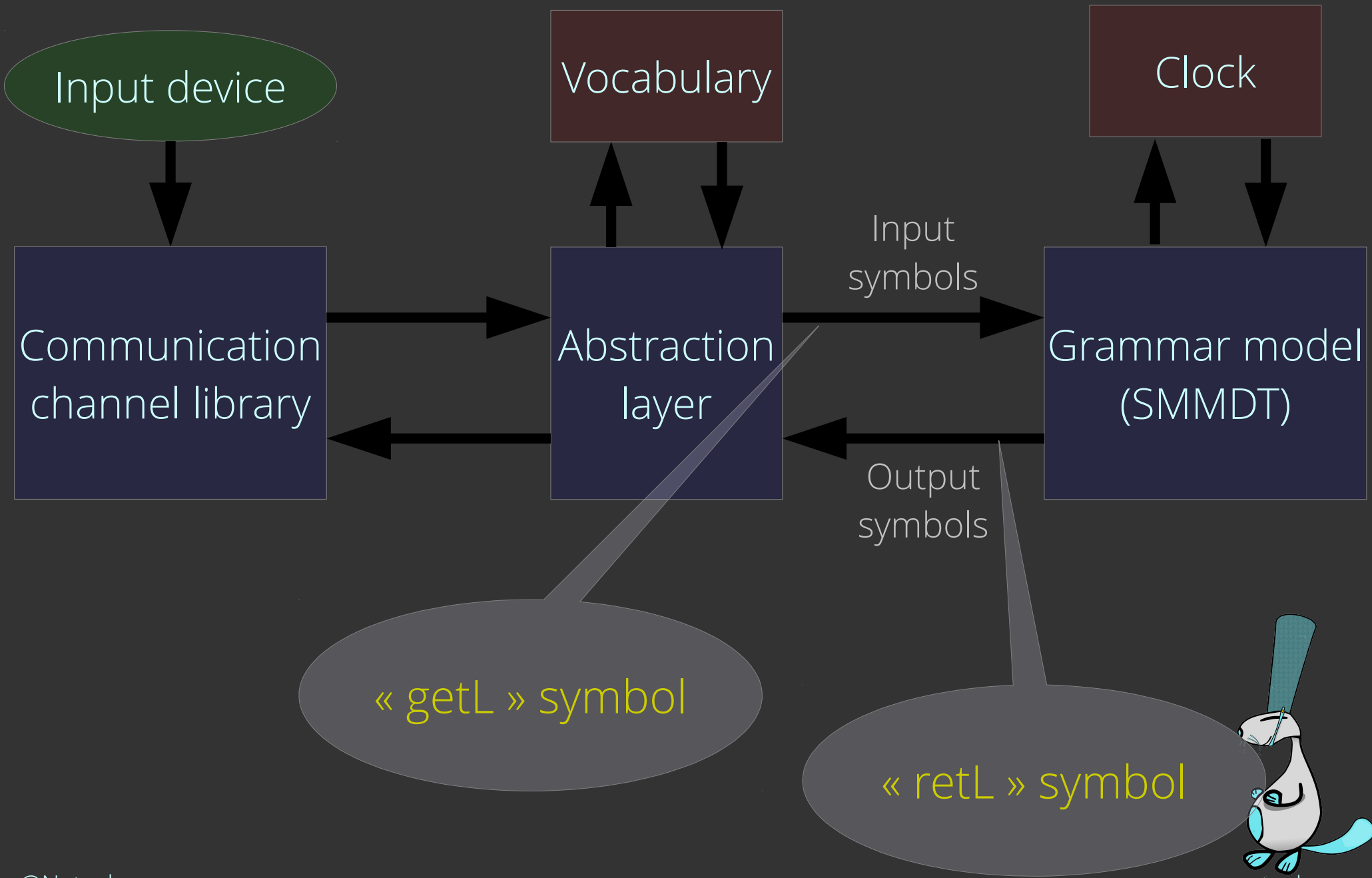
« 70dde8fc00000003 »



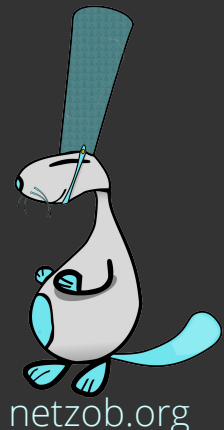
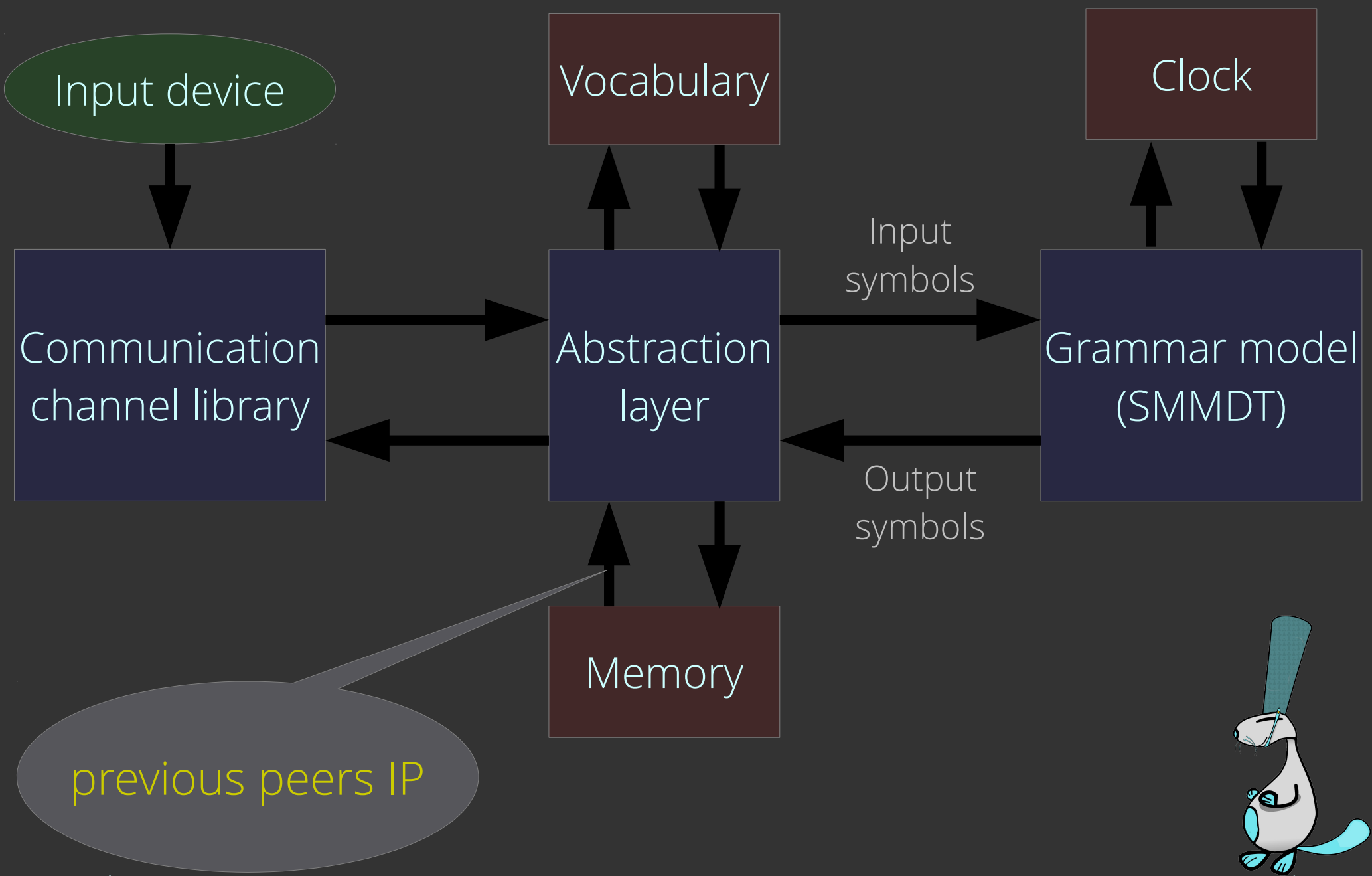
Abstraction and contextualization principles



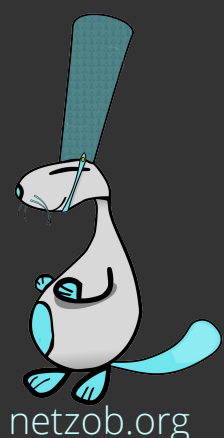
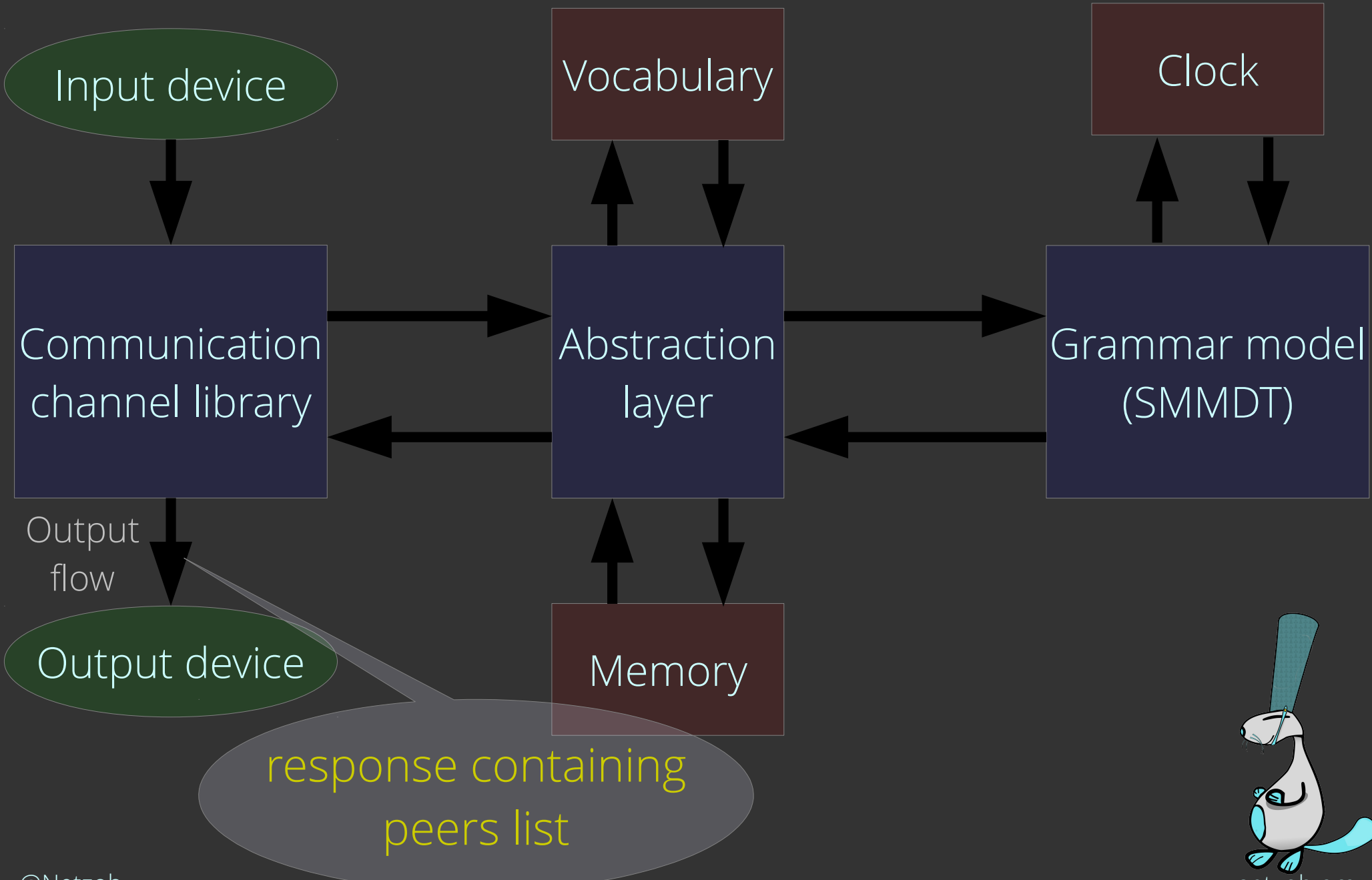
Abstraction and contextualization principles



Abstraction and contextualization principles



Abstraction and contextualization principles



FINAL BATTLE

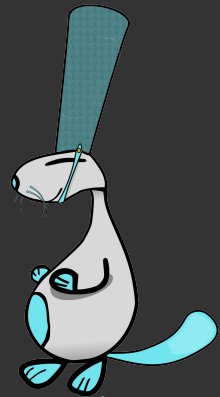
Demos



DEMOS

Demo 1: reversing the protocol of ZA

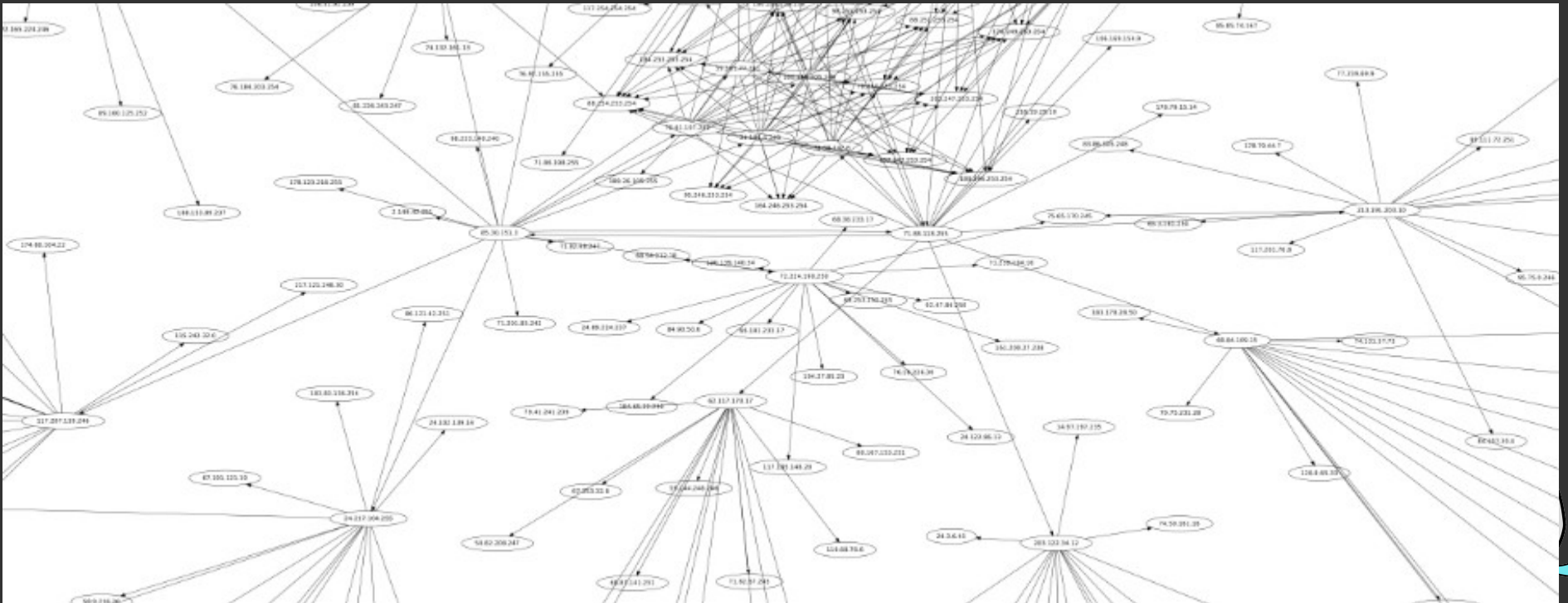
- Import samples of communications
- De-obfuscate exchanges
- Find similar messages
- Split messages in fields
- Abstract Fields
- Search for relations



DEMOS

Demo 2: retrieve the P2P zombie directory

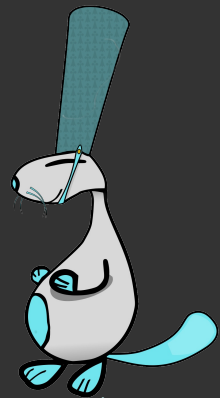
- Simulation of a realistic zombie
- Map the peers neighbours relations



Future improvements
of Netzob...

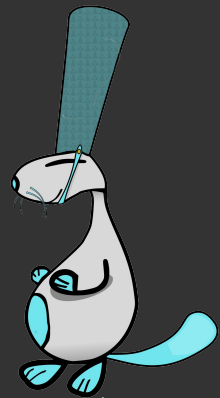
Integrated smart fuzzing, by leveraging the simulator engine

→ Allows to fuzz undocumented and proprietary protocols



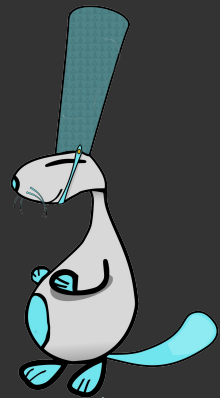
Integrated smart fuzzing, by leveraging the simulator engine

→ Allows to fuzz undocumented and proprietary protocols



Support of more communication channels

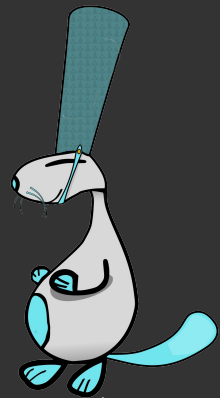
- ▶ USB
- ▶ IOCTL
- ▶ API (`ssl_read`, `ssl_write`, etc.)



Export protocol model in more 3rd party products *(coming soon)*

- ▶ Wireshark
- ▶ Scapy
- ▶ Peach Fuzzer

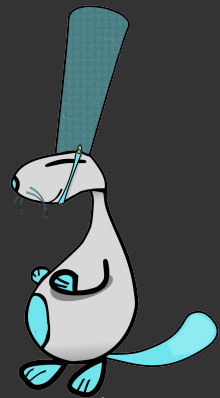
Allows protocol dissection with established tools



Export protocol model in more 3rd party products *(coming soon)*

- ▶ Wireshark
- ▶ Scapy
- ▶ Peach Fuzzer

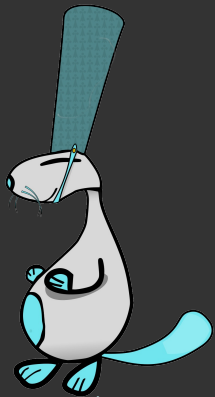
Allows fuzzing of unknown protocols with well-known tools



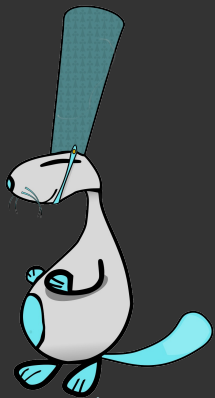
A black and white photograph of a long, empty, brightly lit tunnel or subway station. The floor is covered in a grid pattern of tiles. The ceiling is dark with several long, horizontal light fixtures. The walls are made of large, rectangular panels. A pigeon is visible on the floor in the lower right foreground. The word "Conclusion..." is written in white text across the center of the image.

Conclusion...

- ▶ Protocol RE automation domain is quite active at the academic level
- ▶ But **no real tool available...**
- ▶ Netzob tries to fill this lack by
 - ▶ Supporting academic researches
 - ▶ Being **usable in operational context**



- ▶ **Open** to all kind of **contributions**
 - ▶ Feedback
 - ▶ Bug fix
 - ▶ Feature proposal / implementation
 - ▶ Translation
 - ▶ ...



Thanks for you attention !

Any questions ?

www.netzob.org

@netzob

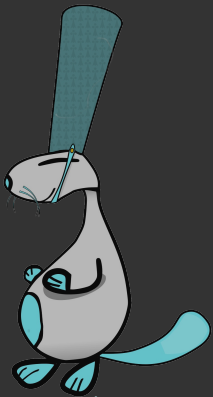
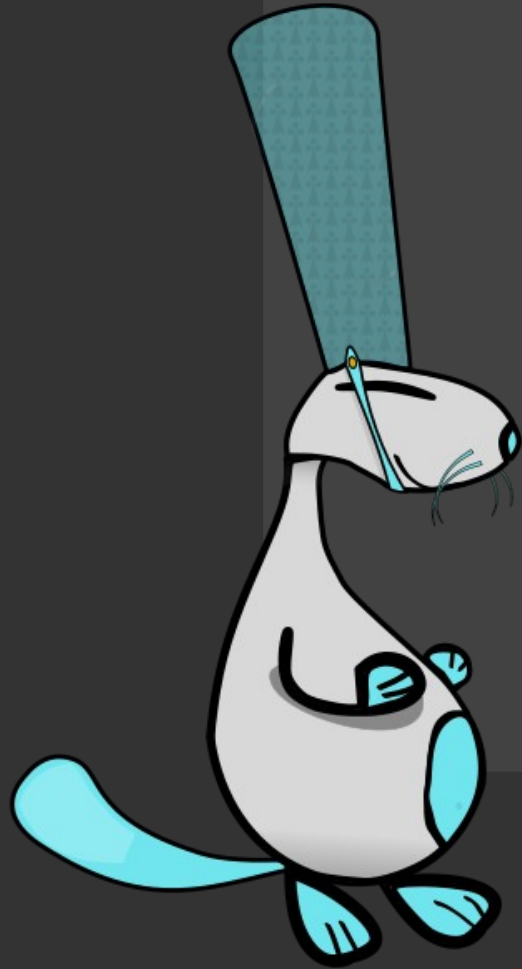


Image licences

<http://www.flickr.com/photos/arne-halvorsen/3346841686/in/pool-1093738@N24>
CC-BY-NC / aha42 | tehaha

<http://www.flickr.com/photos/torek/3280152297/sizes/l/in/pool-1093738@N24/>
CC-BY-ND ar kirainet

http://www.flickr.com/photos/jdawg/295956572/in/pool-security_theater/
CC-BY-NC r lawgeek

<http://www.flickr.com/photos/tjblackwell/7324060440/sizes/l/in/pool-36004471@N00/>
CC-BY-NC r tj.blackwell

<http://www.flickr.com/photos/sterlingely/1418364/sizes/z/in/pool-865293@N20/>
CC-BY-NC-SA DogFromSPACE

<http://www.flickr.com/photos/massalim/8110616773/in/pool-83823859@N00/>
CC-BY-SA И. Максим

<http://www.flickr.com/photos/davidjunyent/8170407965/sizes/l/in/pool-83823859@N00/>
CC-BY-NC-ND davidjunyent

<http://www.flickr.com/photos/omarparada/8165303605/sizes/l/in/pool-83823859@N00/>
CC-BY-NC-ND Omar Parada

<http://www.flickr.com/photos/tonirodrigo/8114182589/sizes/l/in/pool-83823859@N00/>
CC-BY ar Toni Rodrigo

<http://opte.org/maps/>
CC-BY-NC-SA The optet project

<http://www.flickr.com/photos/niznoz/63732753/lightbox/>
CC BY-NC-SA 2.0 by niznoz

