The Future of Protocol Reversing and Simulation Applied on ZeroAccess

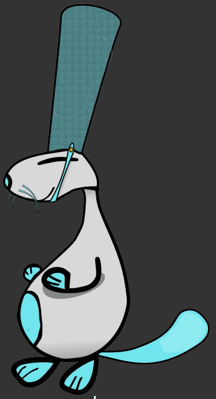
29C3, Hamburg

December 29 2012

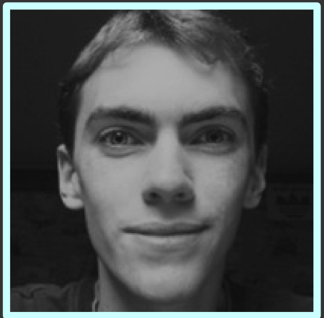
@Netzob

Authors...

« You talkin' to me? »
Travis Bickle

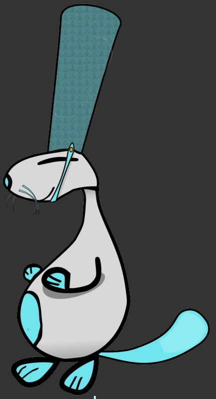



Yes, we're French !



Frédéric GUIHERY (@_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*

AMOSSYS, France

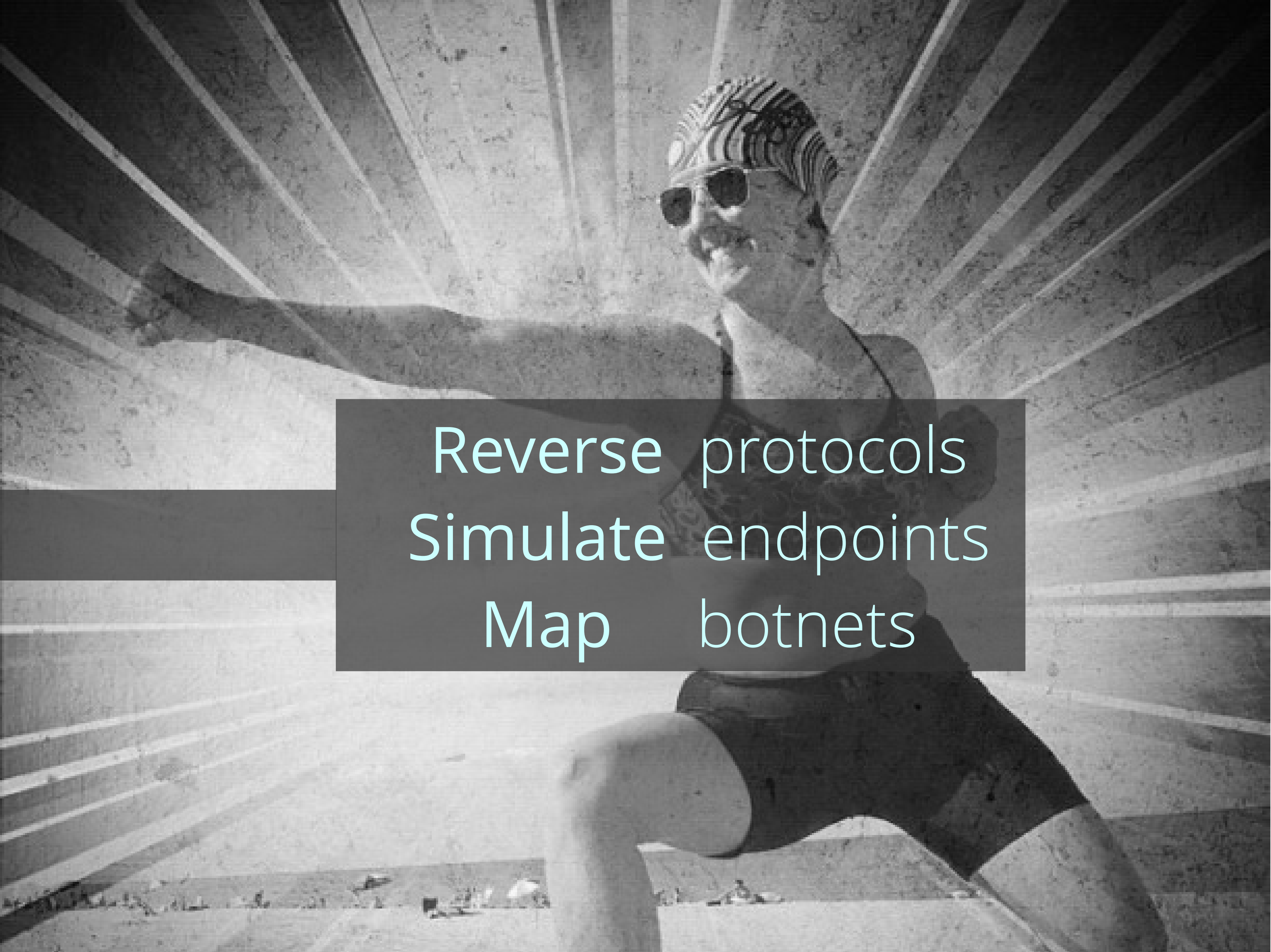
- Audit and evaluation
  - ITSEF lab (Common Criteria, CSPNs, ...)
  - Pentest lab
- R&D

www.amossys.fr

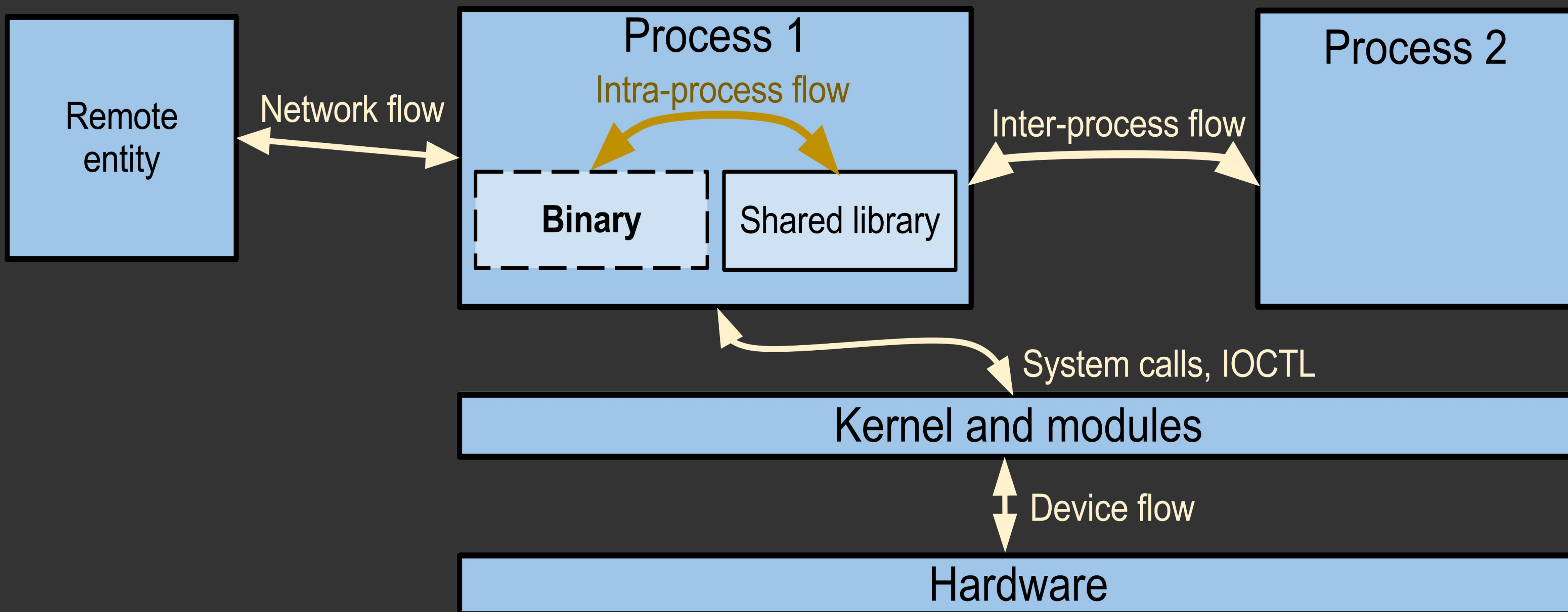
Topics...

« Go ahead, make my day »

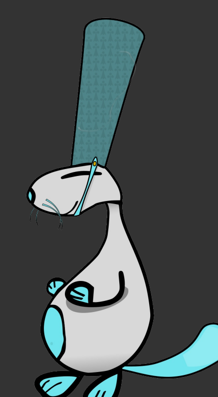
Harry Callahan

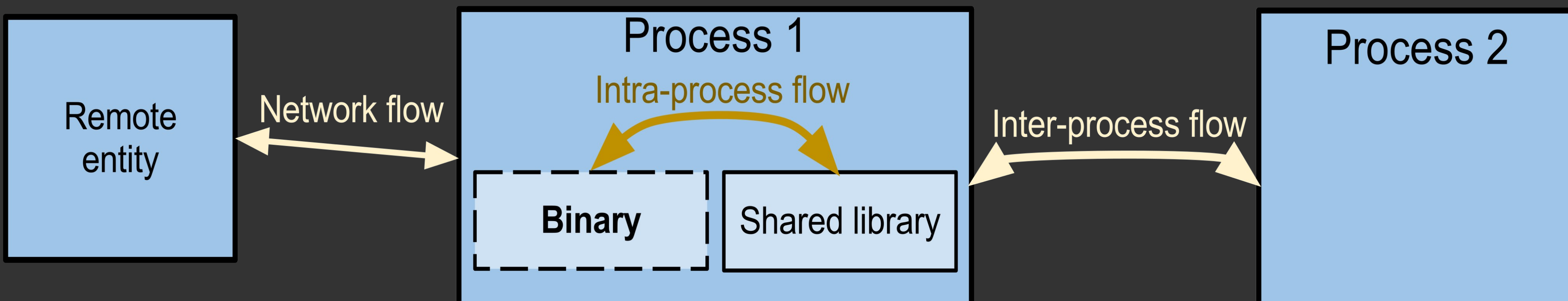

Reverse protocols
Simulate endpoints
Map botnets

Why reverse engineering of protocols ?



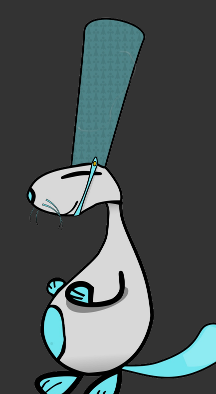
Protocols are everywhere





Protocols are everywhere

(even within Tamagotchis)

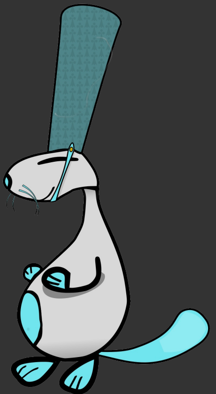


Assess the robustness of implementations

- 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 ?

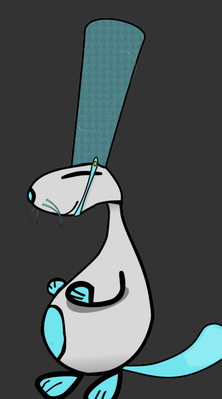
Compare the implementation of a protocol with its official specifications

- 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 ?

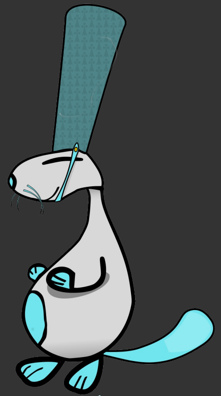
Did it looked like this ?

- Complex
  - Time-consuming
- Mostly Manual

Did it looked like this ?

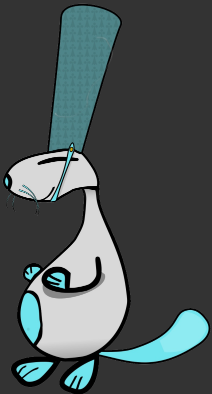
- Complex
  - Time-consuming
- Mostly Manual

MOSTLY VISUAL




No available tool to **reverse** a **proprietary** protocol...
Should we create one?

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

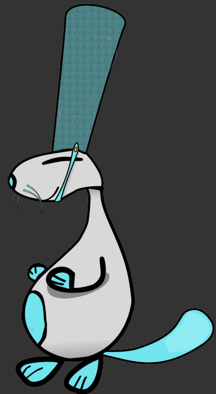


Some reminders about **protocols**

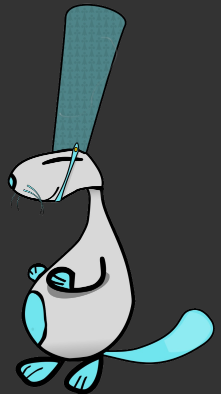
Let's examine the TCP protocol

Different types of messages

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



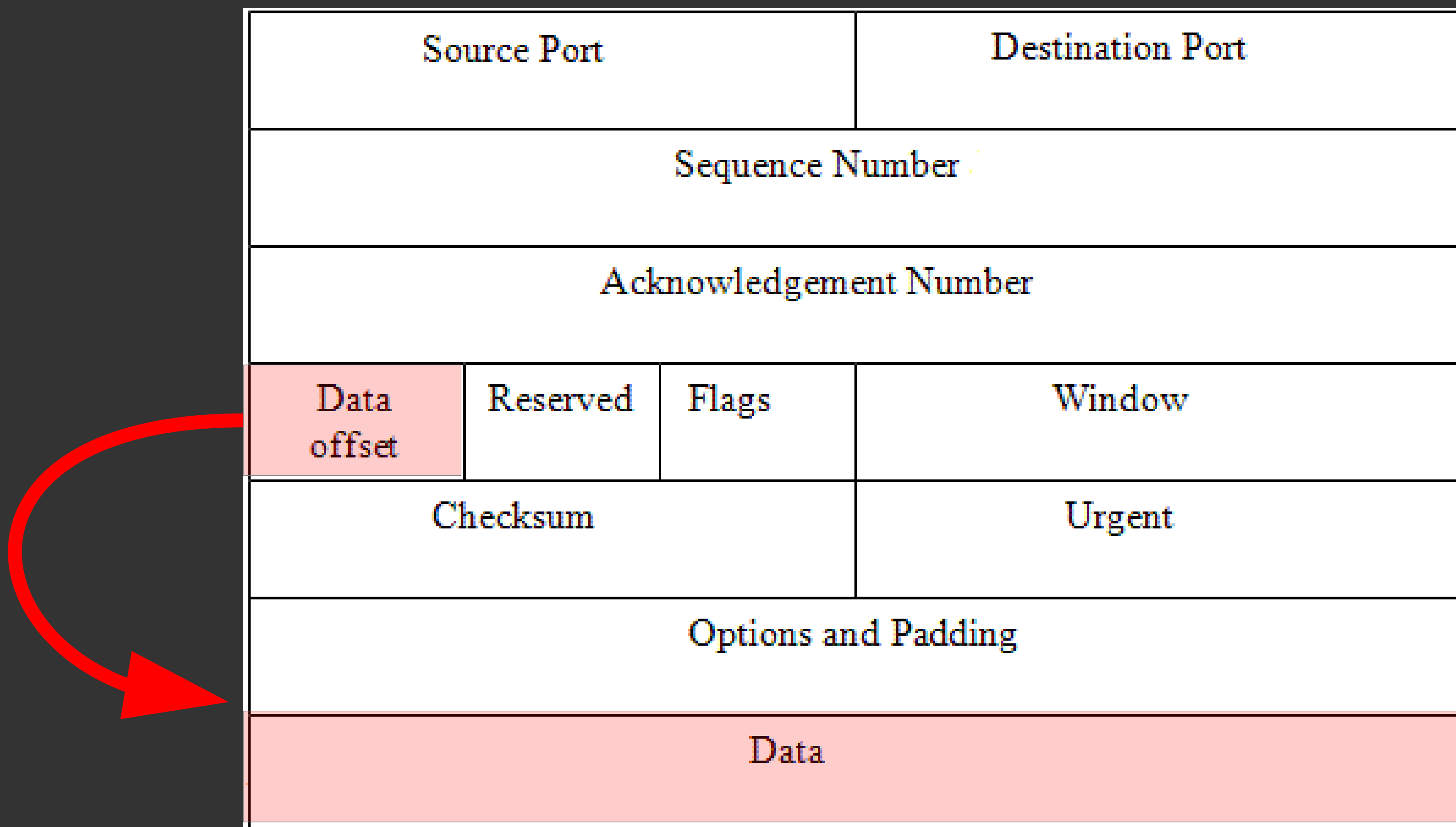
Concept of encapsulation layers



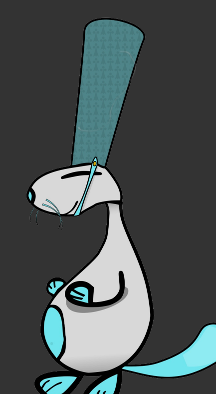
Fields partitioning

<table>
<tr><td colspan="3">Source Port</td><td>Destination Port</td></tr>
<tr><td colspan="4">Sequence Number</td></tr>
<tr><td colspan="4">Acknowledgement Number</td></tr>
<tr><td>Data offset</td><td>Reserved</td><td>Flags</td><td>Window</td></tr>
<tr><td colspan="3">Checksum</td><td>Urgent</td></tr>
<tr><td colspan="4">Options and Padding</td></tr>
<tr><td colspan="4">Data</td></tr>
</table>

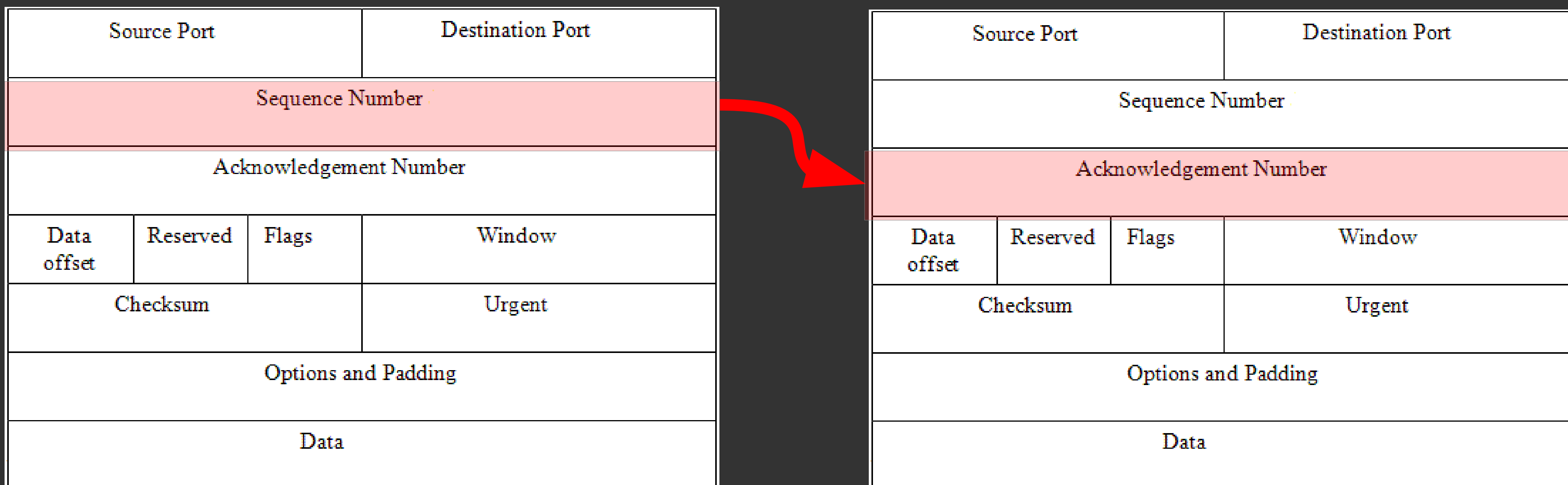
Relations



Intra-message relations



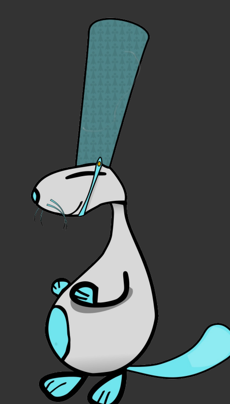
Relations



Inter-message relations

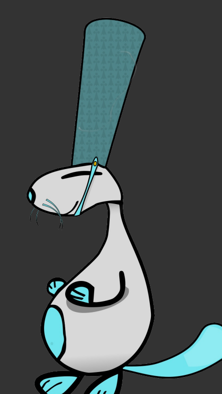
Contextual values

<table>
<tr><td colspan="3">Source Port</td><td>Destination Port</td></tr>
<tr><td colspan="4">Sequence Number</td></tr>
<tr><td colspan="4">Acknowledgement Number</td></tr>
<tr><td>Data offset</td><td>Reserved</td><td>Flags</td><td>Window</td></tr>
<tr><td colspan="3">Checksum</td><td>Urgent</td></tr>
<tr><td colspan="4">Options and Padding</td></tr>
<tr><td colspan="4">Data</td></tr>
</table>

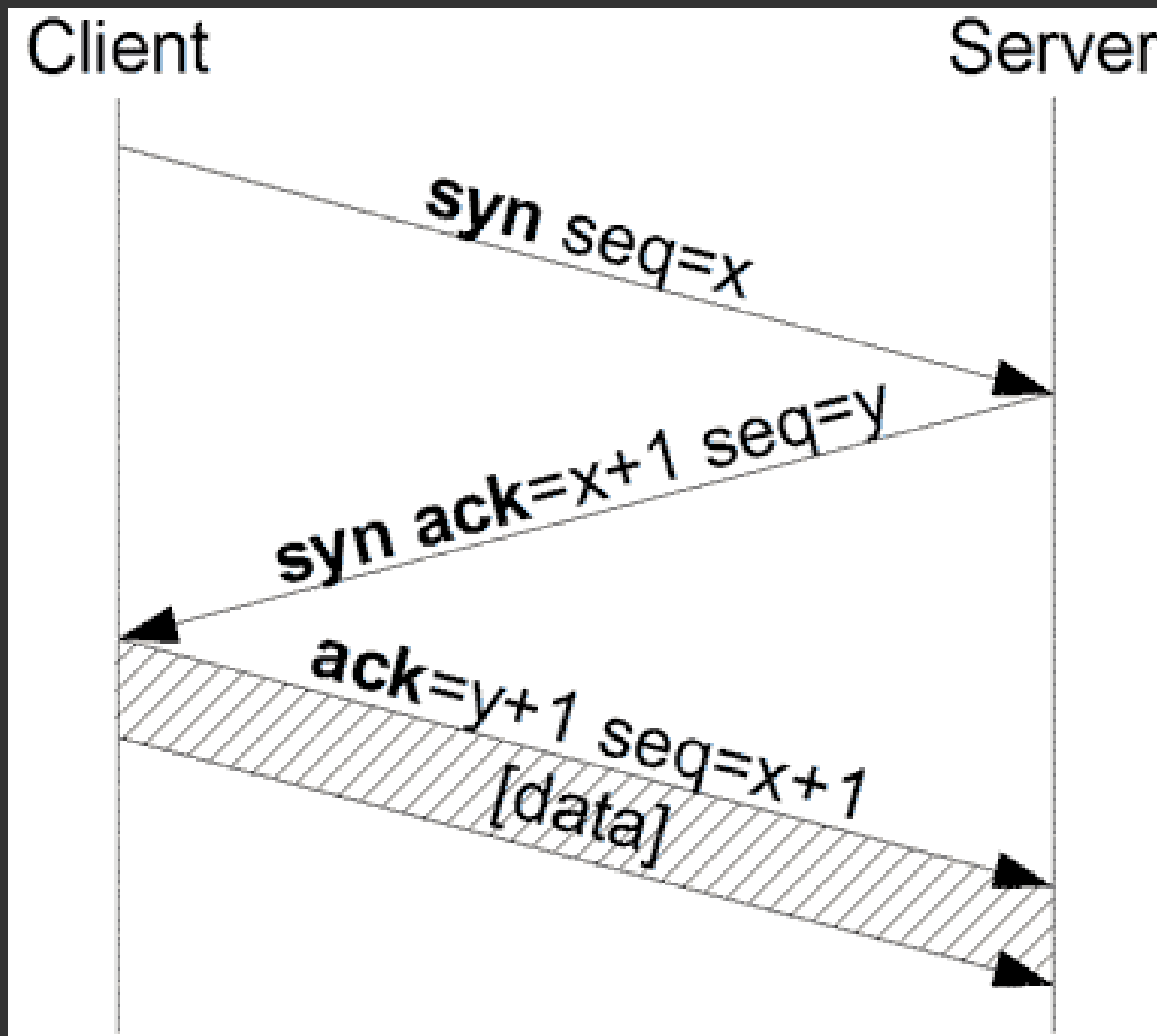


Application-level values

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



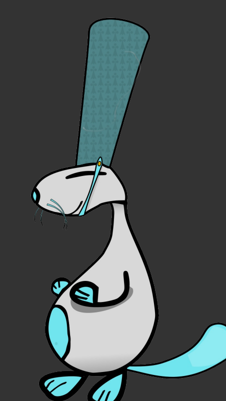
Sequence of valid messages




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

Academics are very good with * models :)

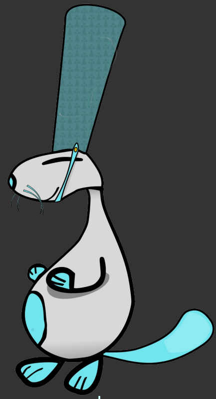
** sometimes useless*



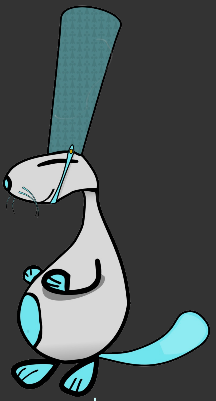
« *Design and Validation of Computer Protocols* »

by G. Holzmann

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



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

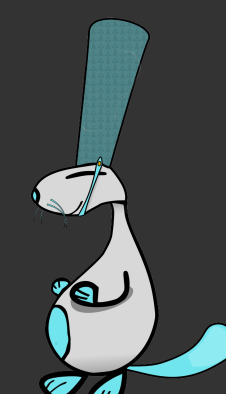
(4/5)

the **procedure rules** (5/5)

Yes, that was an academic model

Reduced model for a Protocol

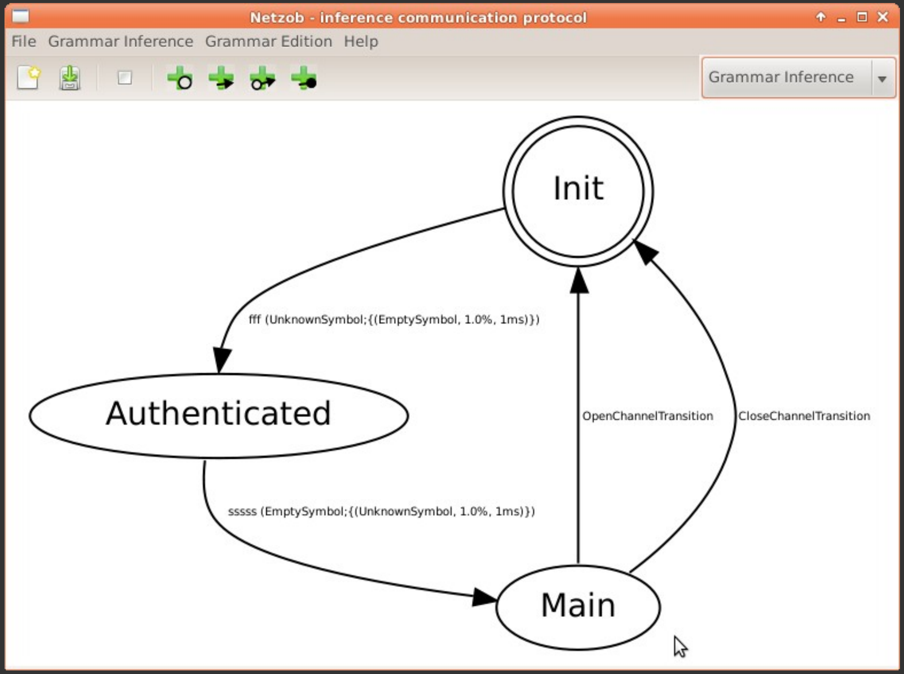
- a vocabulary → a list of **Message Format**
- a grammar → **State Machine**

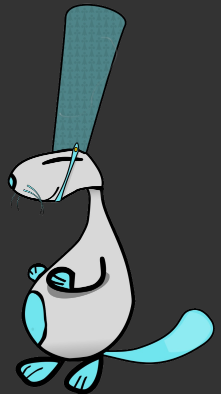



Introducing Netzob ...

Goals of Netzob

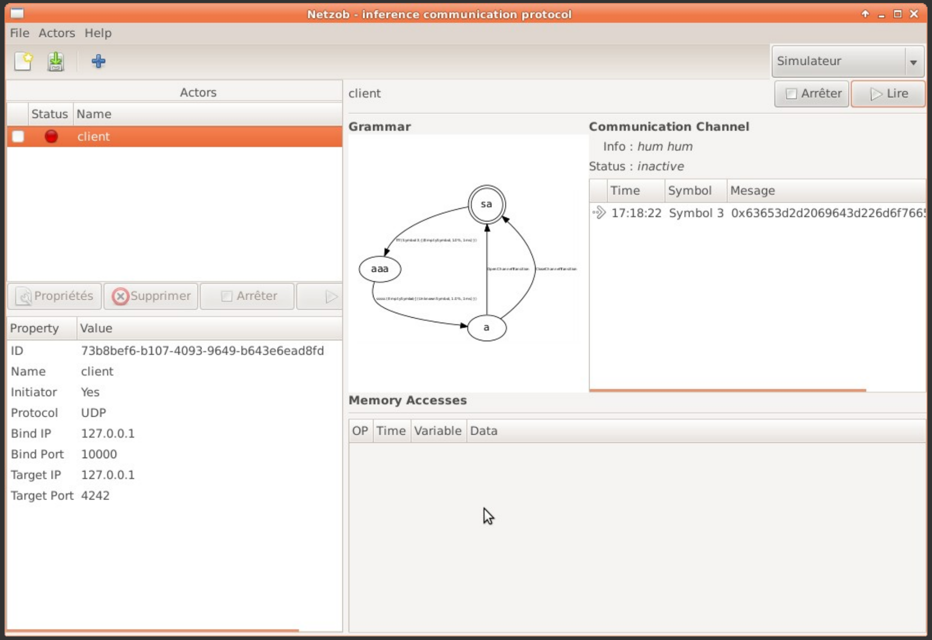
- Infer unknown (proprietary) protocols
- Simulate actors of a communication
- Smart-Fuzz targeted implementations

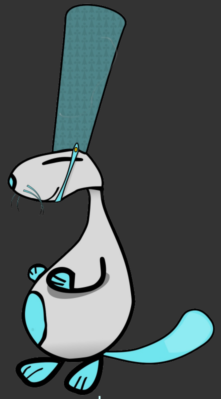




Goals of Netzob

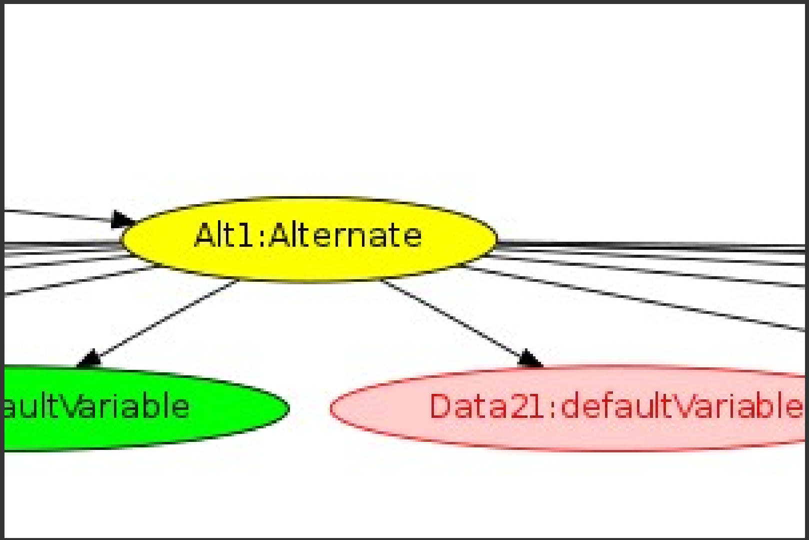
- Infer unknown (proprietary) protocols
- Simulate actors of a communication
- Smart-Fuzz targeted implementations

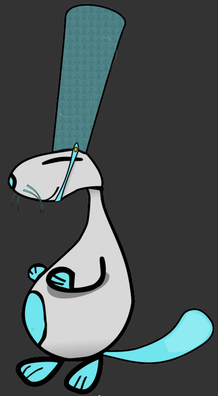




Goals of Netzob

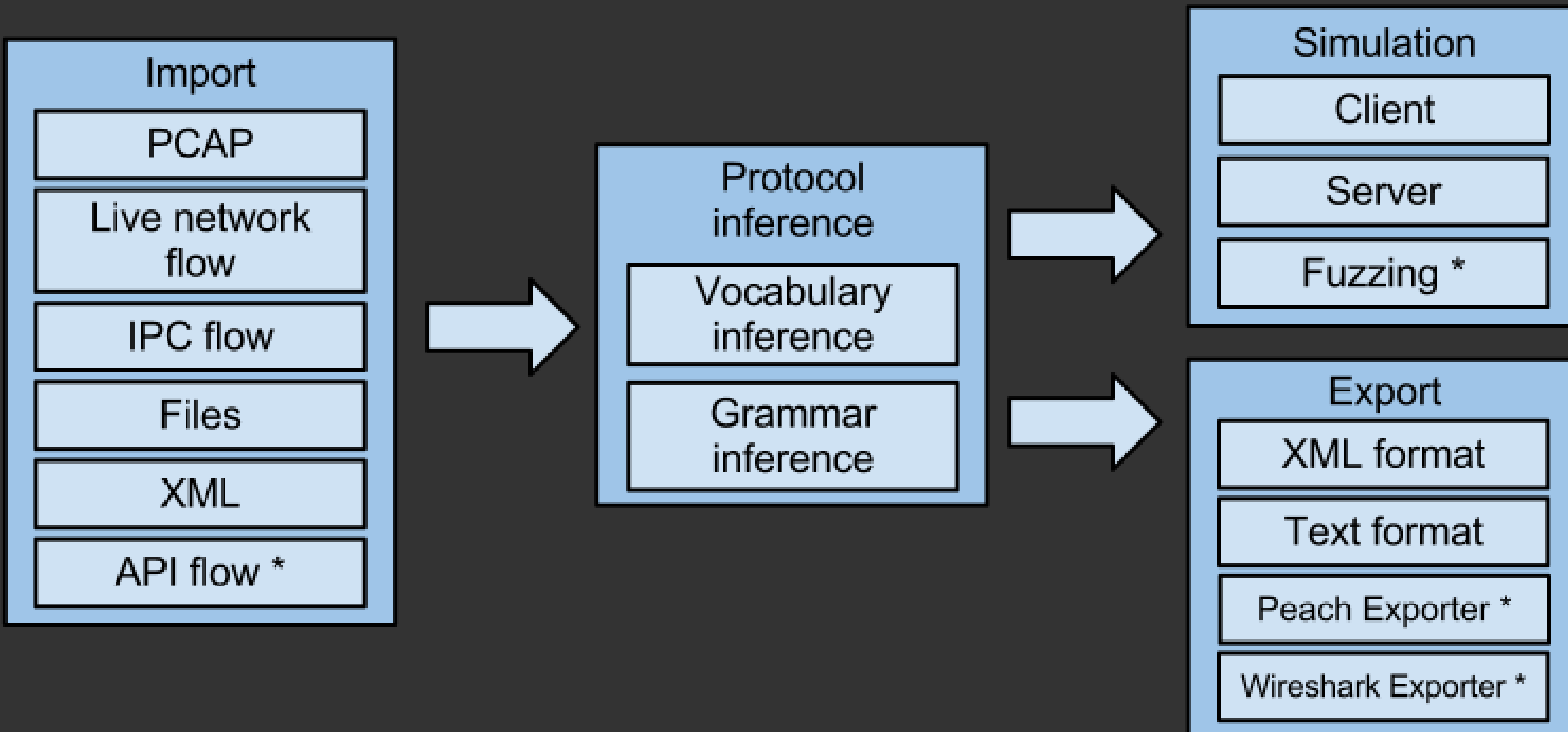
- Infer unknown (proprietary) protocols
- Simulate actors of a communication
- Smart-fuzz targeted implementations

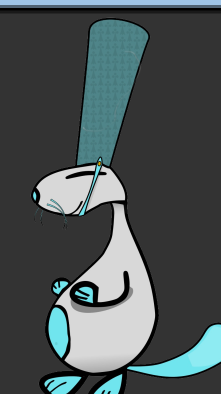


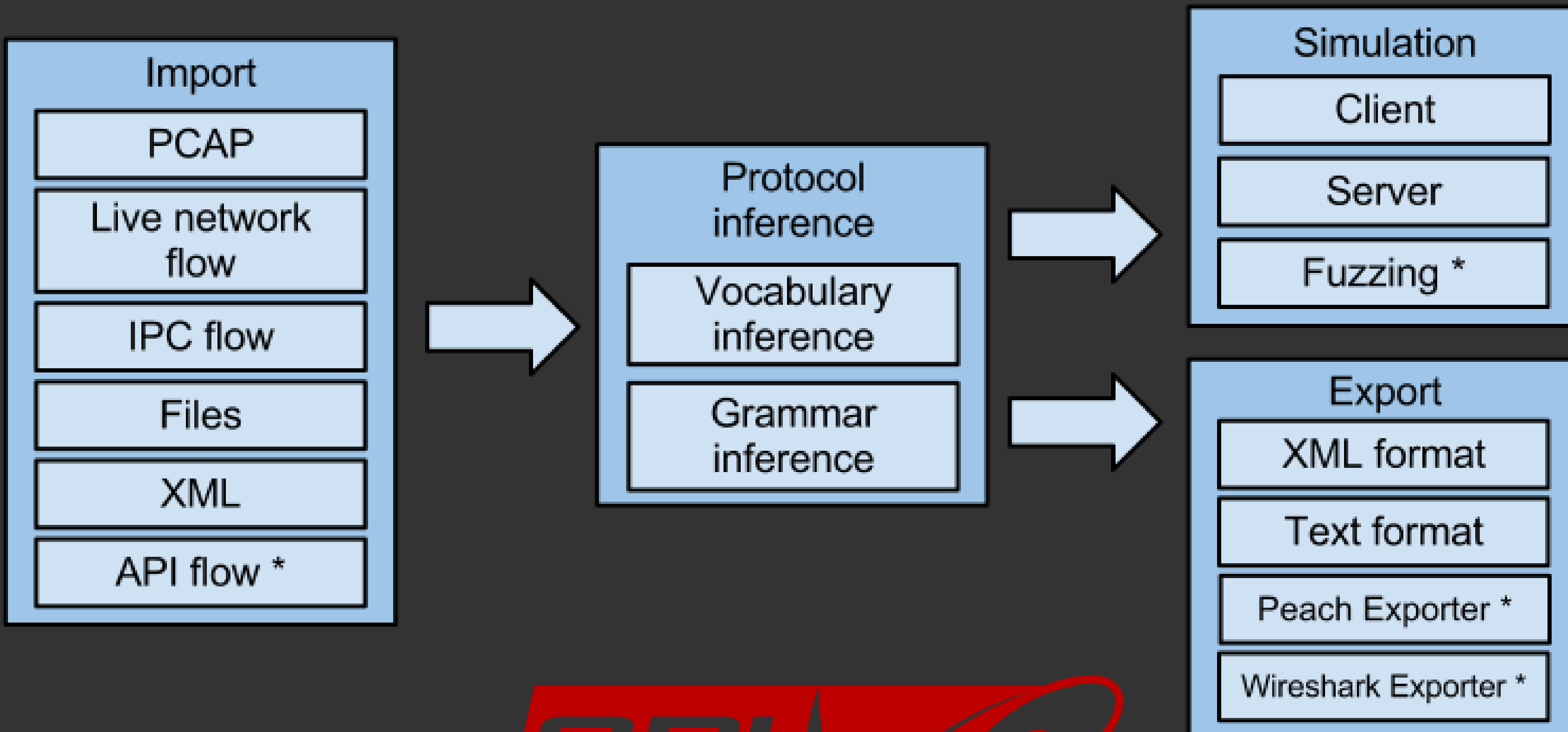


Approach taken by Netzob

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

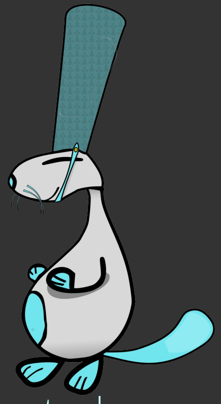

Import
PCAP
Live network flow
IPC flow
Files
XML
API flow *
Protocol inference
Vocabulary inference
Grammar inference
Simulation
Client
Server
Fuzzing *
Export
XML format
Text format
Peach Exporter *
Wireshark Exporter *




Import
PCAP
Live network flow
IPC flow
Files
XML
API flow *
Protocol inference
Vocabulary inference
Grammar inference
Simulation
Client
Server
Fuzzing *
Export
XML format
Text format
Peach Exporter *
Wireshark Exporter *

Netzob implementation

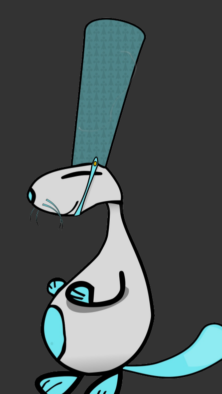
- Graphical interface (GTK3)
- Mostly written in Python and C (46 000 LOC)
- Plugin architecture
- Available through
  - GIT Repositories
  - Python package
  - Per-Os packages (Debian, Ubuntu, Gento, ...)

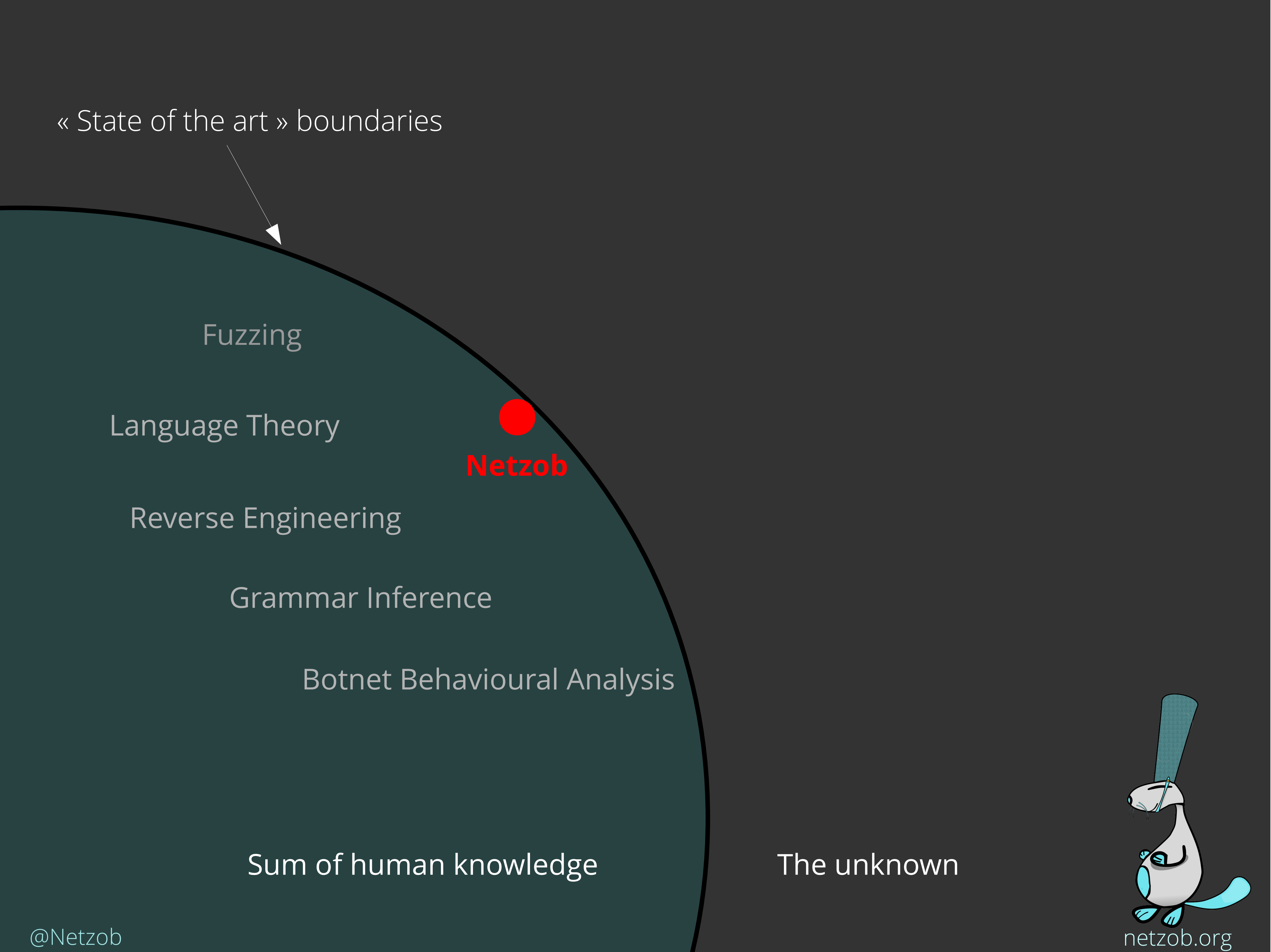


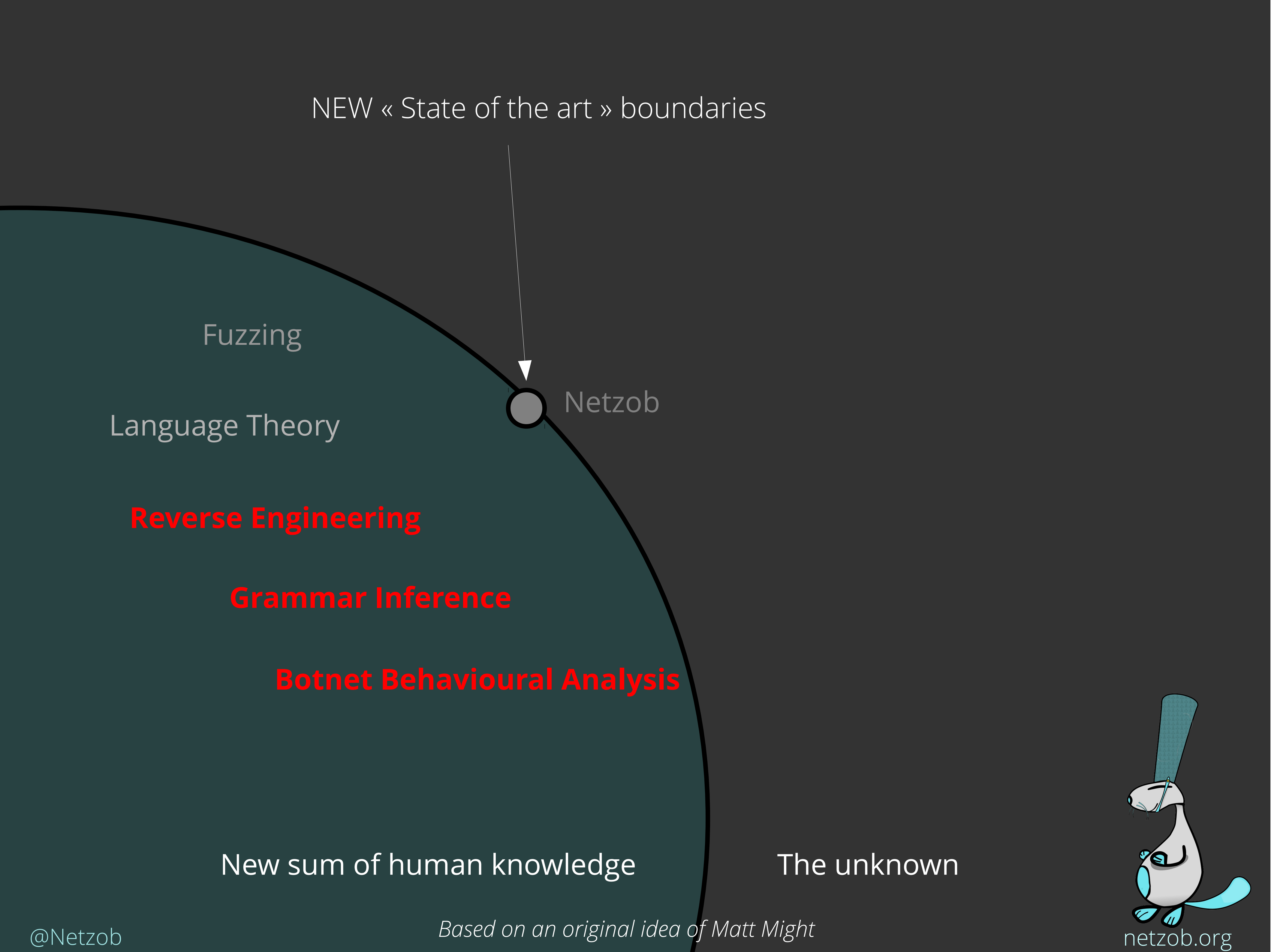
Netzob team (chronologicaly ordered)

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

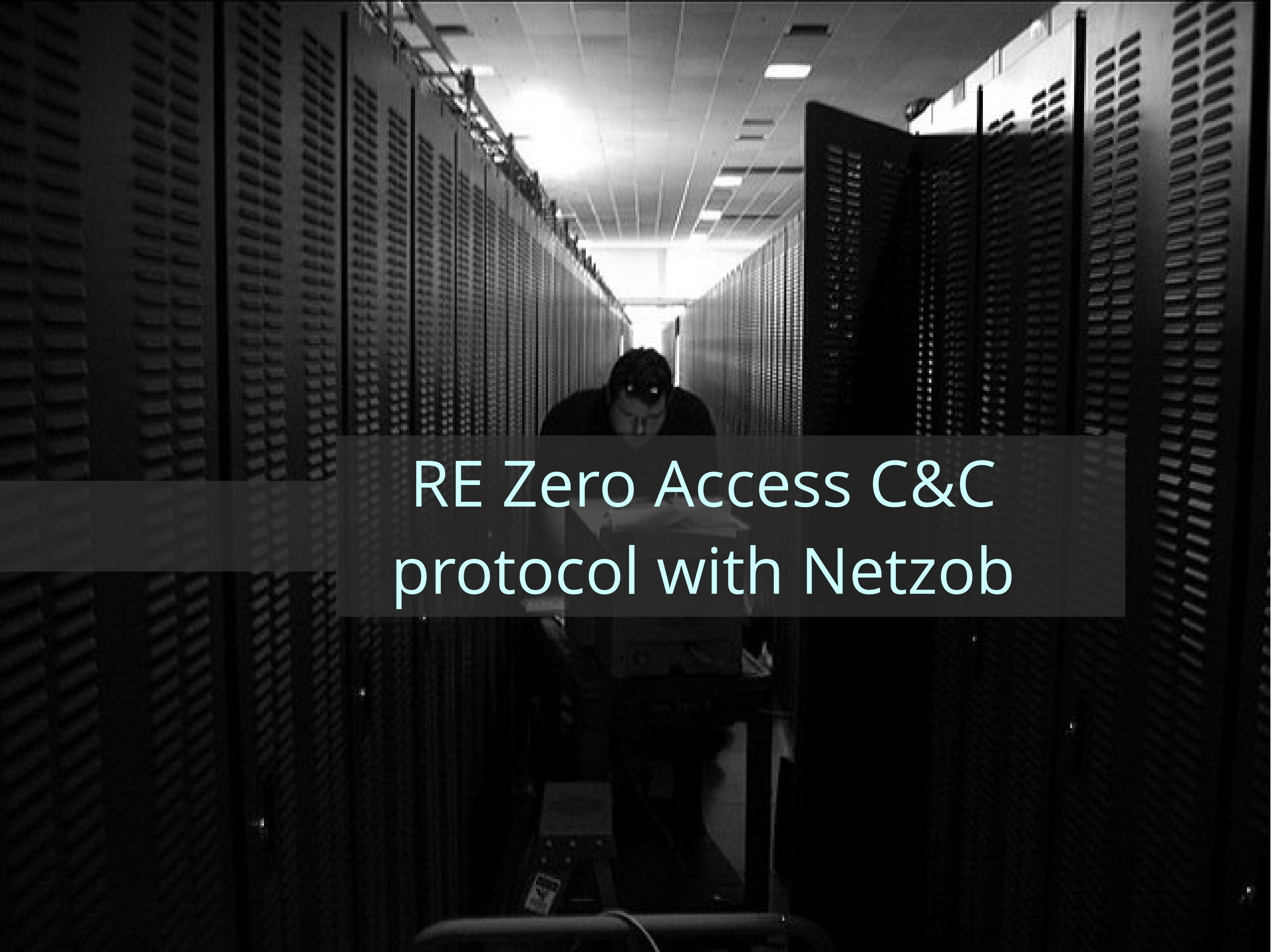
Netzob's Sponsors




« State of the art » boundaries
Fuzzing
Language Theory
Netzob
Reverse Engineering
Grammar Inference
Botnet Behavioural Analysis
Sum of human knowledge
The unknown
@Netzob
netzob.org


NEW « State of the art » boundaries
Fuzzing
Netzob
Language Theory
Reverse Engineering
Grammar Inference
Botnet Behavioural Analysis
New sum of human knowledge
The unknown
@Netzob
Based on an original idea of Matt Might
netzob.org

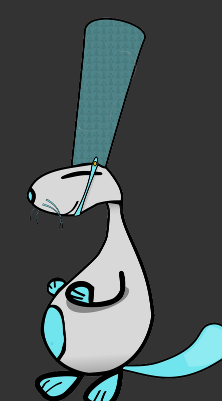
RE Zero Access C&C protocol with Netzob



Zero Access (aka Sirefef)

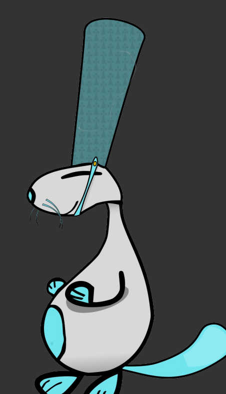
- Recent botnet (Sept. 2011)
  - still in activity
- +/- 1 million zombies (9 millions installed)
- Click fraud and bitcoin miner
- At least 2 versions of the rootkit
  - Upgraded P2P protocol

Based on Sophos and Kindsight Reports



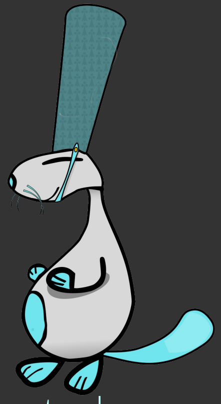
Zero Access (aka Sirefef)

- 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



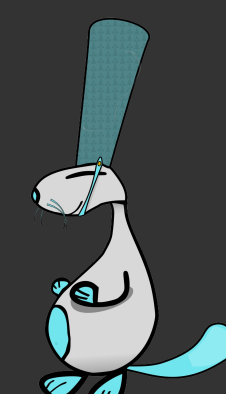
Few real communication traces

ZAccess : some traces were provided by

Kevin McNamee

an infected machine

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



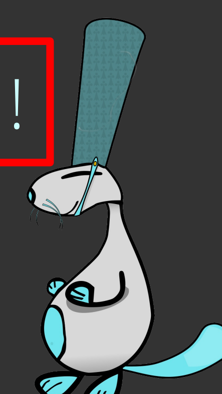
A confined environment and the binary

Adapted Virtual Machines + Firewalls + Torify + management system

A confined environment and the binary

Adapted Virtual Machines + Firewalls + Torify + management system

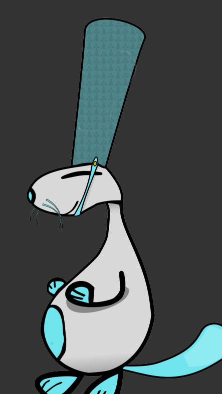
Warning : Consider legal issues before dealing with this !



Step 1 : Get messages

Capture **dataflows**

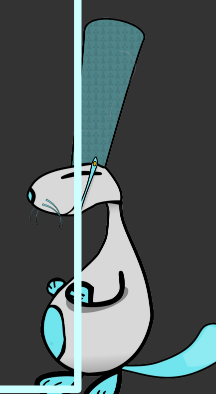
(Network, USB, IPC, API Hooking, Raw files, ...)



Capture dataflows

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

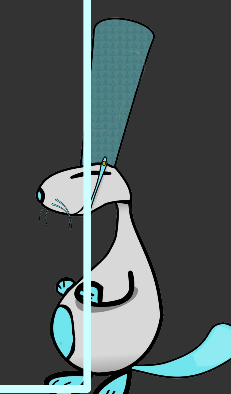
?..b(.......@.p...aU(..........3.Mi.L..."..f3.8:+.6..d...x,'3...>4...L......:3.8;..q..d...Q..3........
L.QsY.8:3..m.0...d.E....3.Zj."...L.F...8:3-.>ad...s
.....3%.GgM....|..C.8:...}...o......(t0...iW......oJ..`?...~.!c.p./e..z......%gI.T.s.mE..+-..;
R.{eoDz.}At..d.QP3.SX=).:.JU.7`..t...g.V..5.2.......OW..f2p.X8(......r..7.......S..&8...!..
'.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......%gI.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.

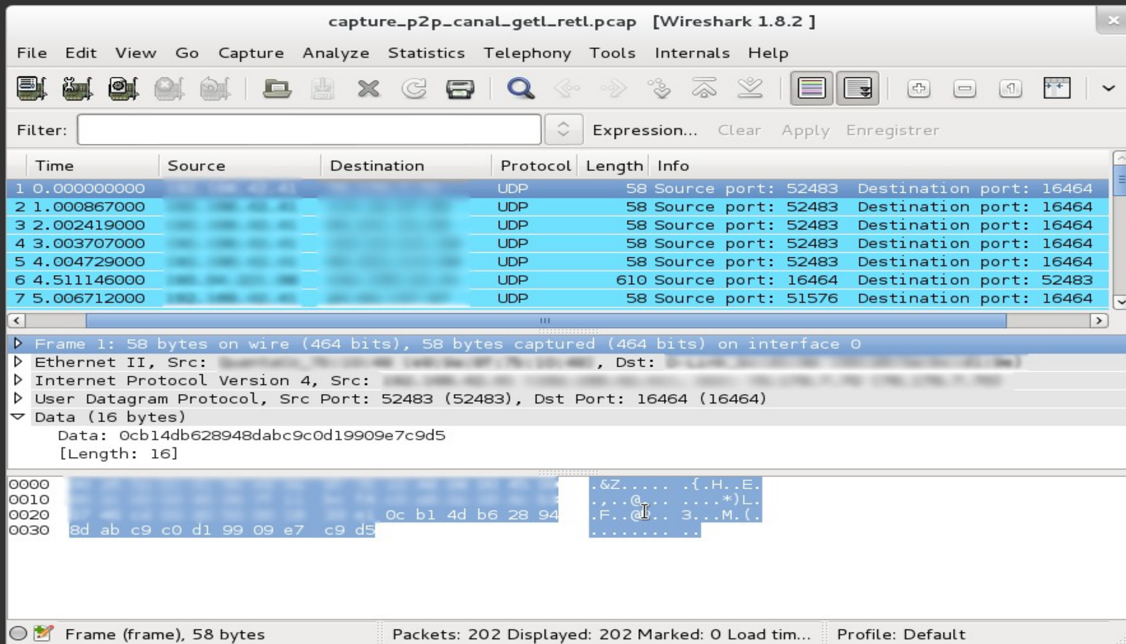


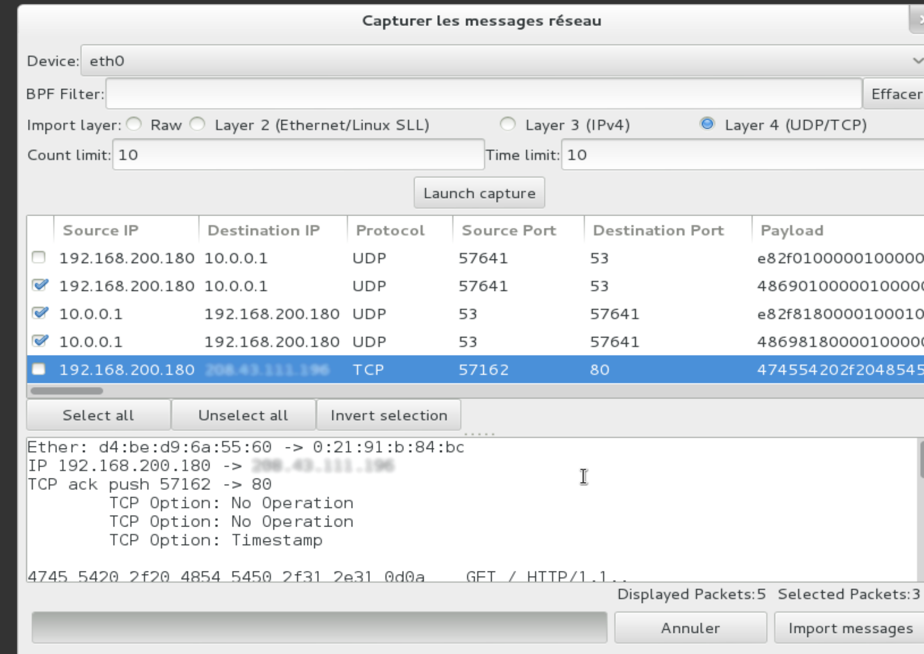
Split dataflows in messages

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

?..b(.......@.p...aU(..........3.Mi.L..."..f3.8:+.6..d...x,'3...>4...L......:3.8;..q..d...Q..3........
L.O..Y.2.2......0.....d.E....3.Zj."...L.F...8:3-.>ad...s
Message 1
.......................8:...}...o......(t0...iW......oJ.. `?...~.!c.p./e..z......%gI.T.s.mE..+-..;
R.{eoDz.}At..d.QP3.SX=).:.JU.7` ..t...g.V..5
Message 2
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.
Message 3
8:3.......................L...k.8:3}.e.d..........3%.GgM....|..C.8:...}...o......(t0...iW......oJ..
`?...~.!c.p./e..z......%gI.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.{....;XV
Message 4
....=.O..?... Qpga("...;...^:.....LU..>.'.i.1
L8.7c....@........}1iN.7...25t.G.)53.,.......S......9.'.L5.J.







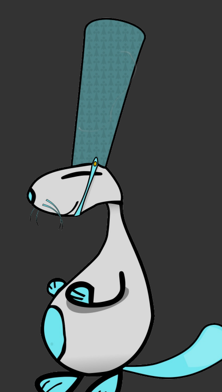
Import

Capture

Netzob framework

Filter imported messages

Choose layer of import



@Netzob

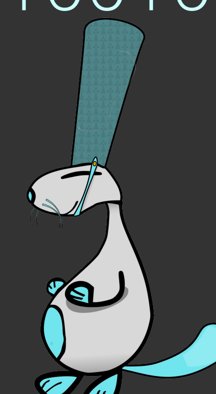
netzob.org

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**100100010101011010010101**1101010001010010

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
- Delimitor-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
- Delimitor-based Alignment
- Sequence Alignment

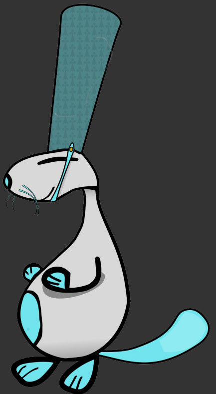
fa0	4c	5519e43e2e27fa6916313dc89ac77569a00d	4c	38f
fa0	4c	5519e43e2e27fa6916313dc89ac77569a00d	4c	38f
fa0	4c	5519e43e2e27fa6916313dc89ac77569a00d	4c	38f

Messages are splitted in **Fields** using

- Simple Alignment
- Delimitor-based Alignment
- Sequence Alignment

Needleman & Wunsch

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



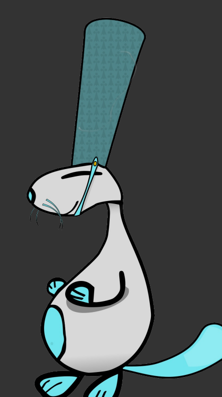
But WTF is Needleman & Wunsch ?

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

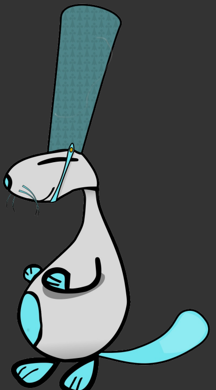
70 c4 00 00



Sequence alignment with Needleman-Wunsh

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

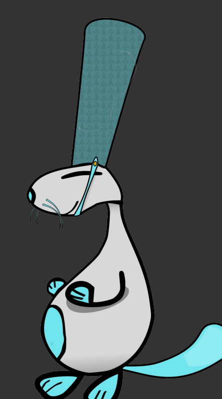
We build a distance matrix



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

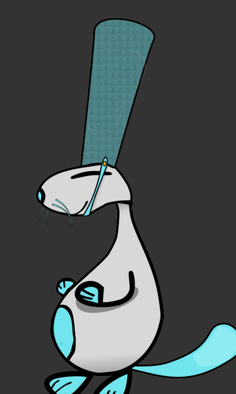
		70	83	2f	65	bd	86	7a	d2	00
	0	0	0	0	0	0	0	0	0	0
70	0	?								
c4	0									
00	0									
00	0									

We fill the matrix with the formula:

$$M(i,j) = Max(M(i-1, j-1) + S, M(i, j-1) + W, M(i-1, j) + W)$$

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									

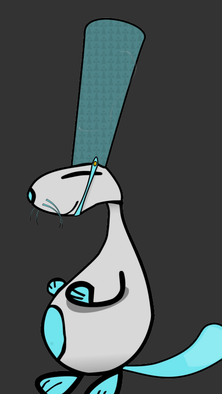
We fill the matrix with the formula:

$$M(i,j) = Max(M(i-1, j-1) + S, M(i, j-1) + W, M(i-1, j) + W)$$

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**

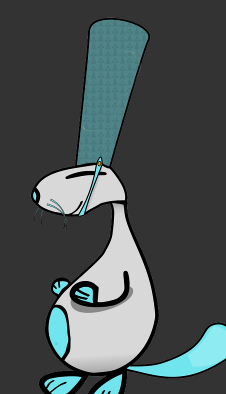
We fill the matrix



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



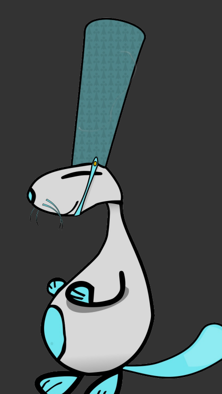
Sequence alignment with Needleman-Wunsh

		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**



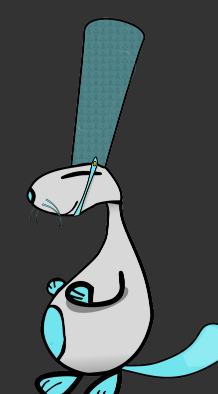
Sequence alignment with Needleman-Wunsh

We finally build a regex

70 83 2f 65 bd 86 7a d2 00

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

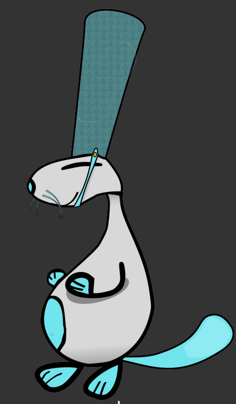
(70) (.*{2,7}) (00)

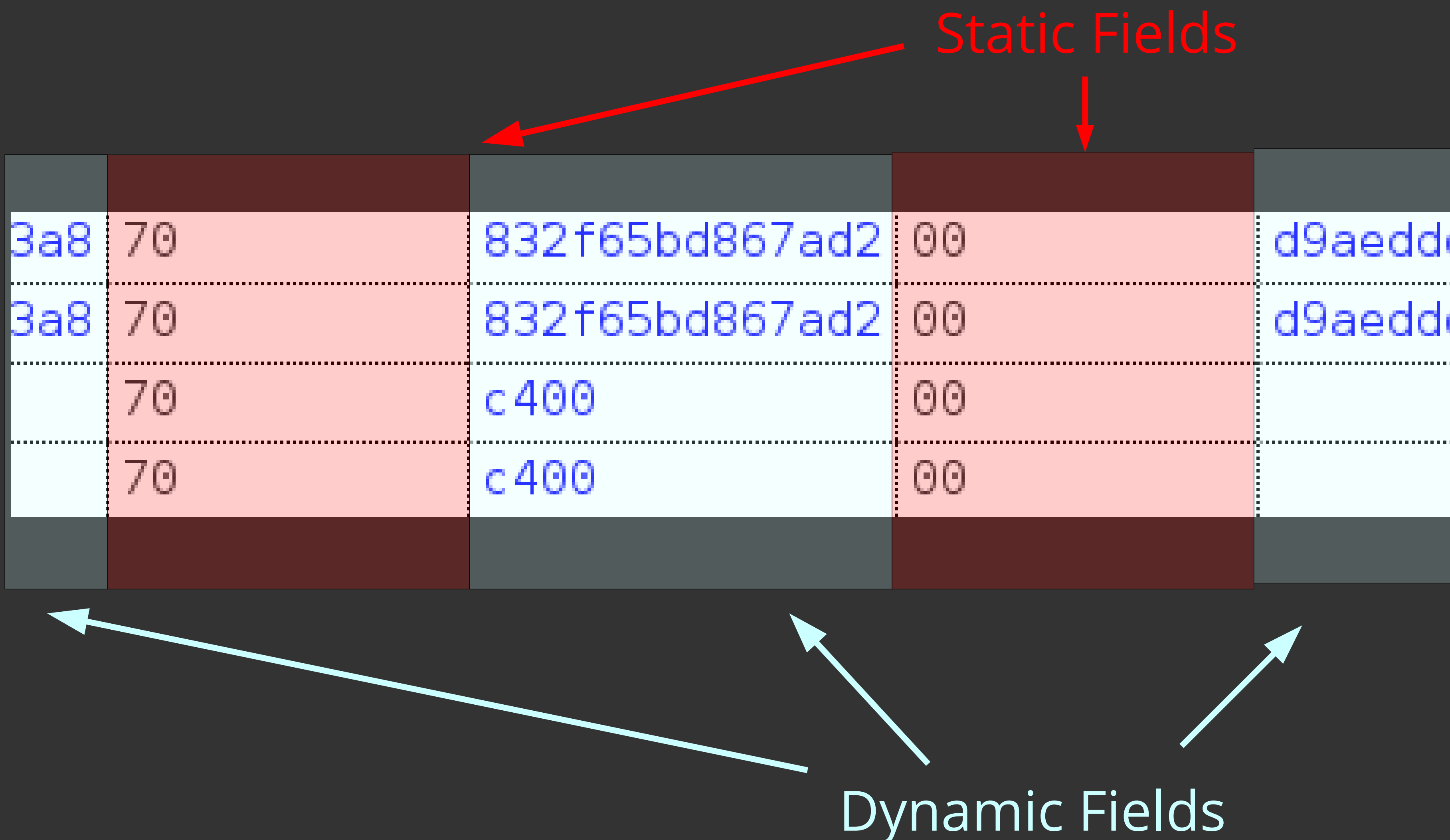


Static Fields

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

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





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

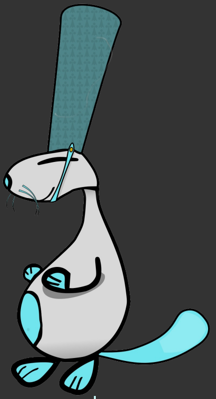
How to measure **similarity** between two messages ?

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

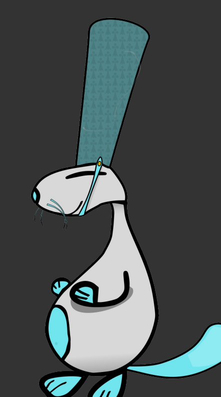
Similarity scores between messages

- 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)



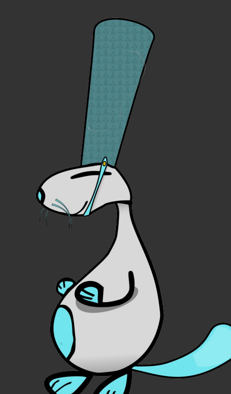
Similarity scores between messages

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

70 83 2f 65 bd 86 7a d2 00

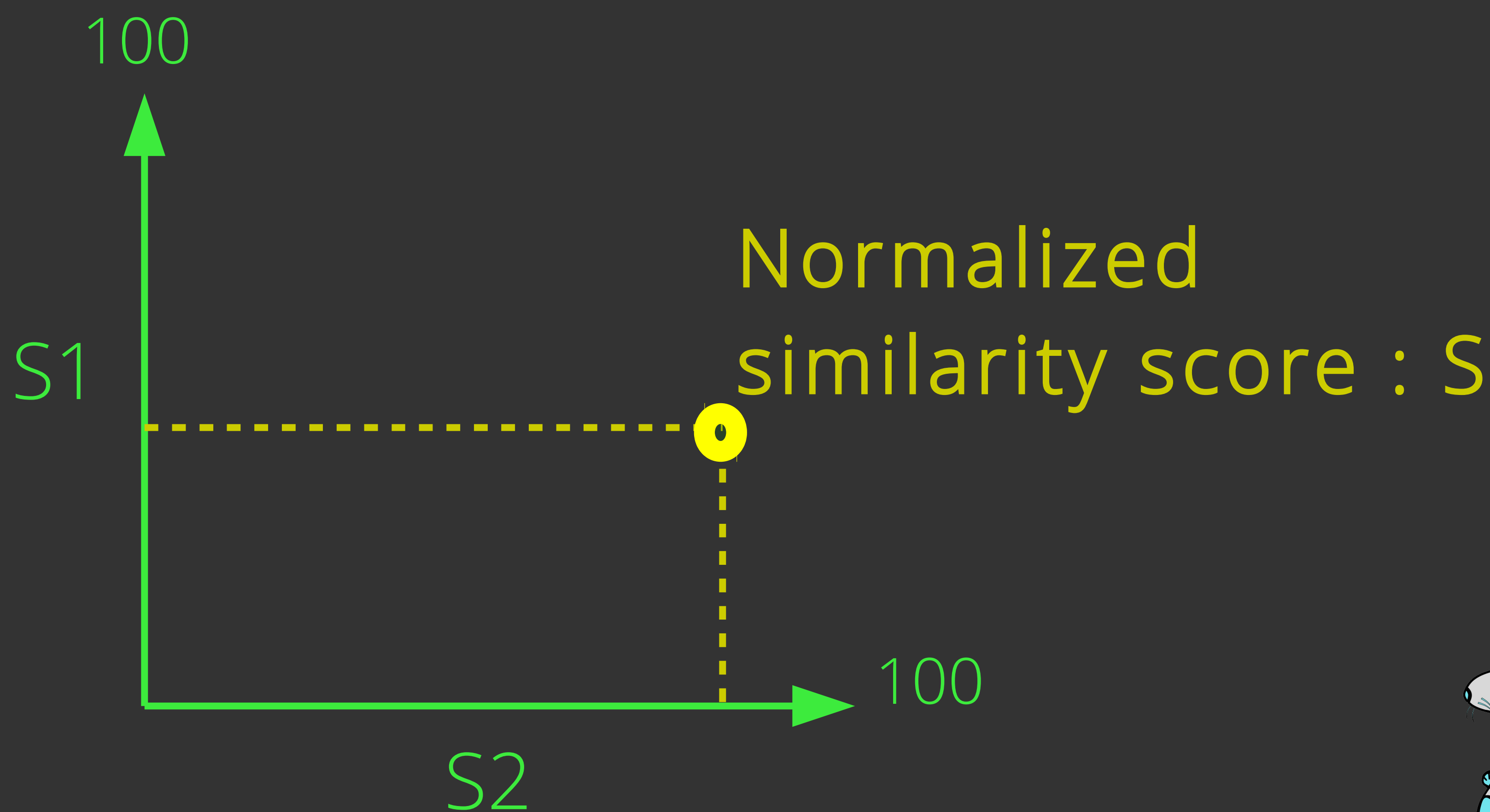
70 c4 00 -- -- -- -- -- 00

S2 = 2 / 7



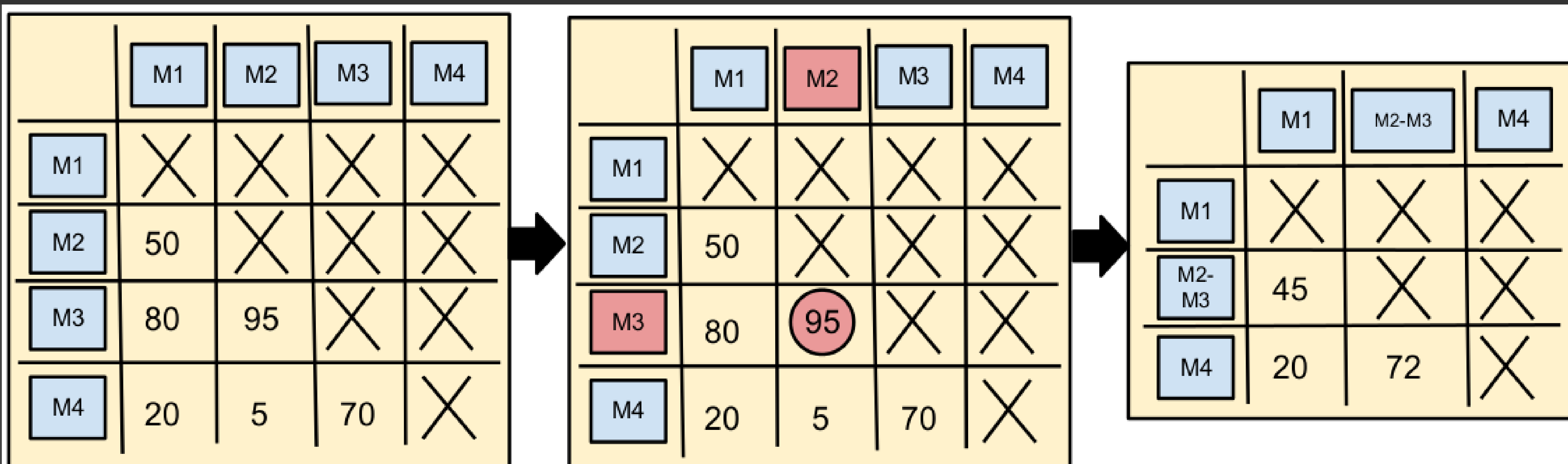
Similarity scores between messages

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

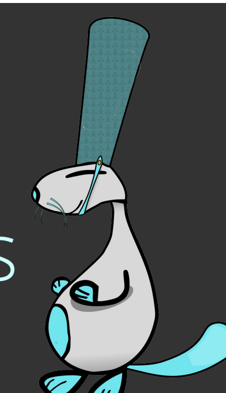


How to retrieve **groups** of similar messages ?

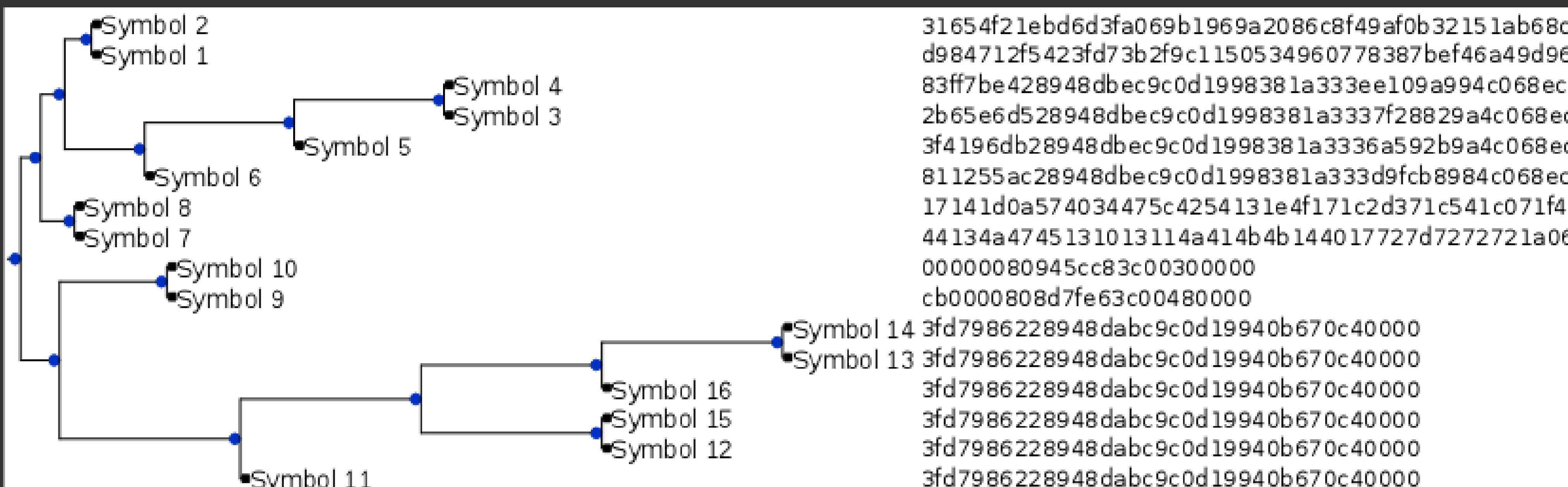
Hierarchical Clustering by similarities:
Similarity matrix
UPGMA

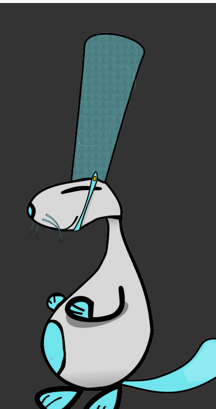


- Filling of a similarity matrix
- Iteratively merge the 2 most similar messages

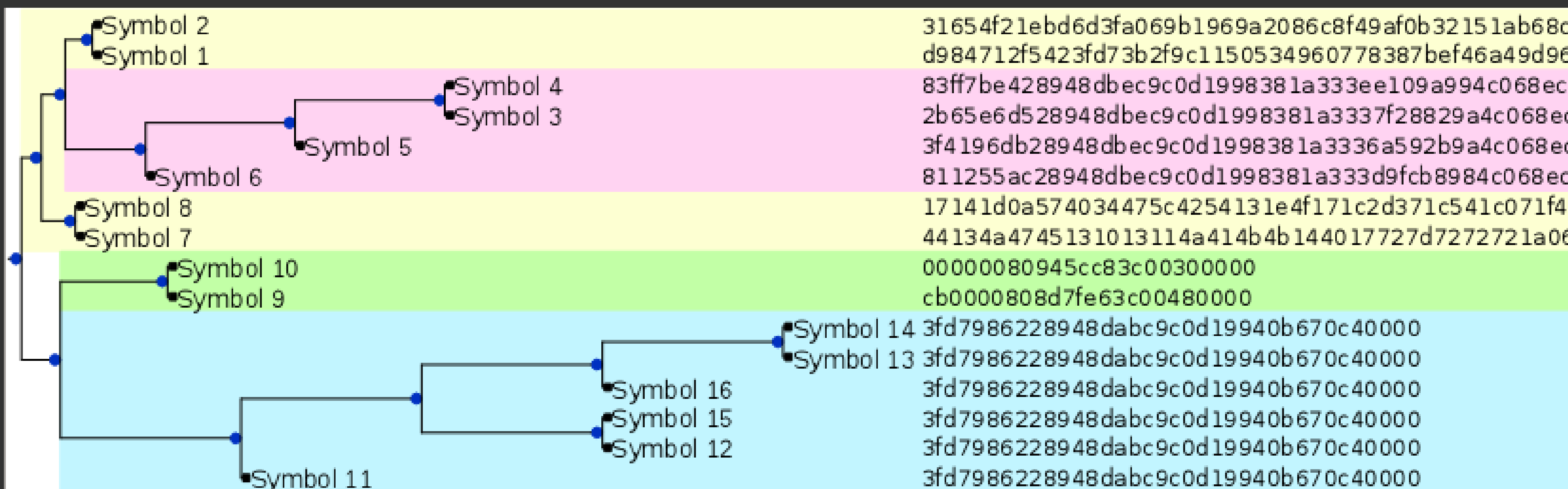


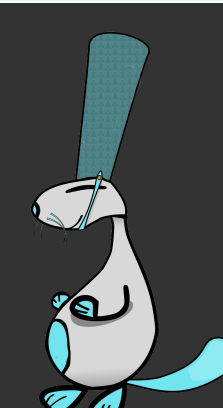
UPGMA creates a similarity tree





UPGMA creates a similarity tree and facilitates clustering





@Netzob

netzob.org

ZAccess Example

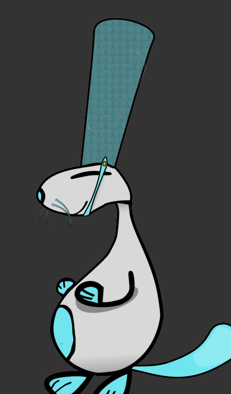
Results of Clustering and Sequence Alignment

Symboles

Nom	Message	Champ
Symbol 10	5	3
Symbol 25	8	3
Symbol 23	10	5
Symbol 5	8	12

Symbol 5

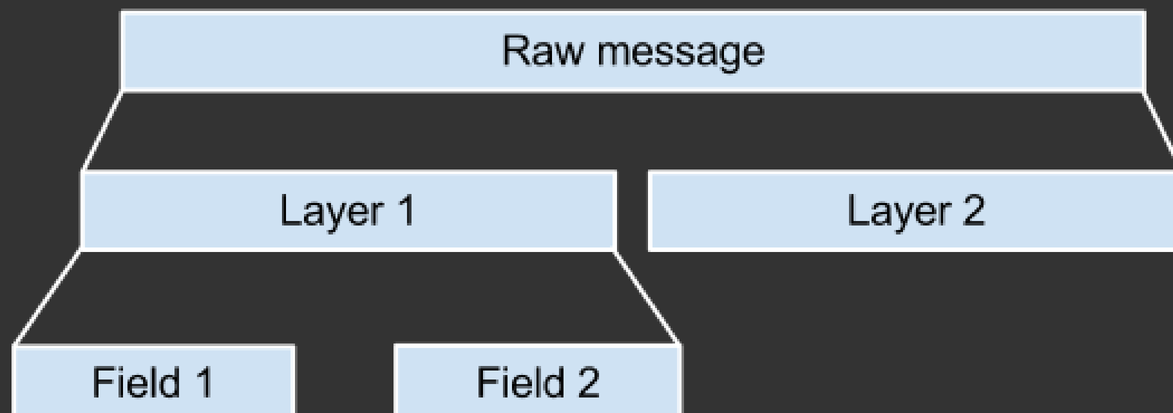
Field 0	Field 1	Field 2	Field 3	Field 4	Field 5	Fie
(.{8})	(4c74657200000000)	(.{2})	(000000)	(.{8})	(00000000)	
hex	hex	hex	hex	hex	hex	
01a76c9f	4c74657200000000	03	000000	bd584ae8	00000000	d2
9782e4fb	4c74657200000000	06	000000	c725c5ca	00000000	b7
46b9c0a9	4c74657200000000	05	000000	d0b4d6d1	00000000	be
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
5d3676f0	4c74657200000000	04	000000	954cfe6f	00000000	6f



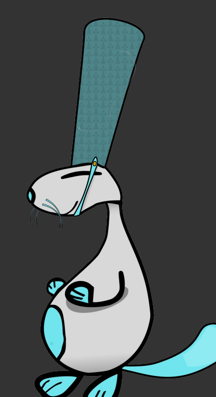
Abstract fields

to decontextualize messages

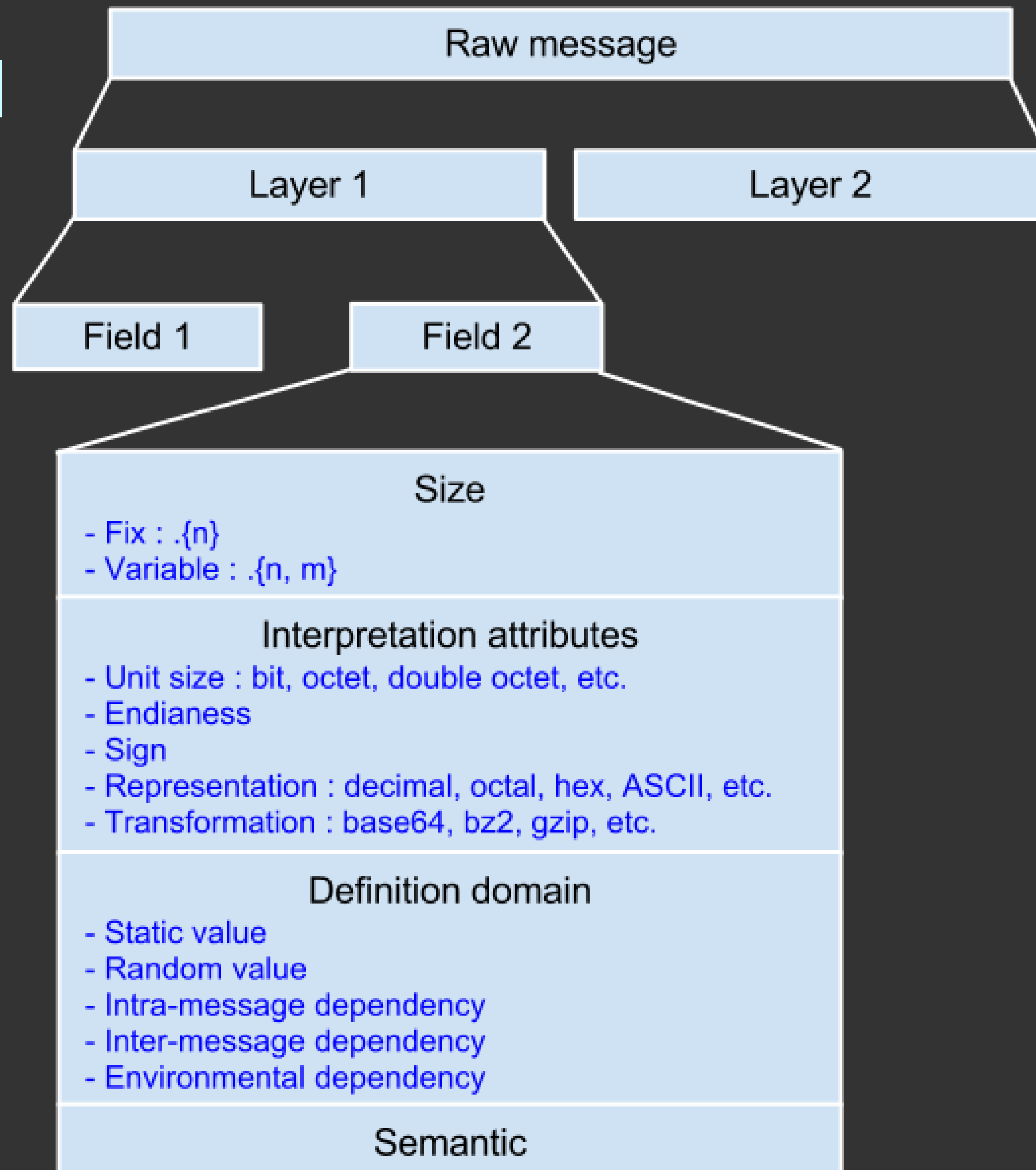
Message format model



Allows multiple partitionment strategies per symbols



Full message format model

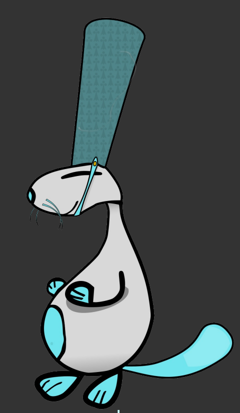


Interlude

'cauze +/- 60 slides left

Lets's do 5 minutes of knitting

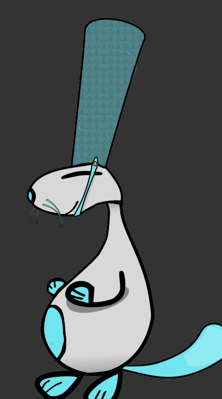
Transformations



How to handle

Encoded values

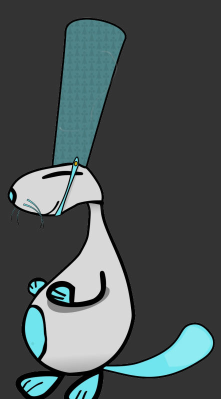
(XOR, ASN.1) ?



How to handle

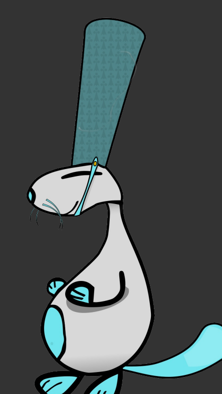
Encrypted values

(symmetric, asymmetric, ...) ?



Messages include **Transformed Fields**

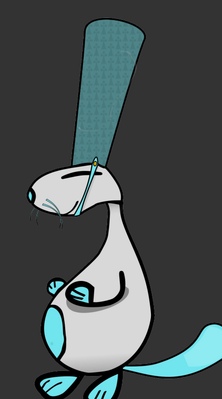
Let's use « **Tranformation Functions** »



The idea

Transform raw bytes into application-level bytes

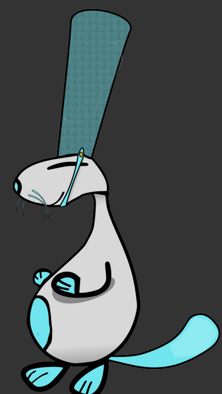
- Applied either on messages, layers or fields
- Provided functions (base64, gzip, bz2, ...)
- Allow custom transformation functions



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

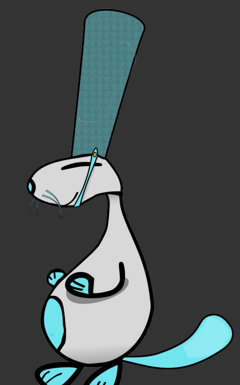
Ex: ZeroAccess XOR-based obfuscation

Create a Custom Tra

Custom Transformation Function

Name of the function: decrypt

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

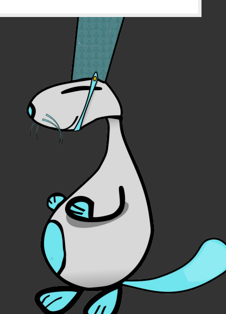


Adding a custom transformation function

Ex: ZeroAccess XOR-based obfuscation

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

87ee1533	4c7465720000000010000000	8b4e2efc	00000000
78ab751d	4c7465720000000010000000	bd71e2fd	00000000
be33b34a	4c7465720000000010000000	ffffffff	00000000



Search for relations

How to handle

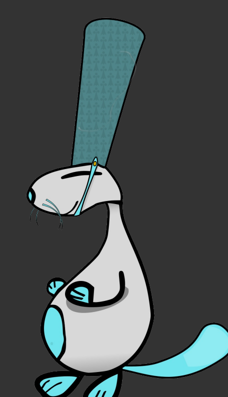
when the **value depends** on something ?

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

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

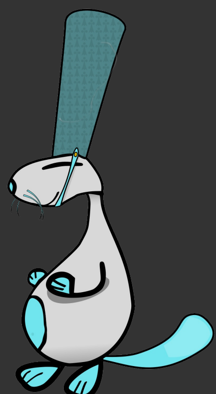
« Inter-Symbol » and « Intra-Symbol » relations

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

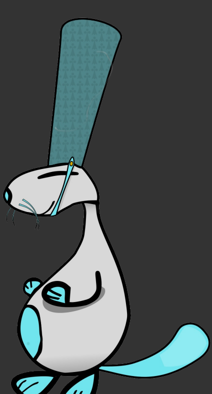
- Measure **dependences** between each pairs
  - Support noisy datasets
- Rank pairs by their score

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

→ Typical Size Field Relation

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

→ Typical CRC Relation

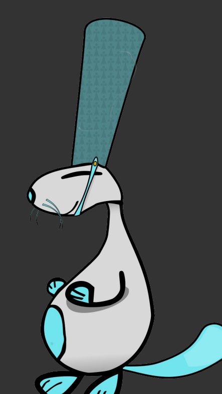


MINE(Value(F1), Size(F2)) = 1

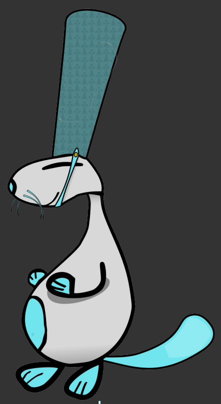
→ Typical Size Field Relation

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

→ Typical CRC Relation



« Environmental » dependencies

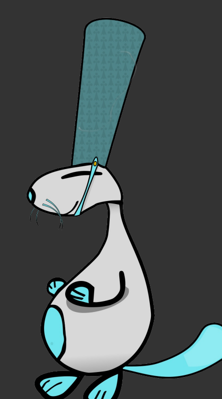


During packets capture:

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

During vocabulary inference:

- Search for meta-data in messages



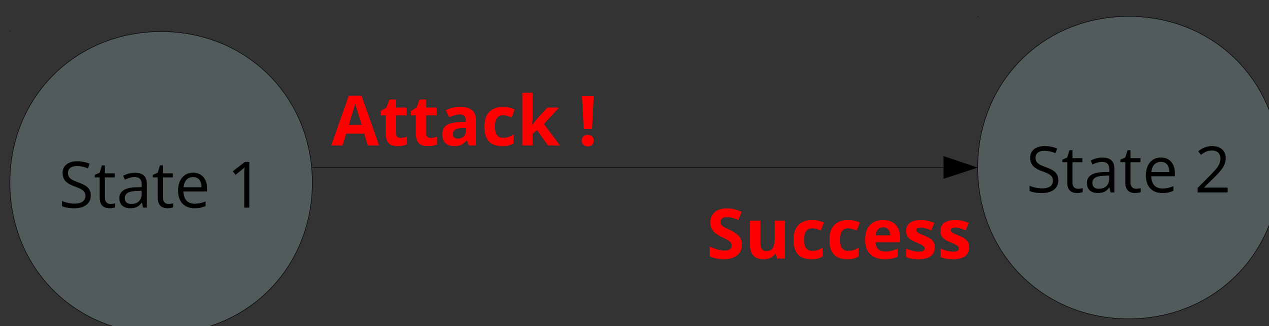
Step 3 : RE grammar

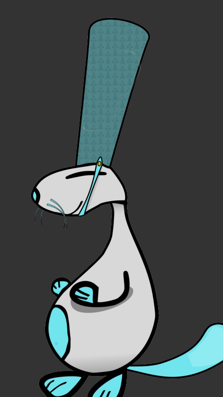
Sequence of valid exchanged symbols.

→ IO Automata

Sequence of valid exchanged symbols.

→ IO Automata

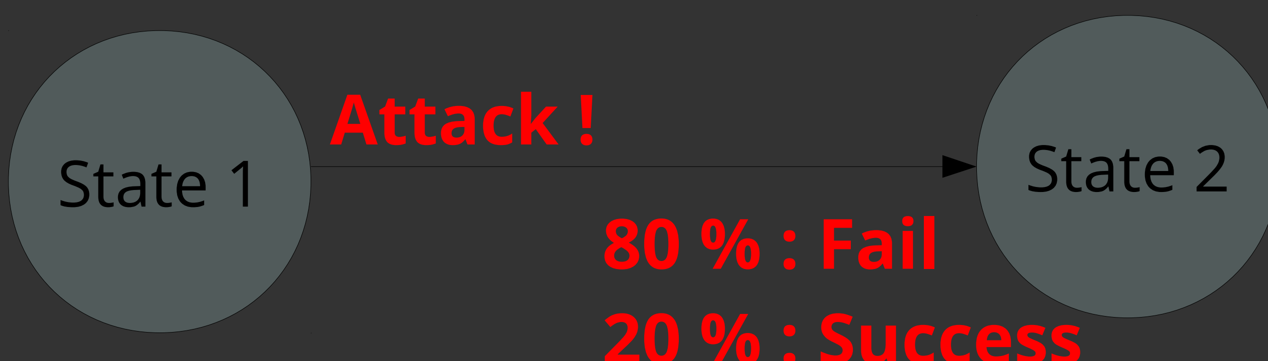




But answers depends on the environment

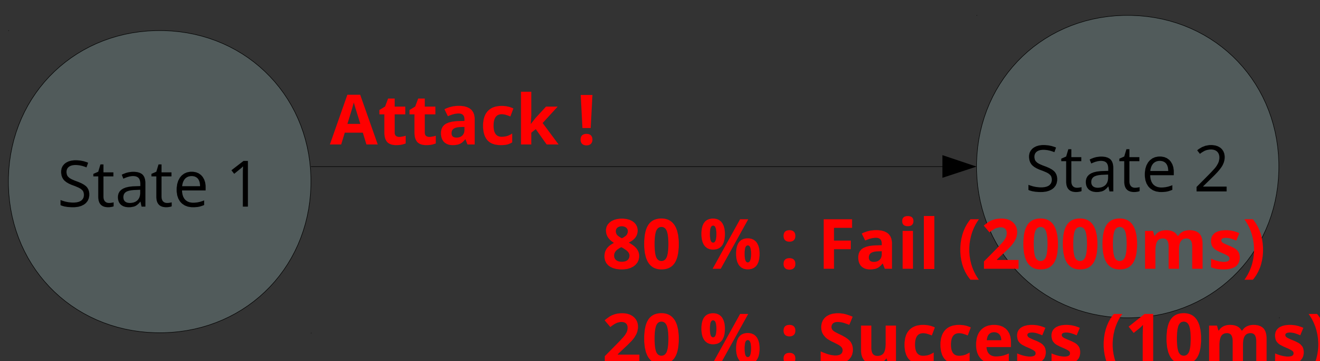
Our model (**SMMDT**)

→ Add probabilities on output messages

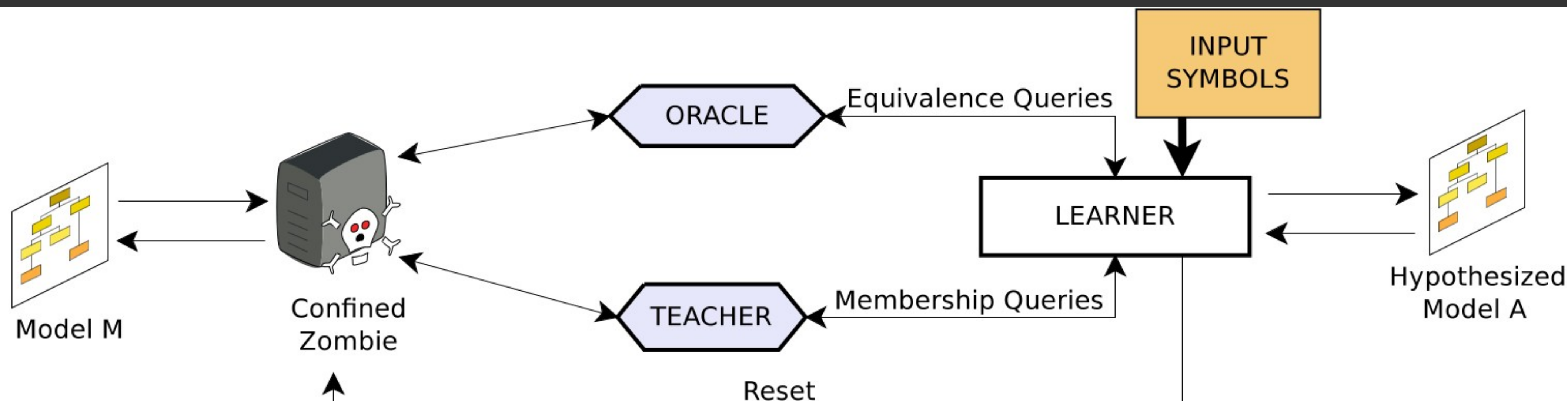


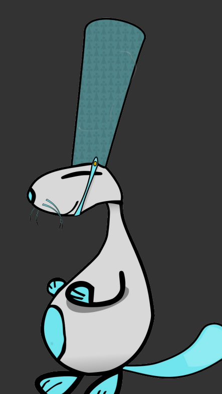
Our model (**SMMDT**)

→ Add the « reaction time »

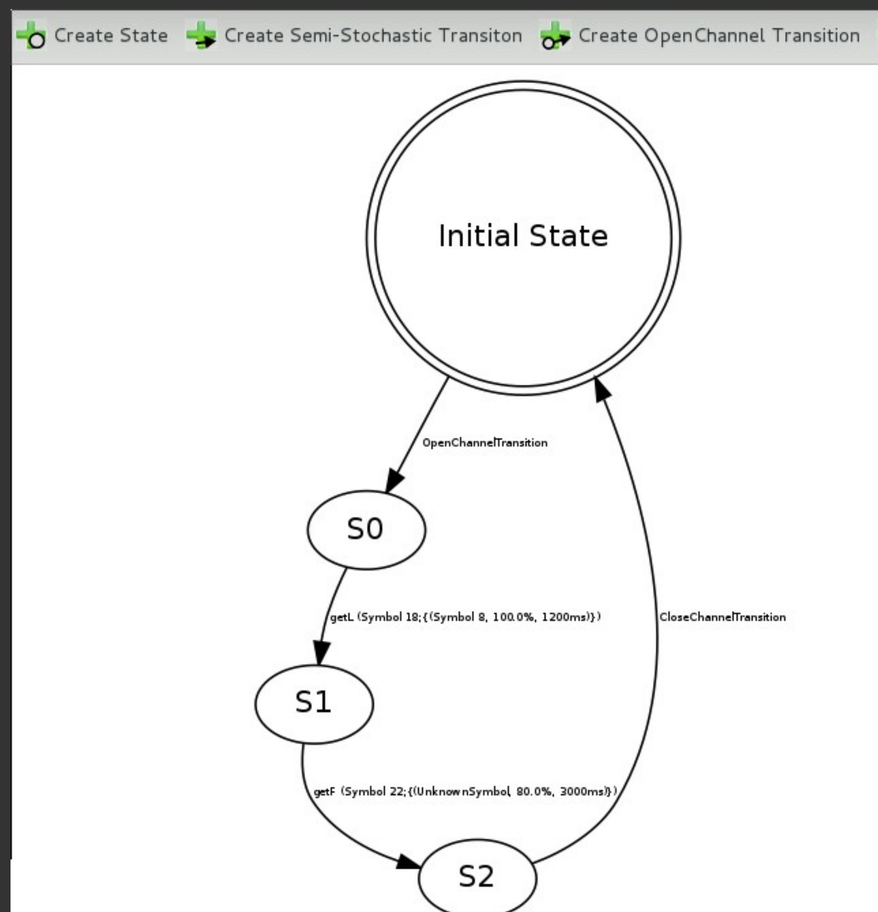


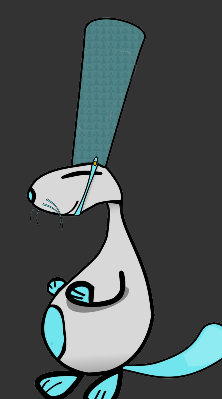
Active Grammatical Inference Process
→ Angluin L*a Algorithm





Active Grammatical Inference Process
→ Angluin L*a Algorithm

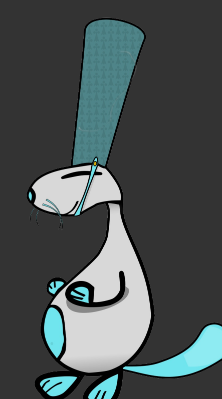





Generating traffic...
FOUNDRY NETWORKS
FastIron Edge 9604 Premium
FES2
FES1
FastIron SuperX
SX1

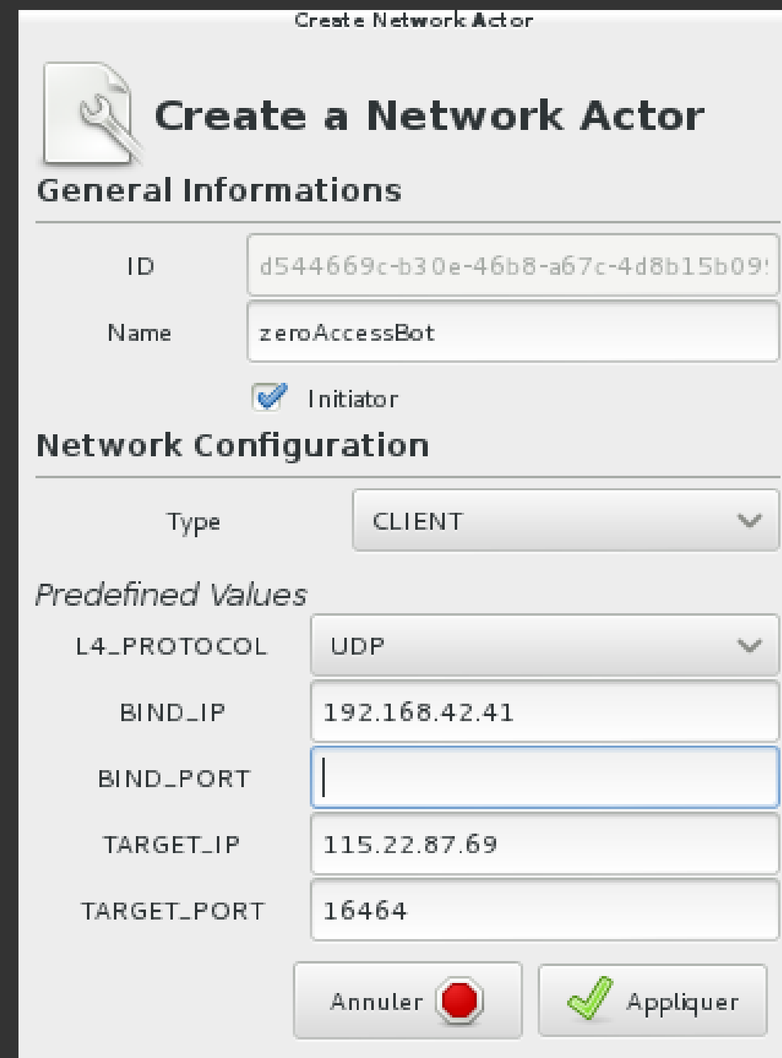
Netzob can generate traffic that:

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



Emulation of different kind of actors and flows

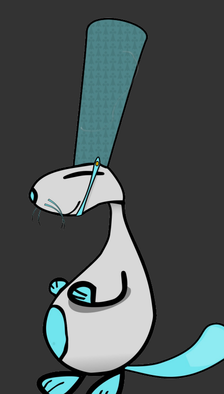
- Client ↔ Real server implementation
- Server ↔ Real client implementation
- Both client(s) and server



Distinction between

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

Ex : TLS with TCP session initiated by the server



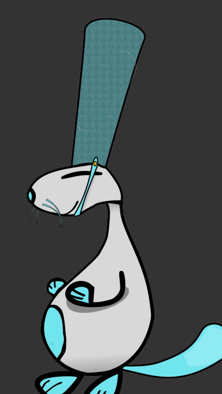
Abstraction from the communication channel

TCP messages

USB channel

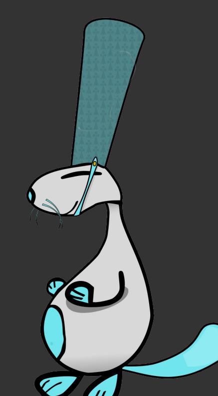
IPC flow

Raw file

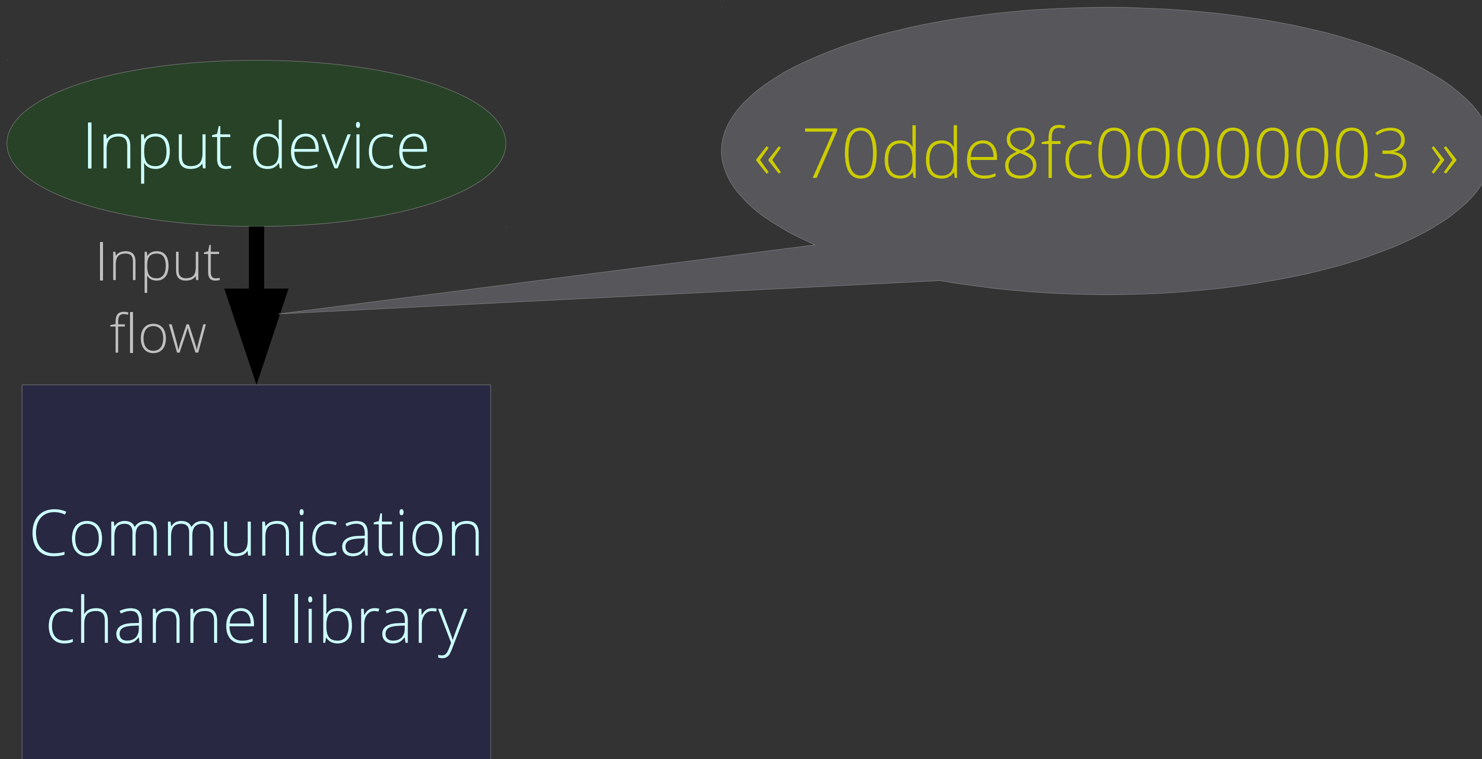


Memory mechanism

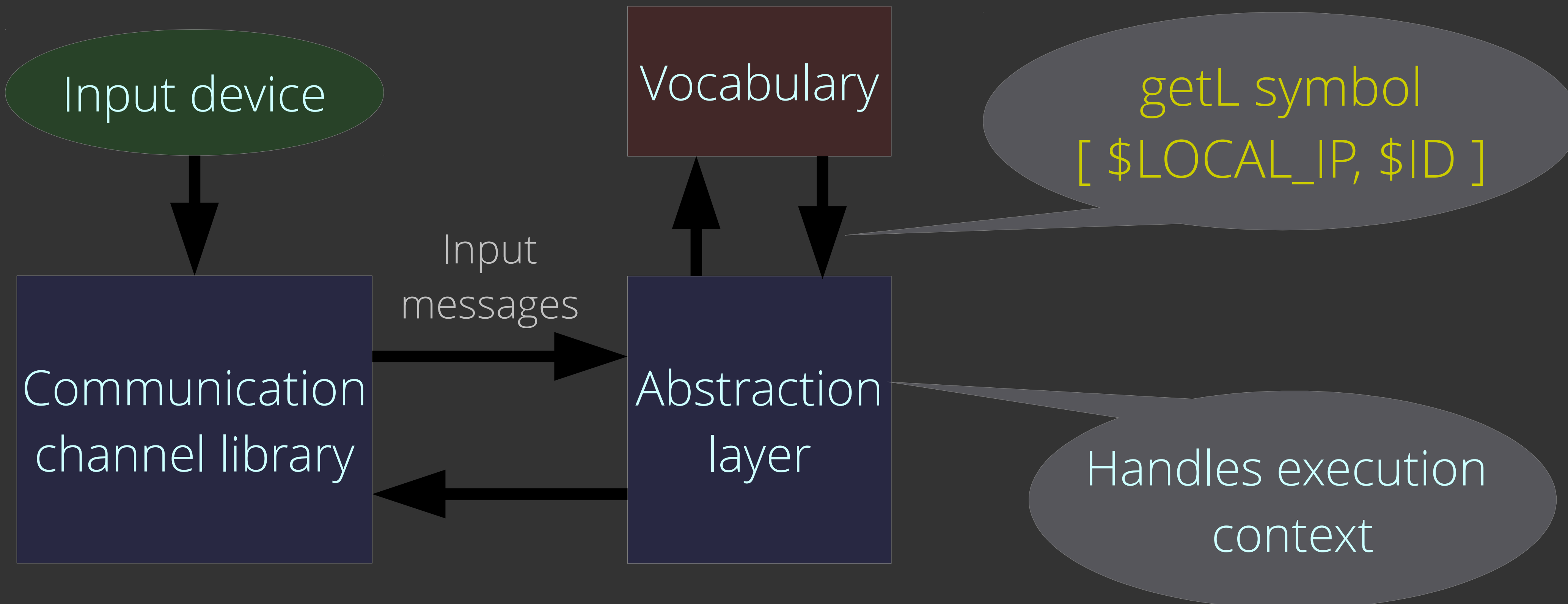
- Some received values are **memorized**…
- …and **reinjected** in future messages
- Also handles contextual values (IP, time, etc.)



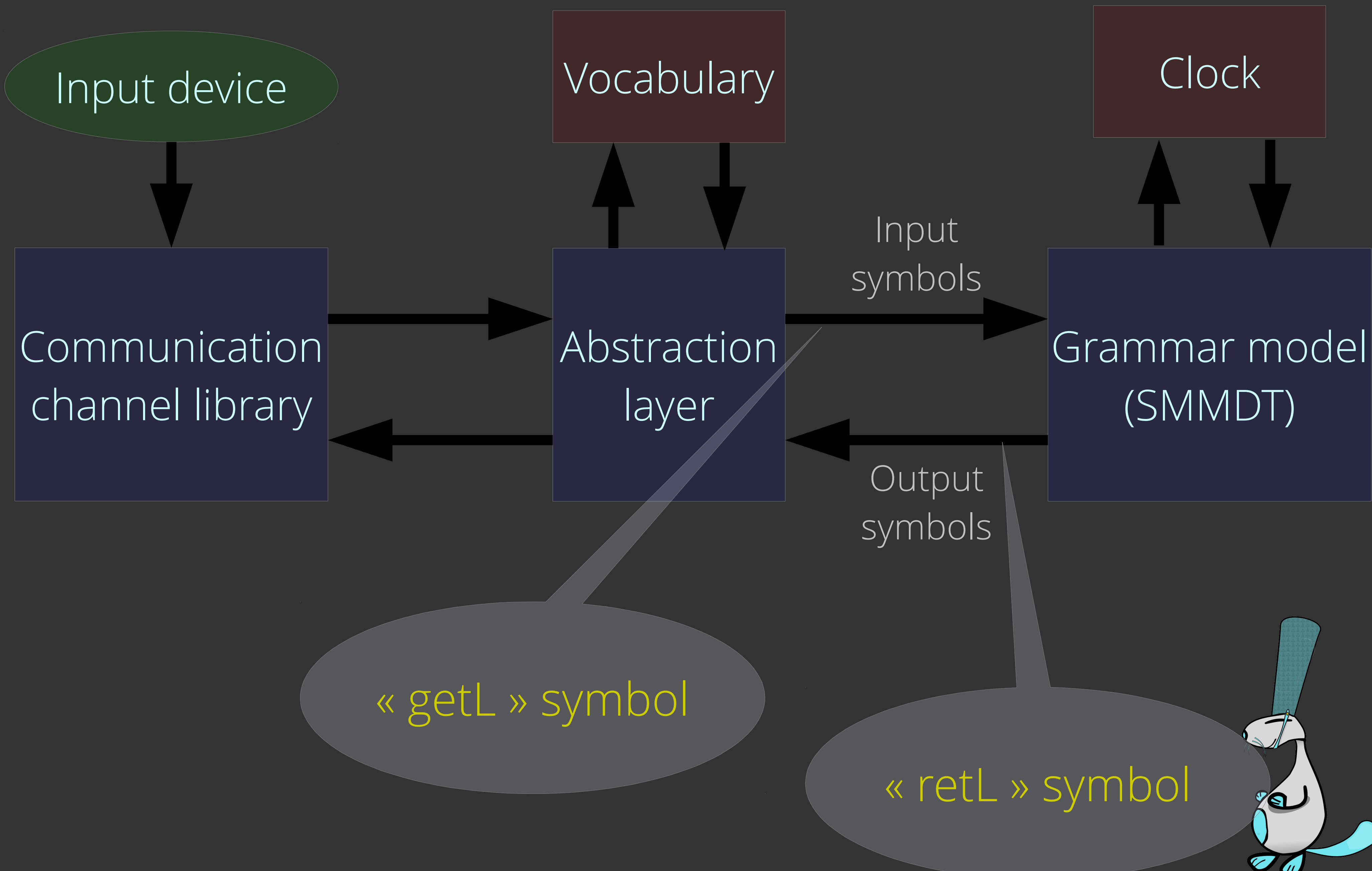
Abstraction and contextualization principles



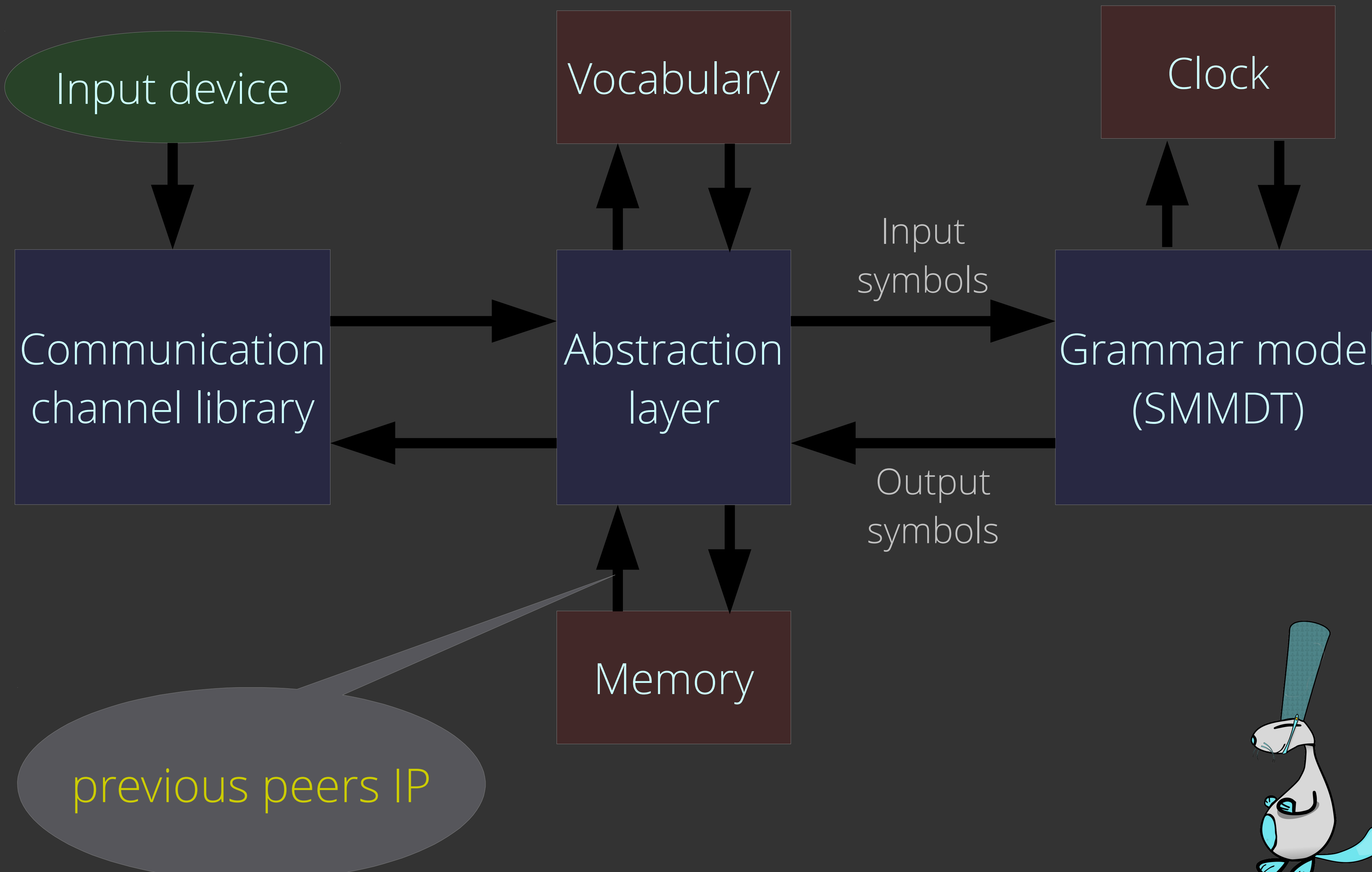
Abstraction and contextualization principles



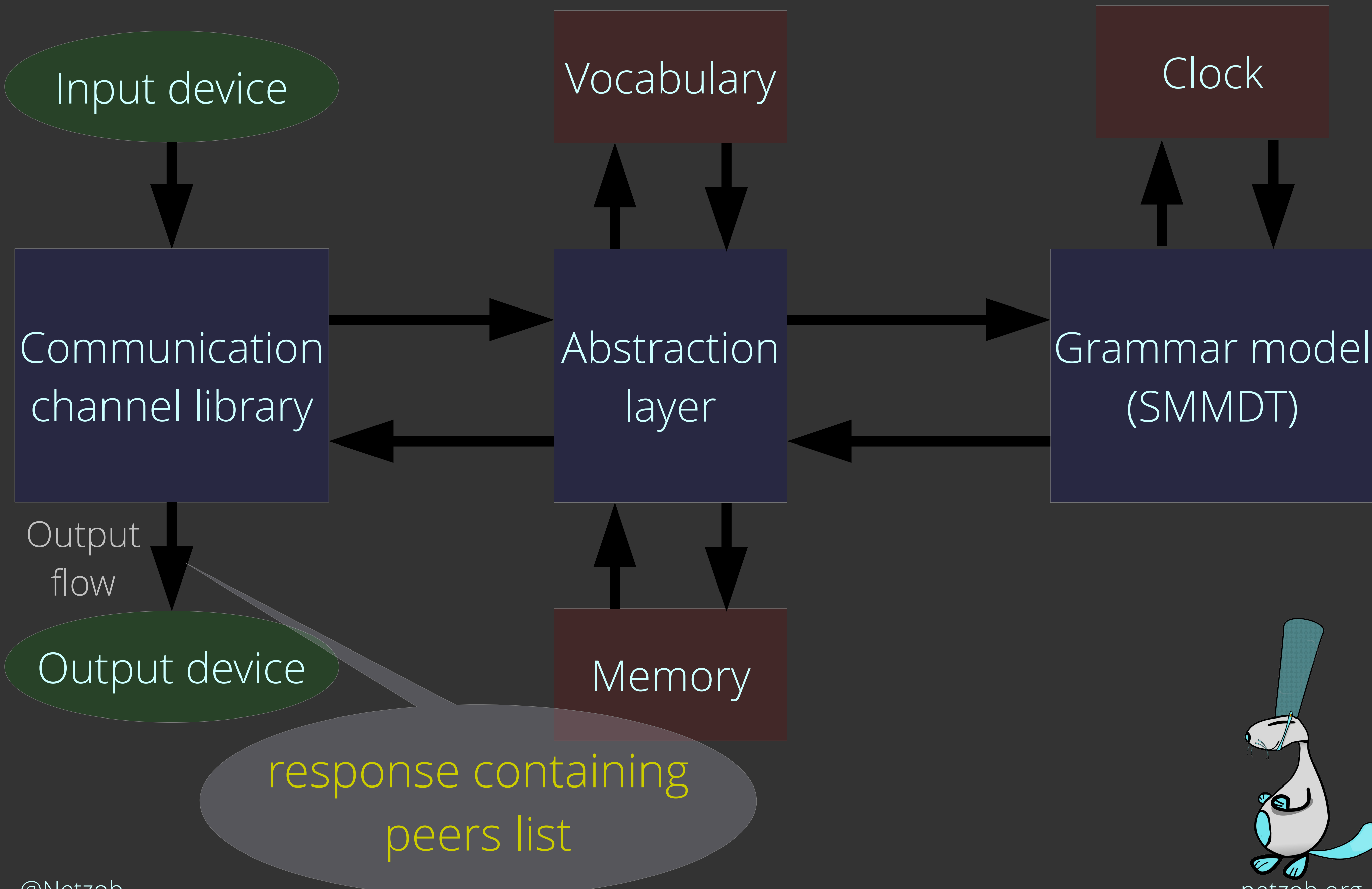
Abstraction and contextualization principles



Abstraction and contextualization principles



Abstraction and contextualization principles

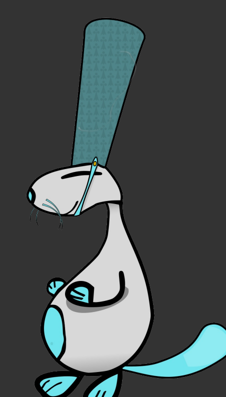


Demos

DEMOS

Demo 1: reversing the protocol of ZA

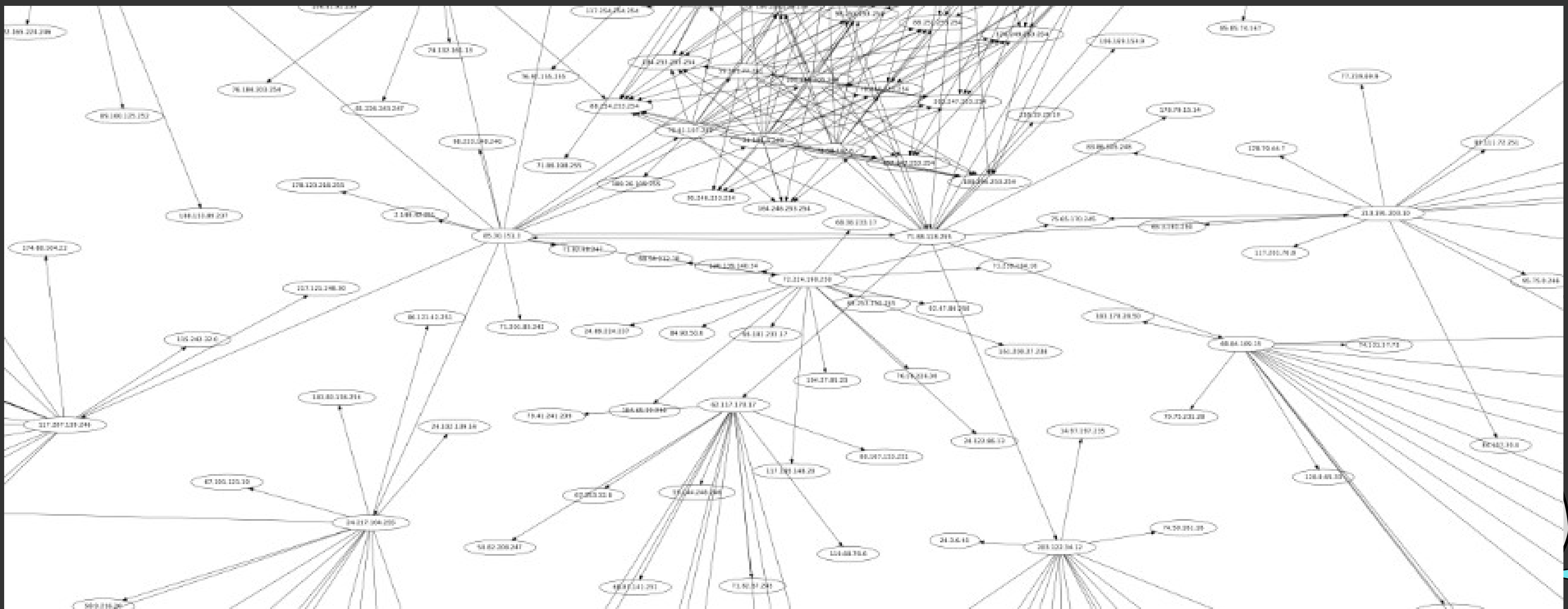
- Import samples of communications
- De-obfuscate exchanges
- Find similar messages
- Split messages in fields
- Abstract Fields
- Search for relations



DEMOS

Demo 2: retrieve the P2P zombie directory

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



Future improvments of Netzob…

Integrated smart fuzzing, by leveraging the simulator engine

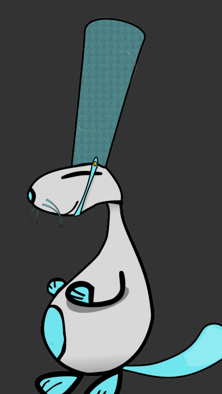
→ Allows to fuzz undocumented and propriatery protocols

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)*

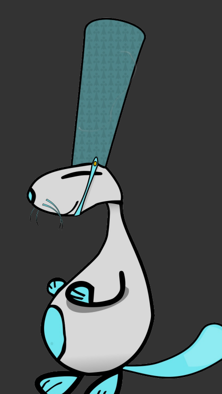
- Wireshark
- Scapy
- Peach Fuzzer

Allows protocol dissection with established tools

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

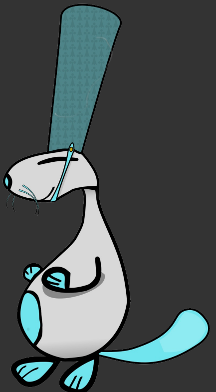
- Wireshark
- Scapy
- Peach Fuzzer

Allows fuzzing of unknown protocols with well-known tools

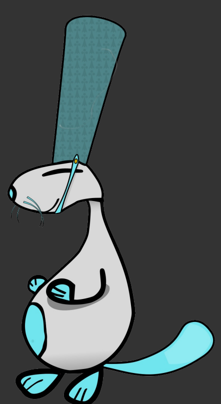



Conclusion…

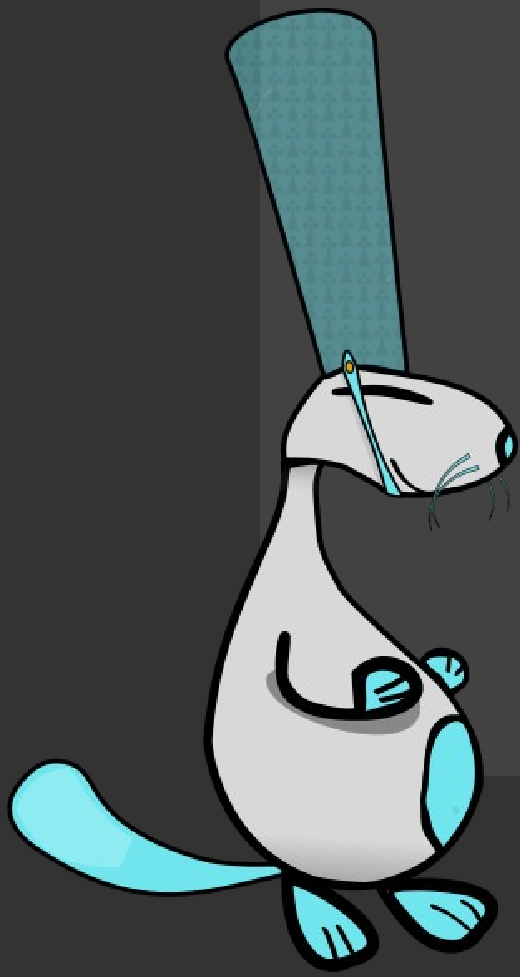
- Protocol RE automation domain is quite active at the academic level
- But no real tool available…
- Netzob tries to fill this lack by
  - Supporting academic researches
  - Being usable in operational context



- 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