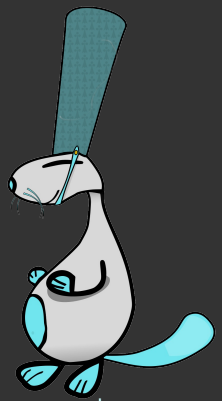
Reverse and Simulate your Enemy Botnet C&C

Blackhat Abu Dhabi

December 5 2012

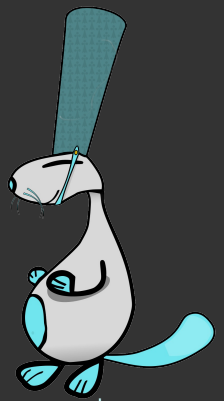
Please complete the Speaker Feedback Surveys.



Authors...

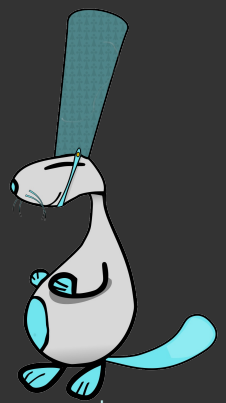
« You talkin' to me? »

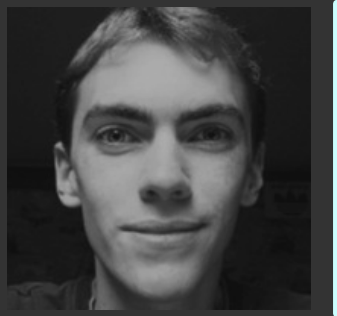
Travis Bickle



Frédéric GUIHERY

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



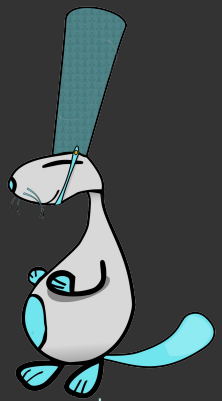


Georges BOSSERT

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

Supelec CIDre
Research team

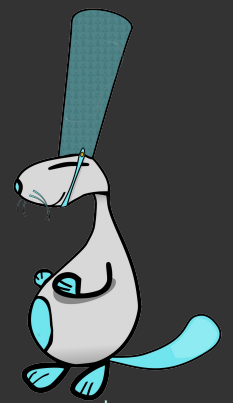
Advisers :
- *Guillaume Hiet*
- *Ludovic Mé*



AMOSSYS, France

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

www.amossys.fr

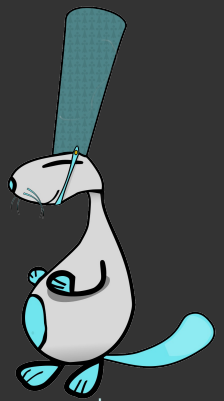


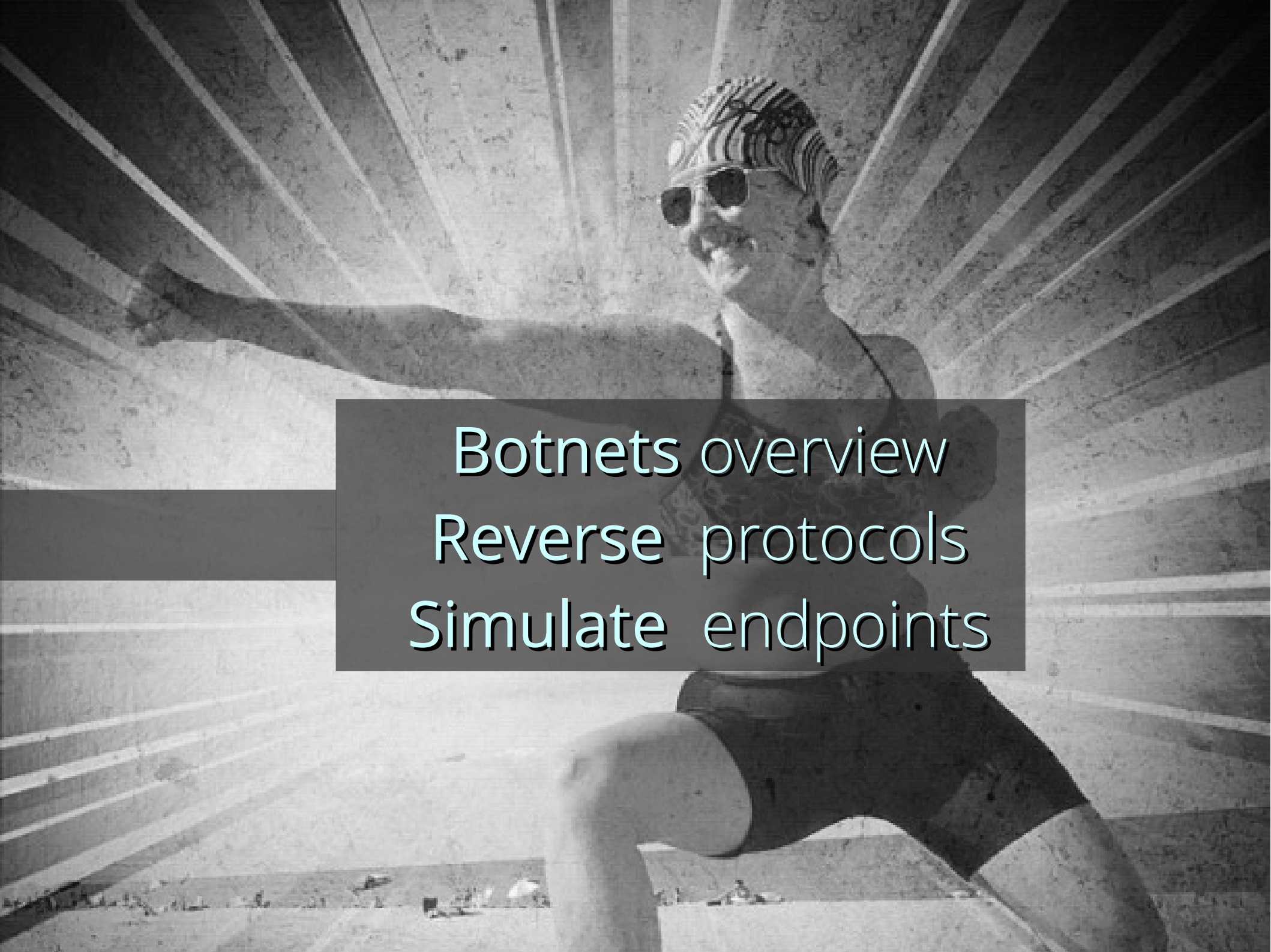

Yes, we're French !

Topics...

« Go ahead, make my day »

Harry Callahan

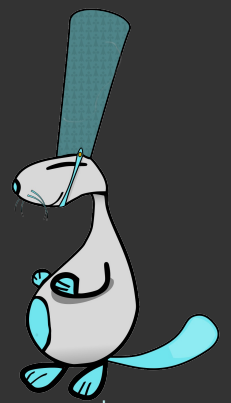



Botnets overview
Reverse protocols
Simulate endpoints


Why ?

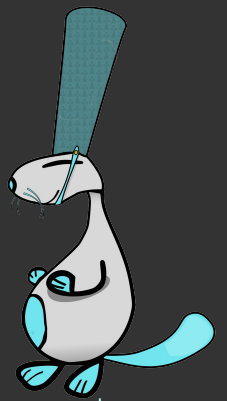
Botnets

- *2 different binaries (or part of)*
  - *the master*
  - *the zombies*
- *A set of actions to perform*
- *A Communication Channel*
  - *A network topology*
  - *A proprietary protocol*



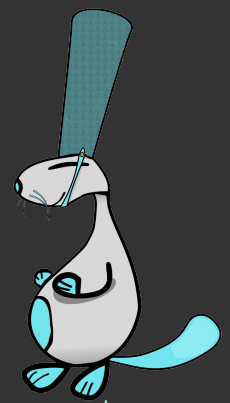
Botnets

- *2 different binaries (or part of)*
  - *the master*
  - *the zombies*
- *A set of actions to perform*
- *A Communication Channel*
  - *A network topology*
  - *A proprietary protocol*



Botnets

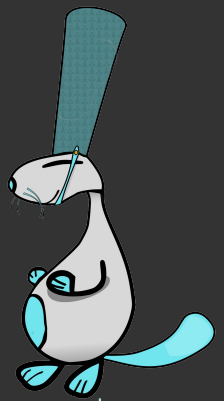
- *2 different binaries (or part of)*
  - *the master*
  - *the zombies*
- *A set of actions to perform*
- *A Communication Channel*
  - *A network topology*
  - *A proprietary protocol*



Botnets:

- *2 different binaries (or part of)*
  - *the master*
  - *the zombies*
- *A set of actions to perform*
- *A Communication Channel*
  - *A network topology*
  - *A proprietary protocol*

Analysts' point-of-view

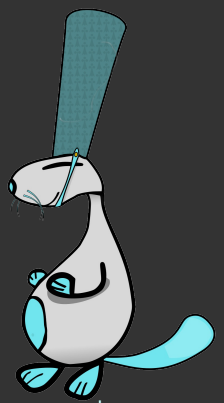


Botnets:

- *2 different binaries (or part of)*
  - *the master*
  - *the zombies*
- *A set of actions to perform*
- *A Communication Channel*
  - *A network topology*
  - *A proprietary protocol*

Analysts' point-of-view

Binary RE
(static/dynamic)



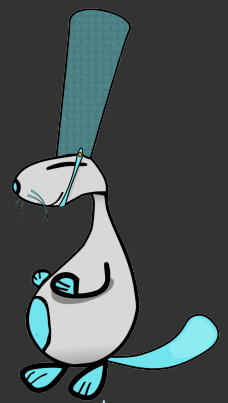
Botnets:

- *2 different binaries (or part of)*
  - *the master*
  - *the zombies*
- *A set of actions to perform*
- *A Communication Channel*
  - *A network topology*
  - *A proprietary protocol*

Analysts' point-of-view

Binary RE
(static/dynamic)

Behavioural Analysis
(AV/HIDS)



Botnets:

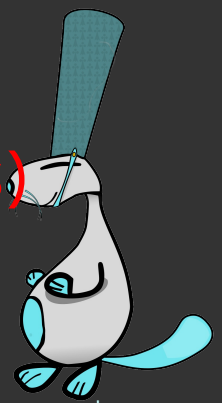
- *2 different binaries (or part of)*
  - *the master*
  - *the zombies*
- *A set of actions to perform*
- *A Communication Channel*
  - *A network topology*
  - *A proprietary protocol*

Analysts' point-of-view

Binary RE
(static/dynamic)

Behavioural Analysis
(AV/HIDS)

Macro Analysis
(ISPs/consortiums)



Botnets:

- *2 different binaries (or part of)*
  - *the master*
  - *the zombies*
- *A set of actions to perform*
- *A Communication Channel*
  - *A network topology*
  - *A proprietary protocol*

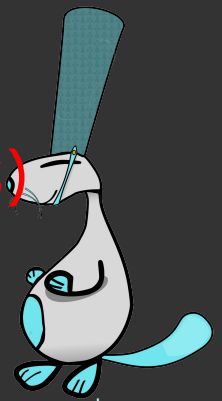
Analysts' point-of-view

Binary RE
(static/dynamic)

Behavioural Analysis
(AV/HIDS)

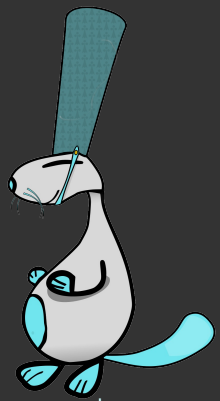
Macro Analysis
(ISPs/consortiums)

Protocol RE
(??????????????????)



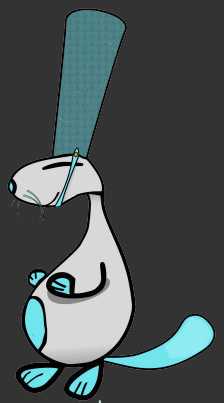

No available tool to reverse
a proprietary protocol...
Should we create one?

Other « use-cases » for protocol RE



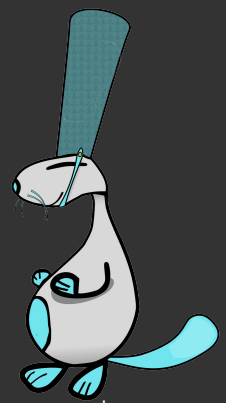
To assess the robustness of implementations

- *Ex : Fuzz the control API of a centrifuge*



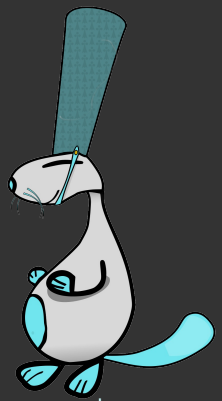
To analyze traffic and identify potential data leakage

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



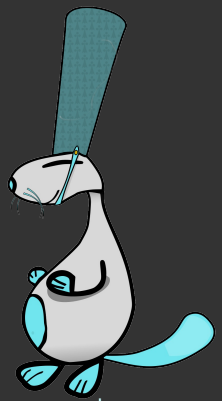
To compare the implementation of a protocol with its official specifications

- *Ex : CC evaluations of crypto products*

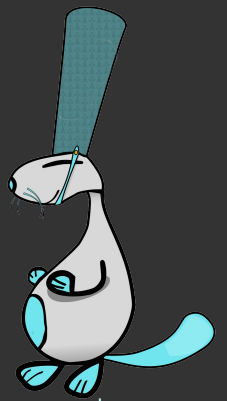


To develop a free version of a proprietary implementation

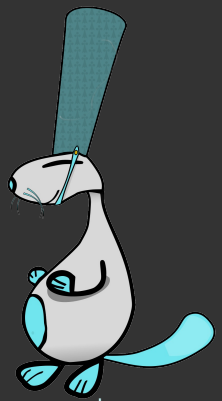
- *Ex : Drew Fisher's talk @ 28C3 on Kinect RE*



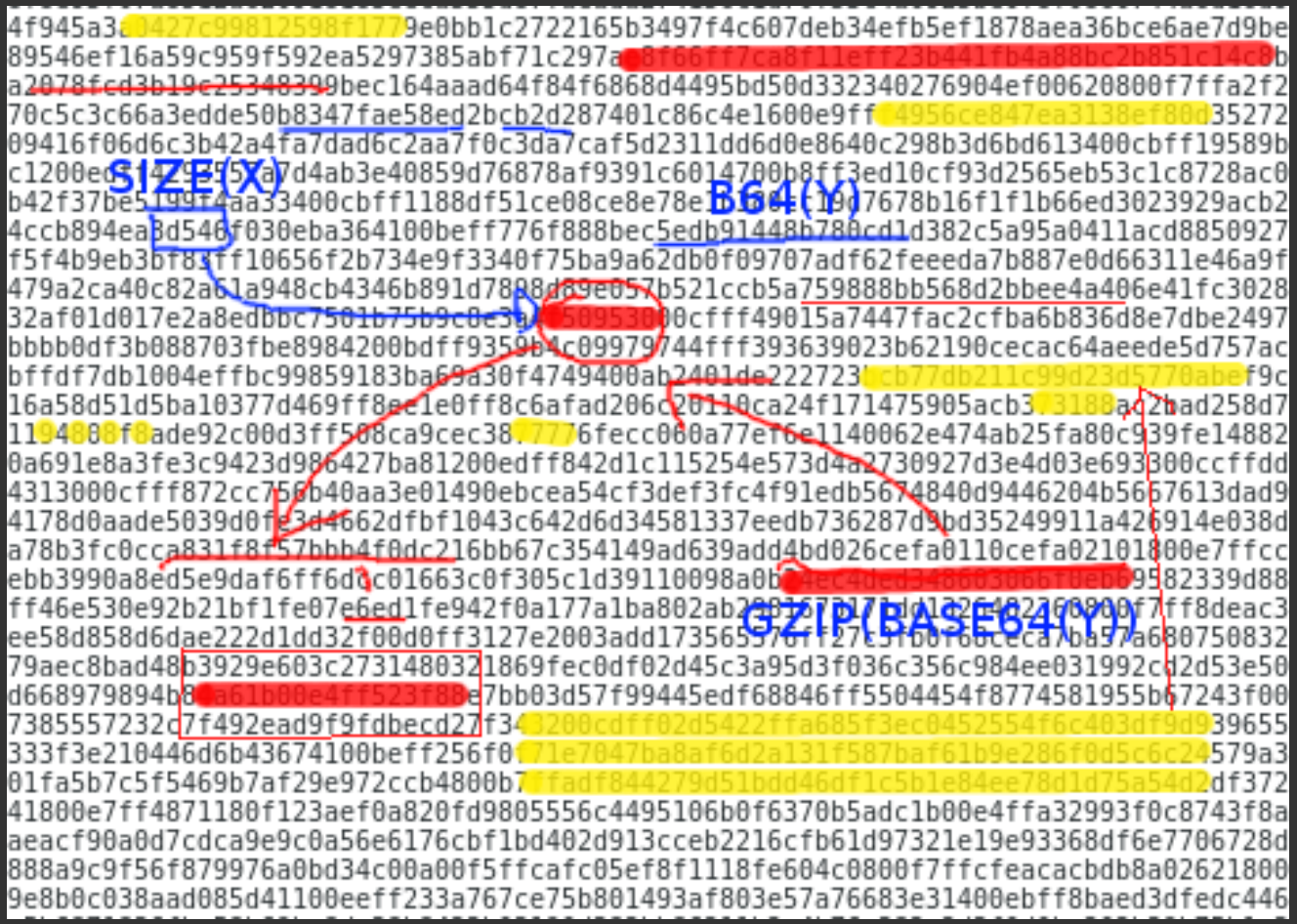
Current reverse engineering approach...

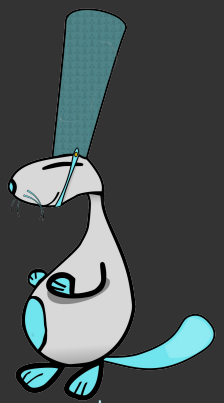


Do you remember the last time you reversed a protocol ?



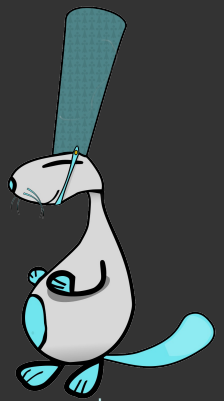
Do you remember the last time you reversed a protocol ?





Do you remember last time you reversed a protocol ?

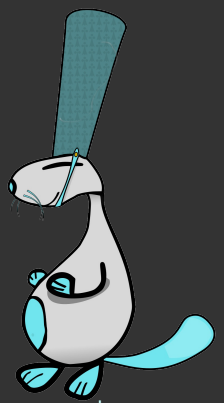
- Complex
  - Time-consuming
- Mostly Manual



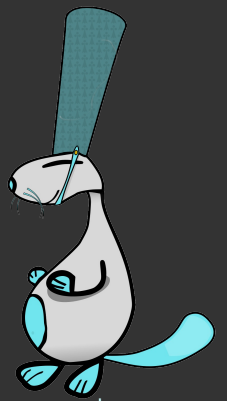
Do you remember last time you reversed a protocol ?

- Complex
  - Time-consuming
- Mostly Manual

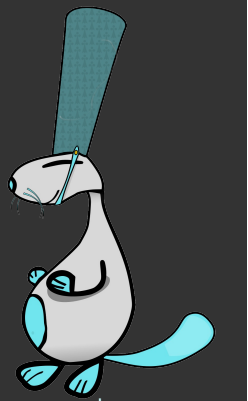
MOSTLY VISUAL



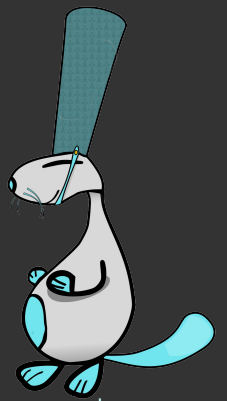
But wait... Couldn't we automate many RE tasks ?



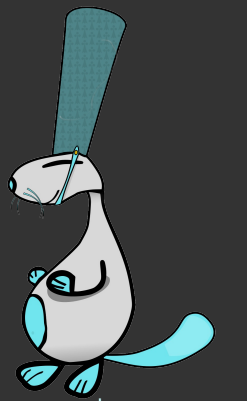
Examples of tasks we would like to automate



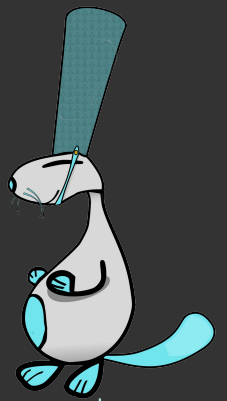
... capture **samples** from its network communications



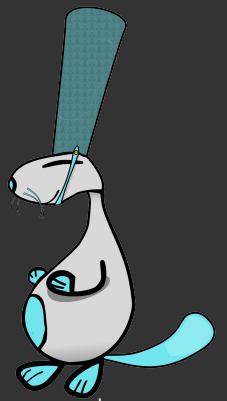
… split **messages** in « **equivalent** » groups



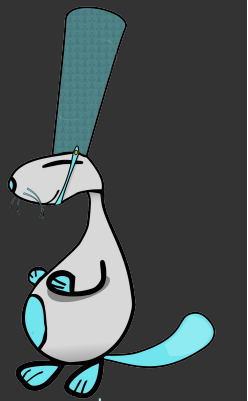
... understanding field **semantics**



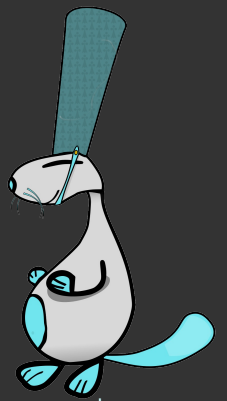
... find **size fields** and associated payloads



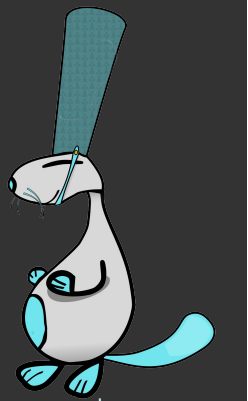
... find CRCs, hashes and other
relations between bits



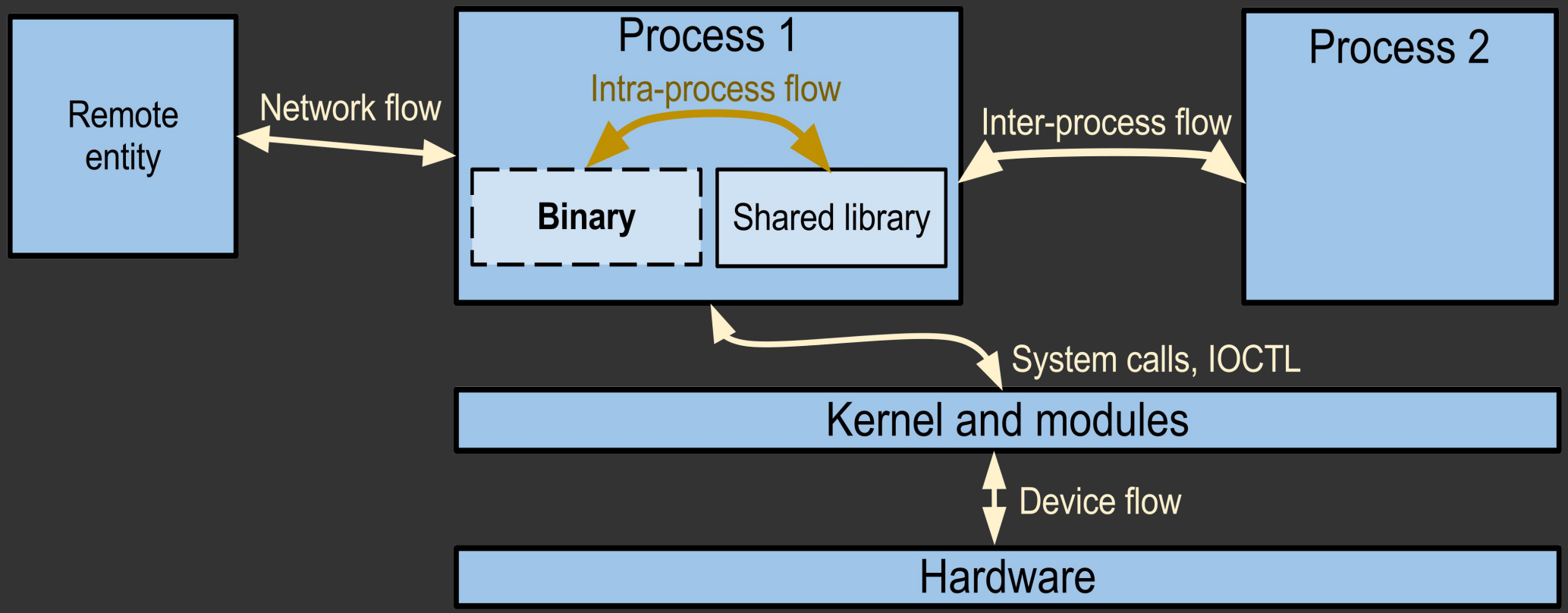
... understand the **valid sequences** of exchanged messages



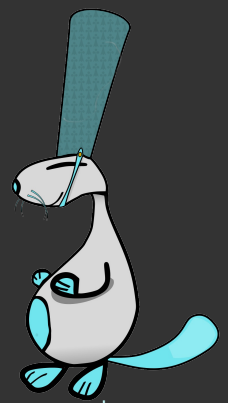
... simulate **realistic** and **controllable** actors



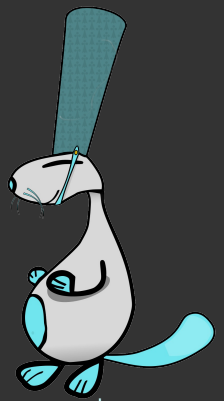
Some reminders about protocols



Protocols are everywhere



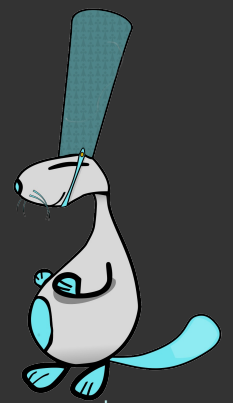
Before reversing we need a model for protocols



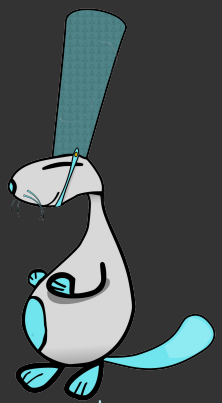
Academics are very good with models :)

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

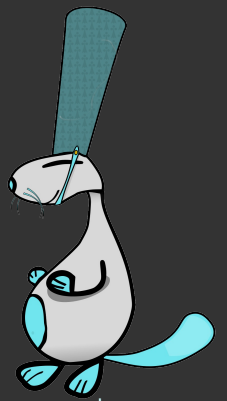
A Communication Protocol is made of 5 distinct parts ...



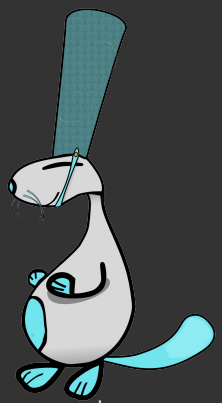
a service (1/5)



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

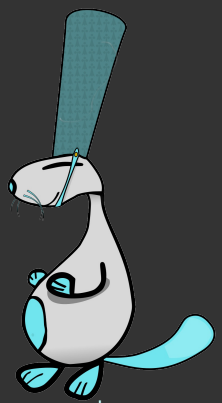


a **vocabulary** of messages **(3/5)**

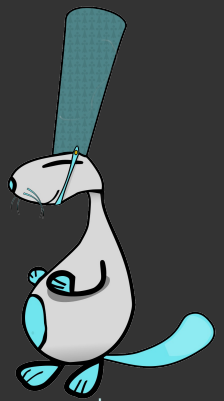


the **encoding** (format) of each message

(4/5)

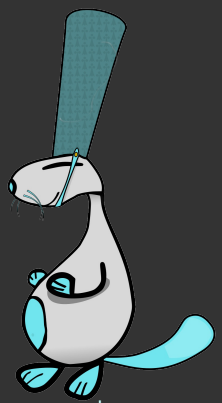


the procedure rules (5/5)

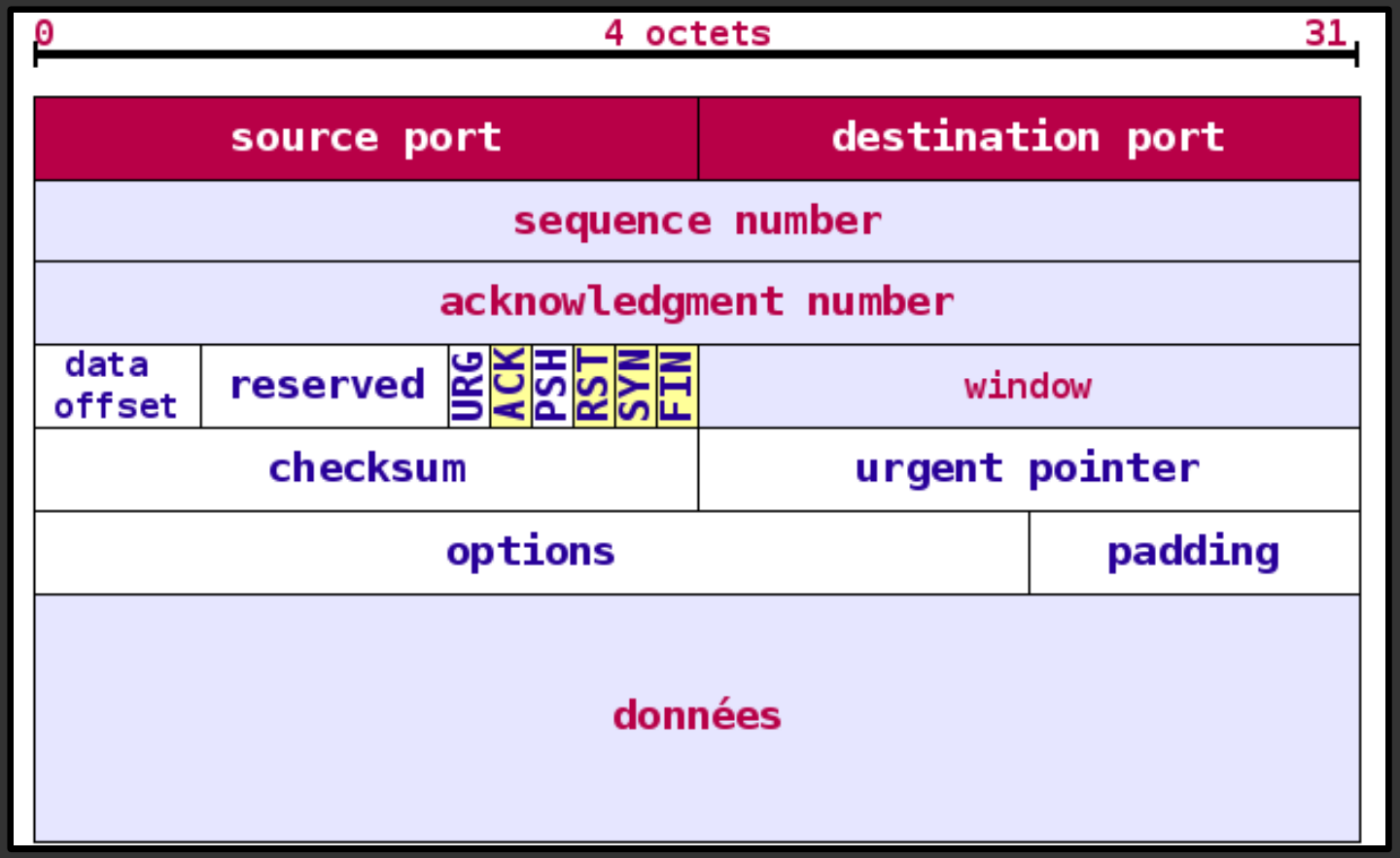


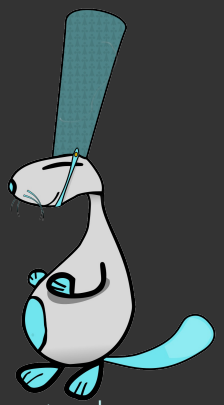
Reduced model for a Protocol

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

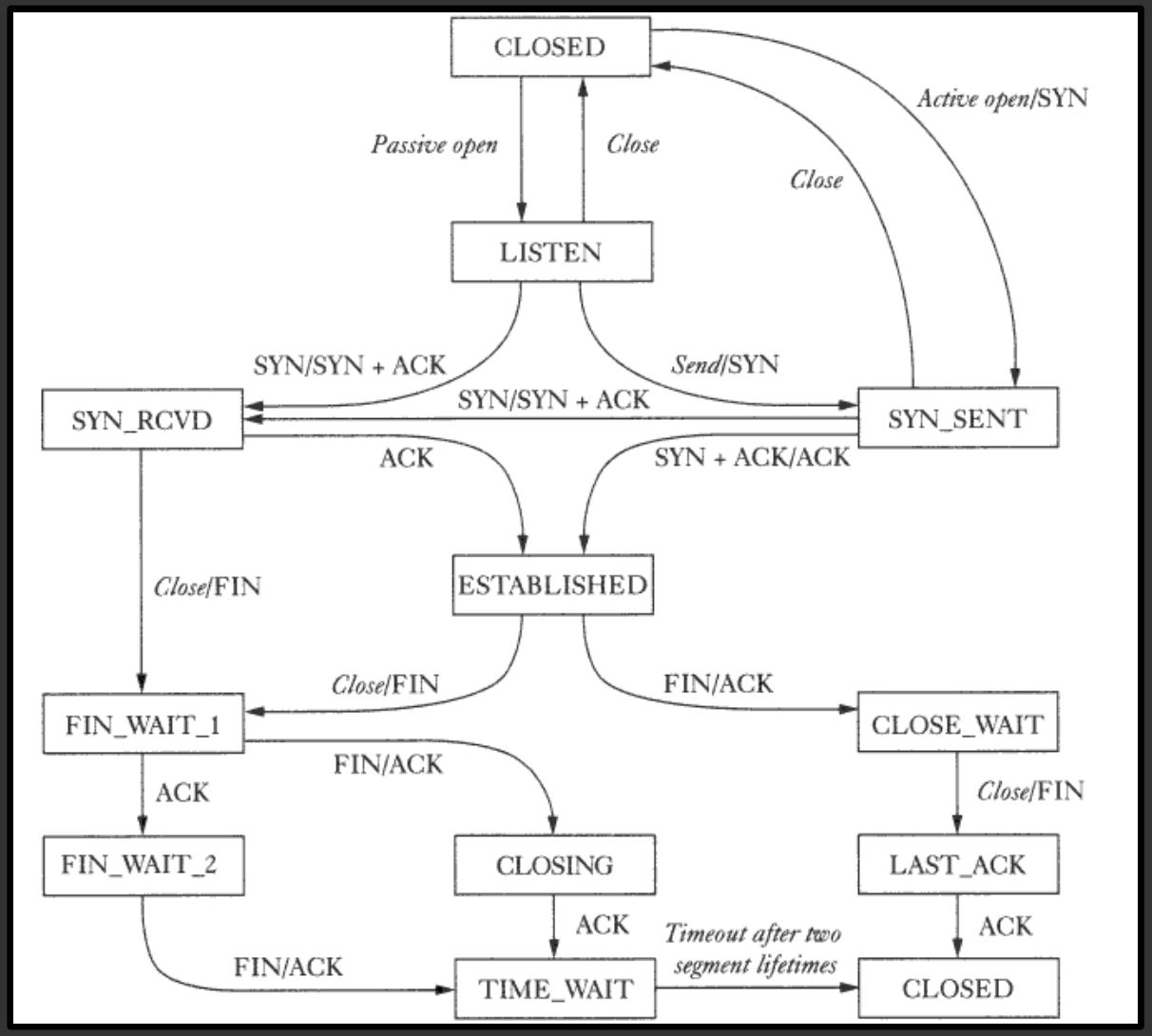


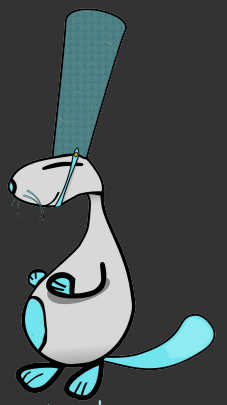
Message Format

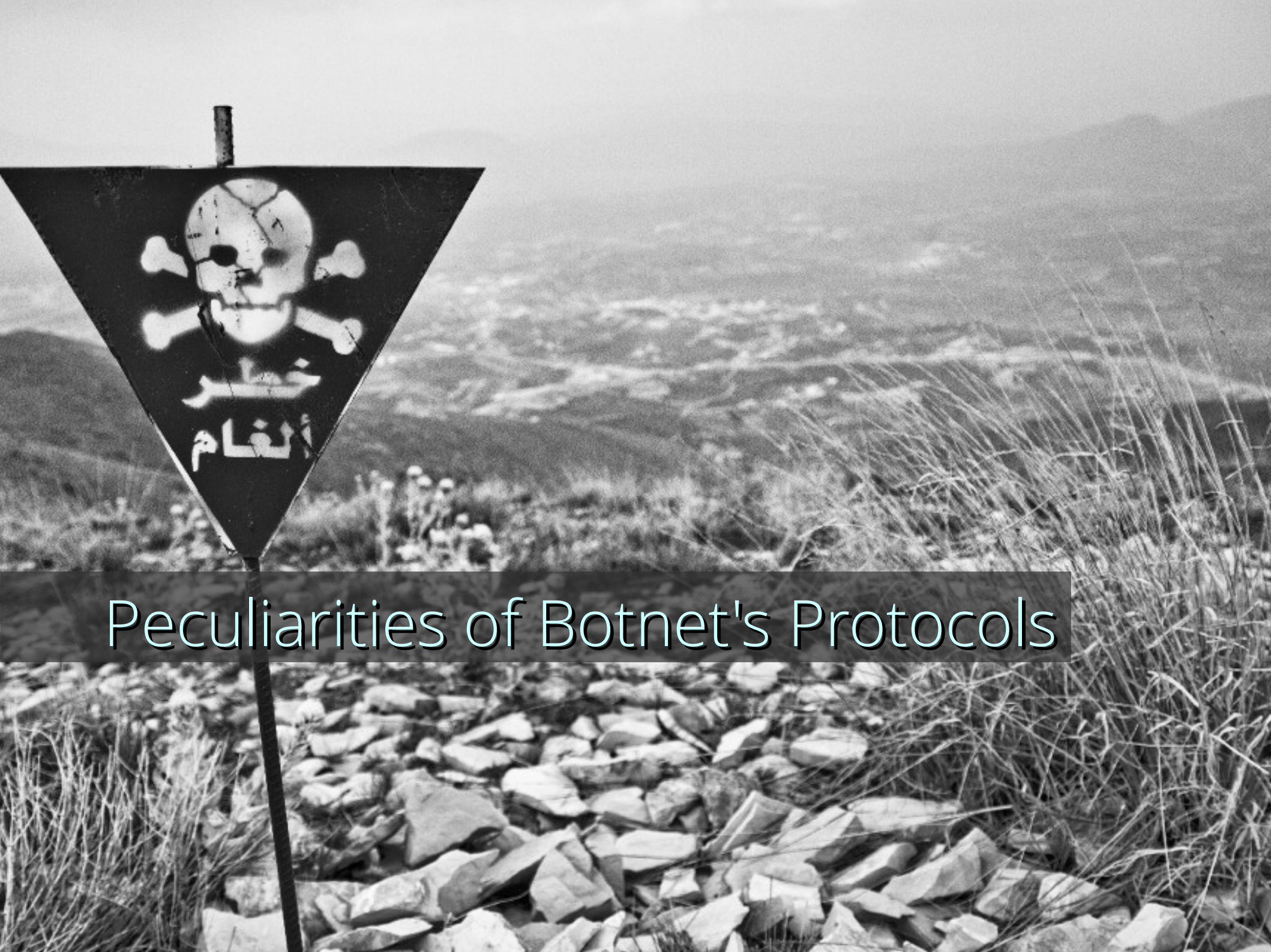




State Machine



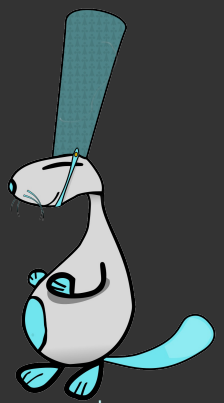




Peculiarities of Botnet's Protocols

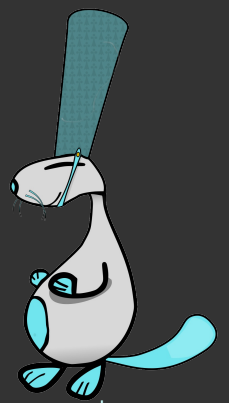
Focus on C&C protocols

- *Send orders to zombies*
- *Receive status from zombies*
- *Zombie « Heartbeat » channel*



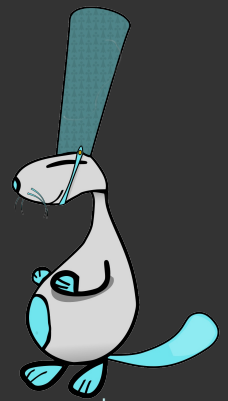
Focus on C&C protocols

- *Send orders to zombies*
- *Receive status from zombies*
- *Zombie « Heartbeat » channel*



Focus on C&C protocols

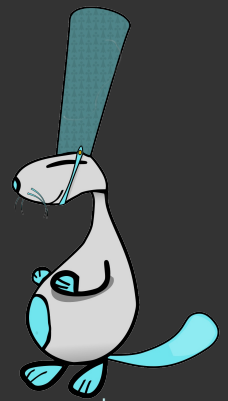
- *Send orders to zombies*
- *Receive status from zombies*
- *Zombie « Heartbeat » channel*



Two types of protocols

- *ASCII explicit messages*
- *High diversity of messages*
- *High interactivity in C&C*
- *Single version*
- *Very-little encryption*

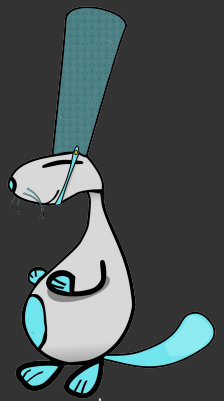
- *Binary messages*
- *Low diversity of messages*
- *Low/No interactivity in C&C*
- *Multiple versions*
- *Little encryption*



Two types of protocols

- *ASCII explicit messages*
- *High diversity of messages*
- *High interactivity in C&C*
- *Single version*
- *Very-little encryption*

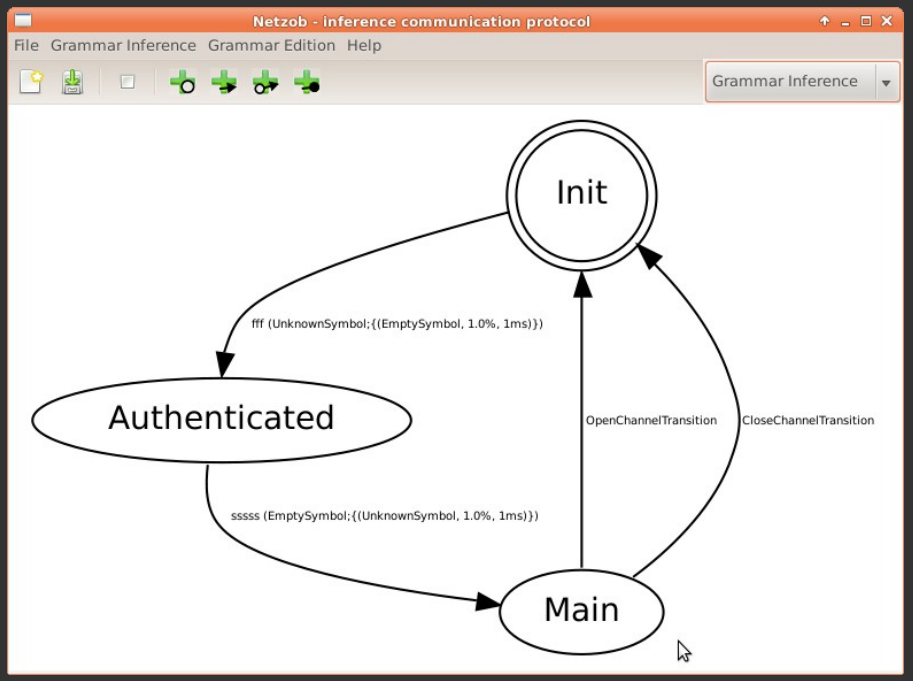
- *Binary messages*
- *Low diversity of messages*
- *Low/No interactivity in C&C*
- *Multiple versions*
- *Little encryption*

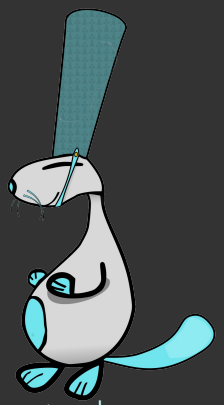



Introducing Netzob ...

Goals of Netzob

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



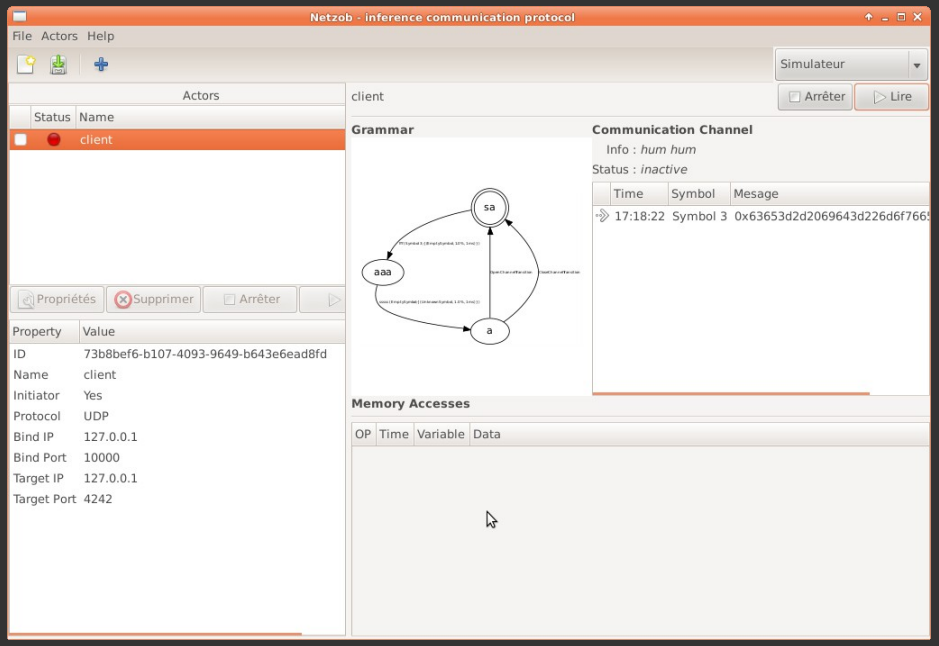


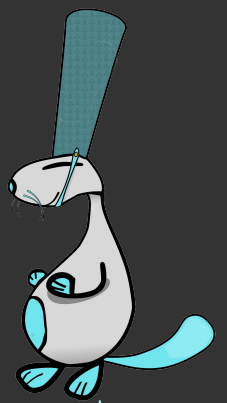
Goals of Netzob

Infer proprietary protocols

- Simulate actors of a communication

Smart-Fuzz targeted implementations



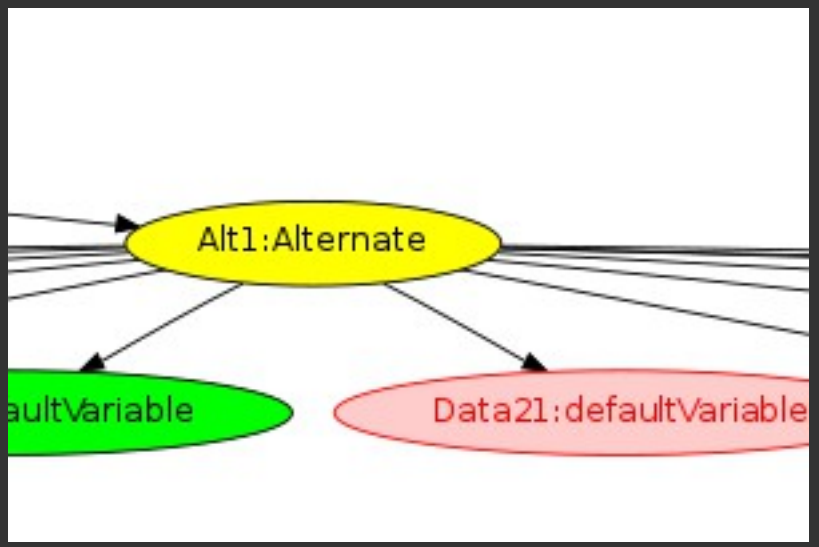


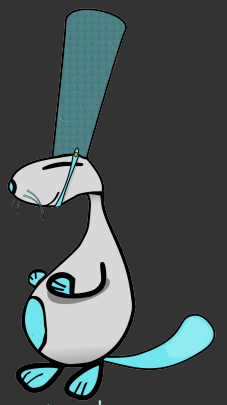
Goals of Netzob

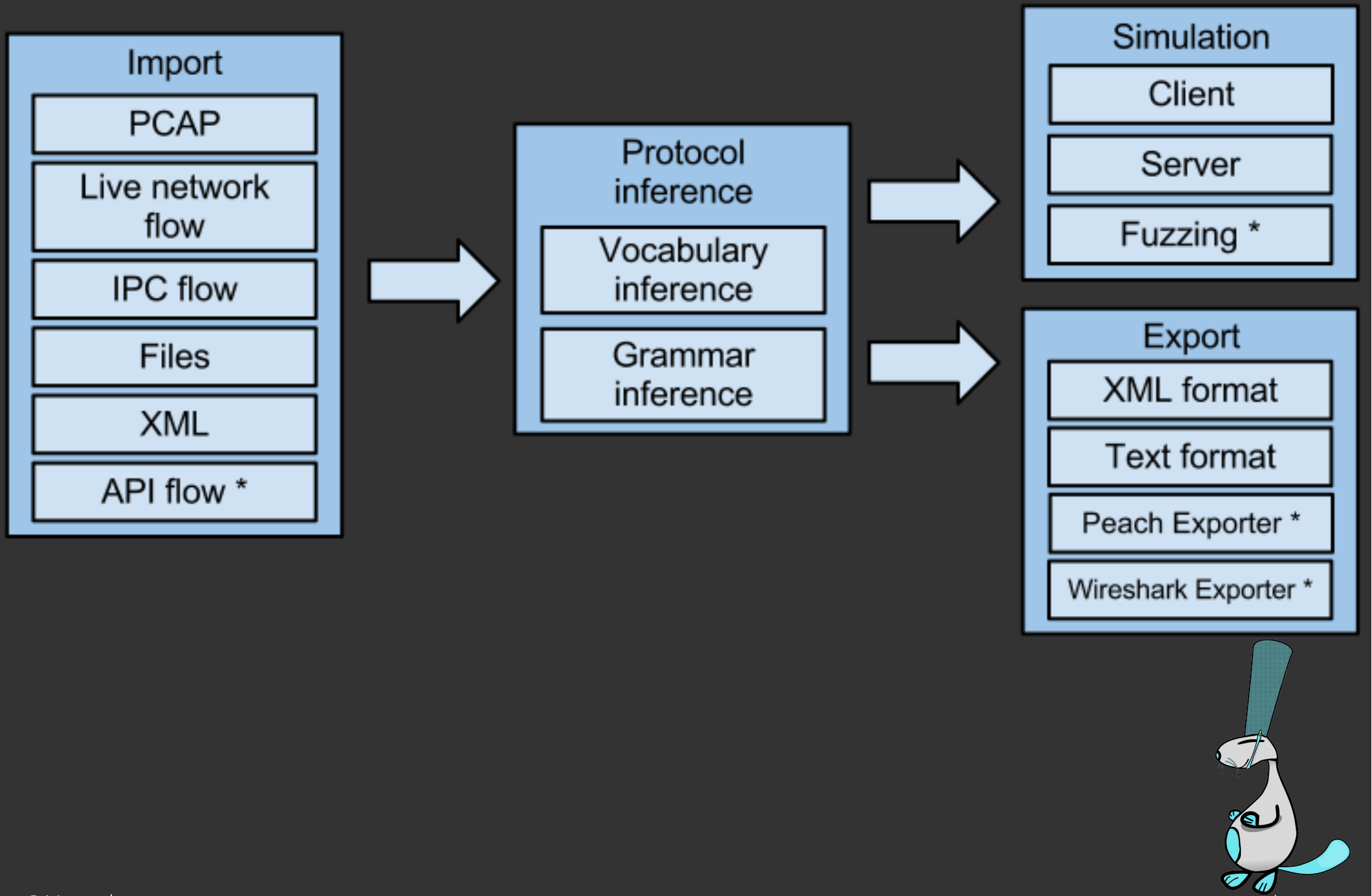
Infer proprietary protocols

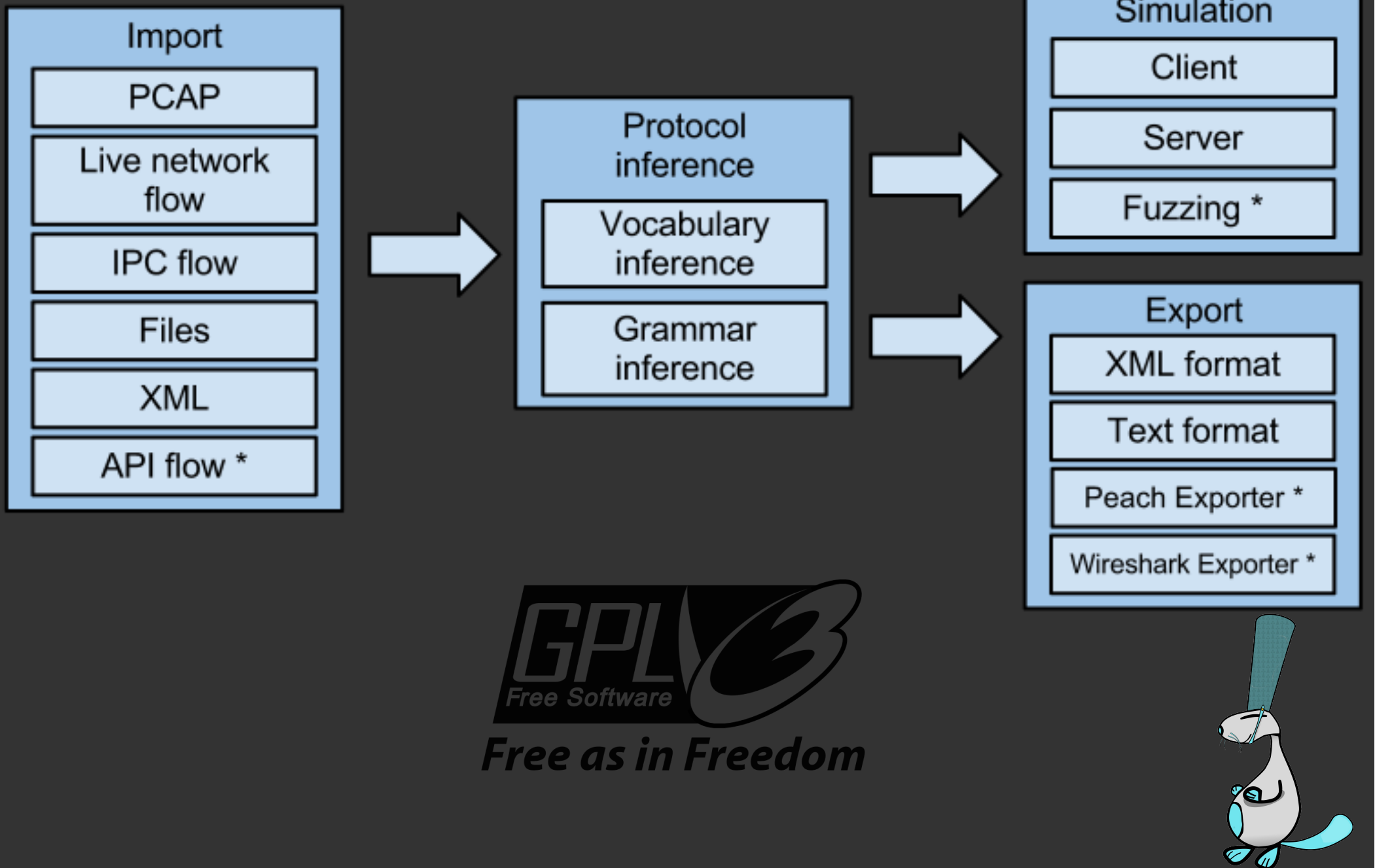
Simulate actors of a communication

- Smart-fuzz targeted implementations






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 *
GPL
Free Software
Free as in Freedom

« State of the art » boundaries

Fuzzing

Language Theory

Netzob

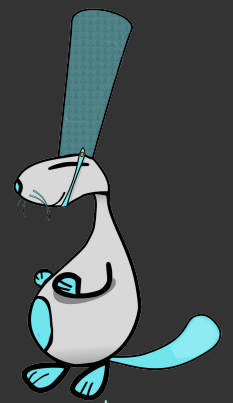
Reverse Engineering

Grammar Inference

Botnet Behavioural Analysis

Sum of human knowledge

The unknown



NEW « State of the art » boundaries

Fuzzing

Netzob

Language Theory

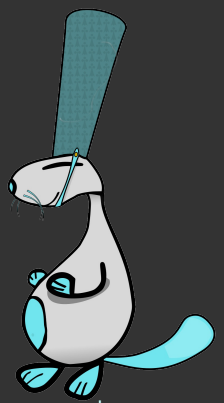
Reverse Engineering

Grammar Inference

Botnet Behavioural Analysis

New sum of human knowledge

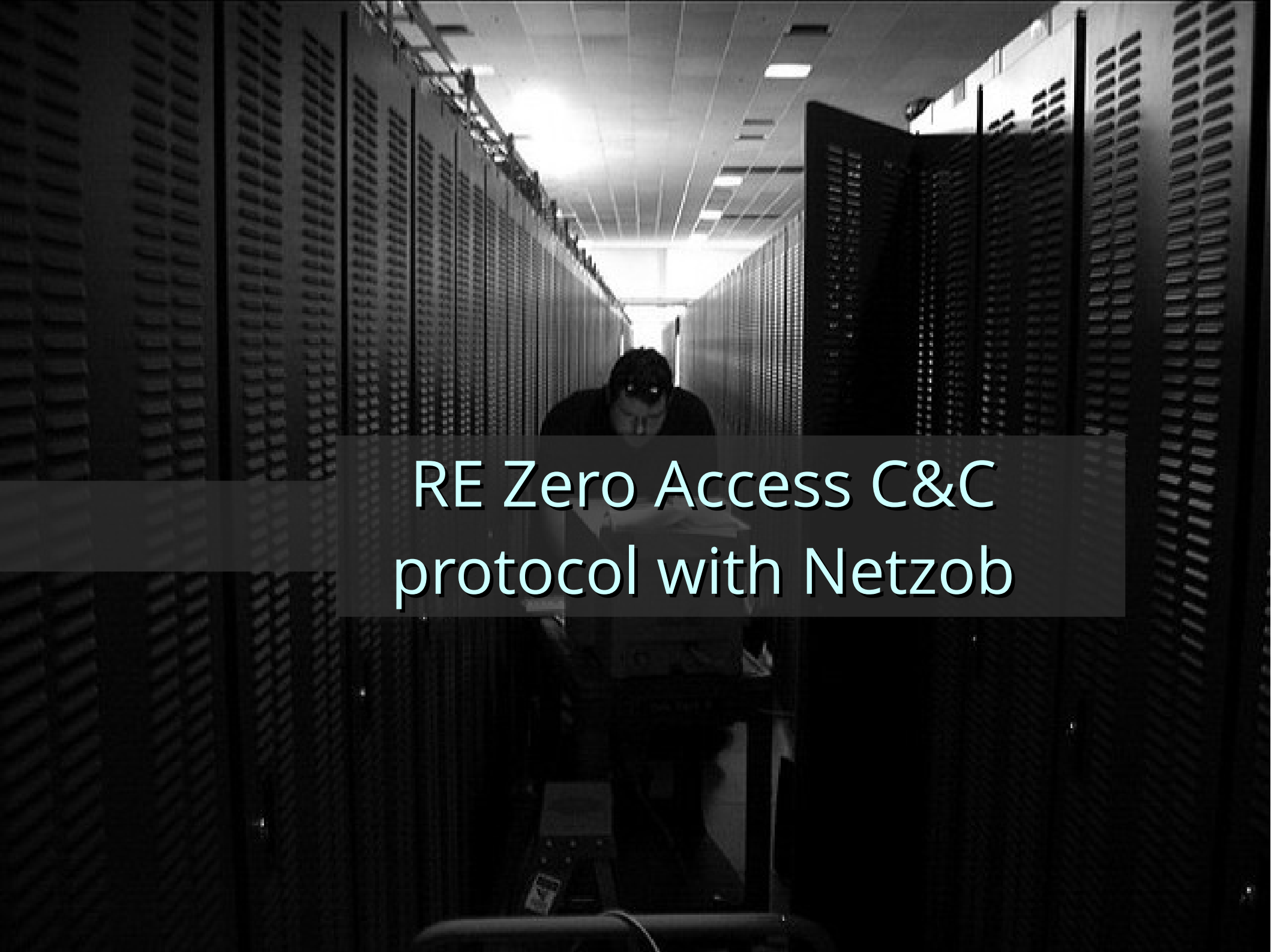
The unknown



@Netzob

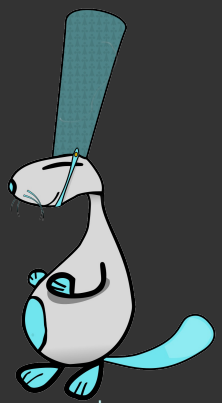
Based on an original idea of Matt Might

netzob.org



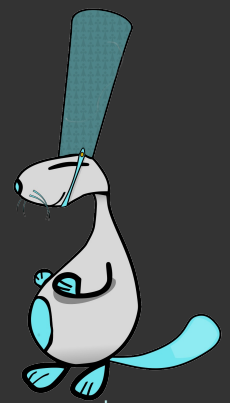
RE Zero Access C&C protocol with Netzob

Requirements

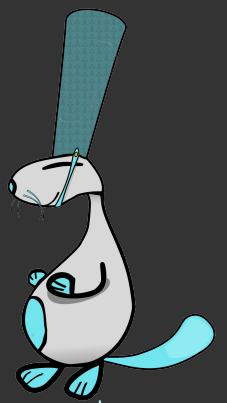


Few real communication traces

ZAccess : some traces were provided by *Kevin McNamee* and from an infected machine

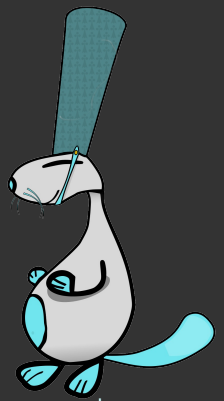


The malware binary



A confined environment

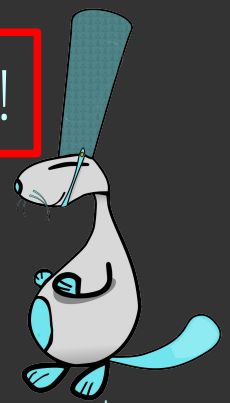
Adapted Virtual Machines + Firewalls + Torify + management system



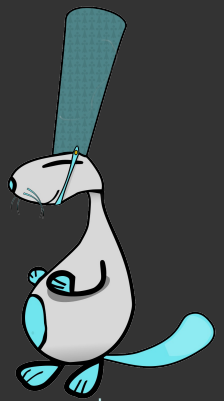
A confined environment

Adapted Virtual Machines + Firewalls + Torify + management system

Warning : Consider legal issues before dealing with this !

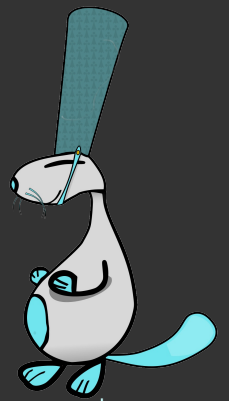


Step 1 : Get messages



Capture dataflows

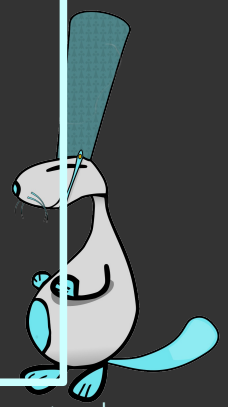
(Network, USB, IPC, API Hooking, Raw files, ...)



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...d.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...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....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.{....;XV

....=.O..?... Qpga("...;...^:....LU..>.'.i.

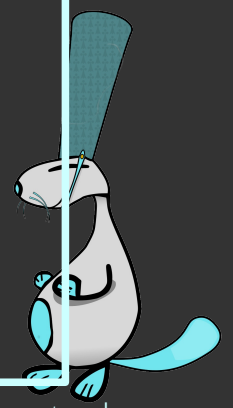
L8.7c....@.....}1iN.7...25t.G.)53.,.....S.....9.'.L5.J.

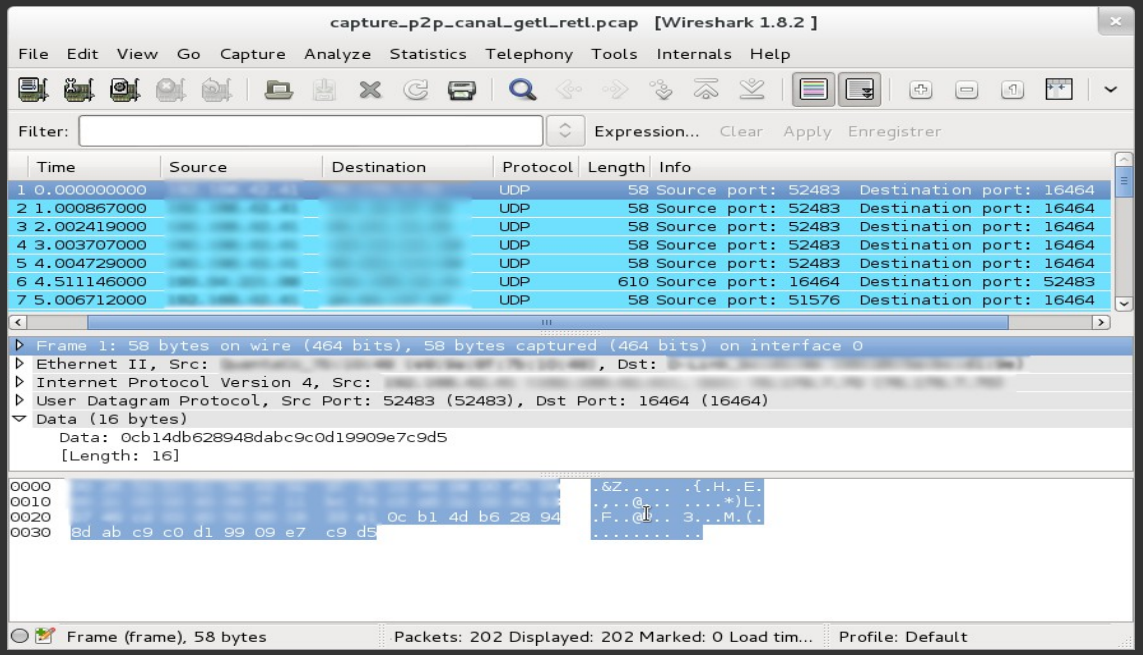
Message 1

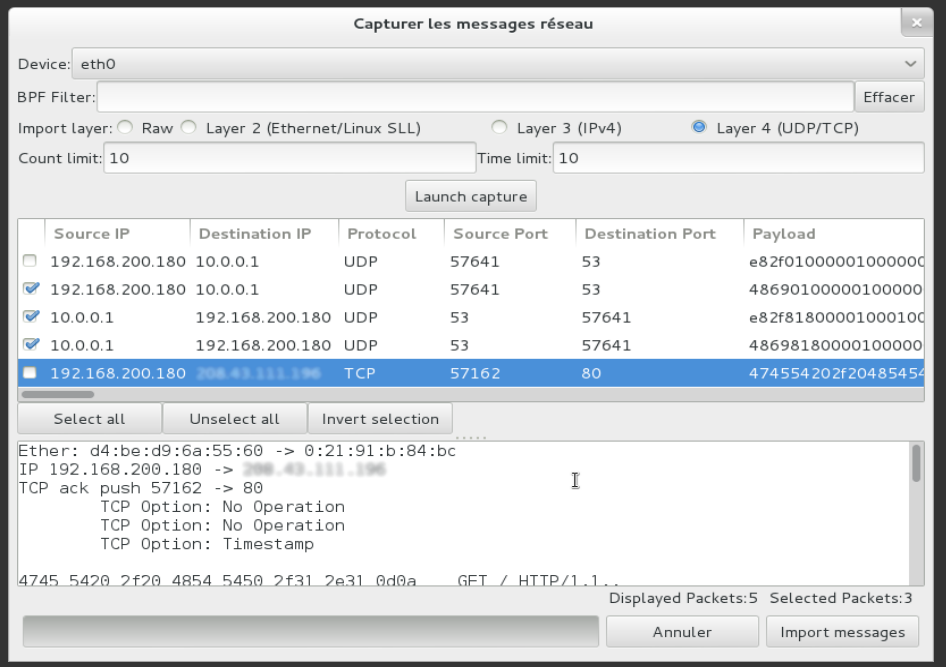
Message 2

Message 3

Message 4







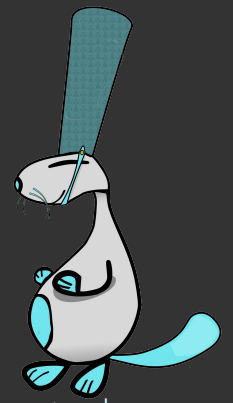
Import

Capture

Netzob framework

Filter imported messages

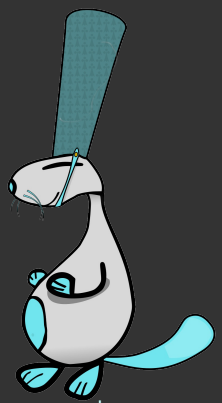
Choose layer of import



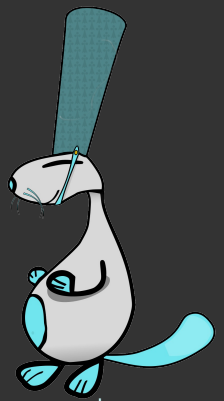
@Netzob

netzob.org

Step 2 : RE vocabulary



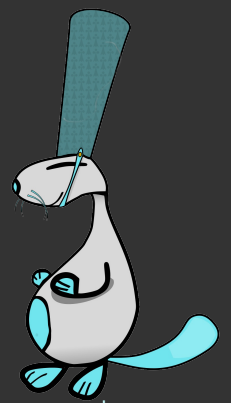
Abstract messages



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

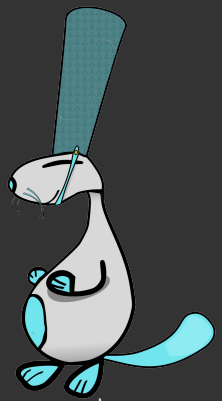
specific to a context

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

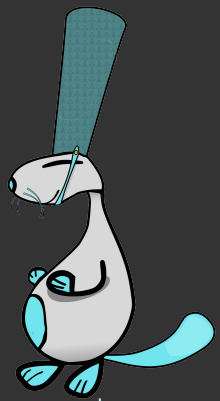


We have to decontextualize messages

and regroup similar ones



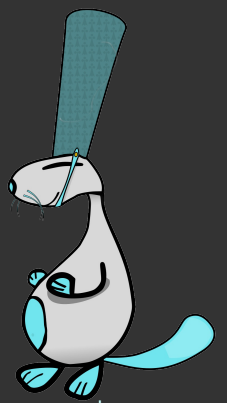
Messages are splitted in **Fields** using



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

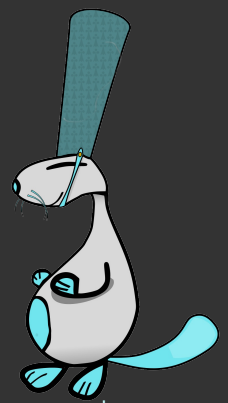
cfa0	4c	5519e43e2e27fa6916313dc89ac77569a00d	4c	38f
cfa0	4c	5519e43e2e27fa6916313dc89ac77569a00d	4c	38f
cfa0	4c	5519e43e2e27fa6916313dc89ac77569a00d	4c	38f

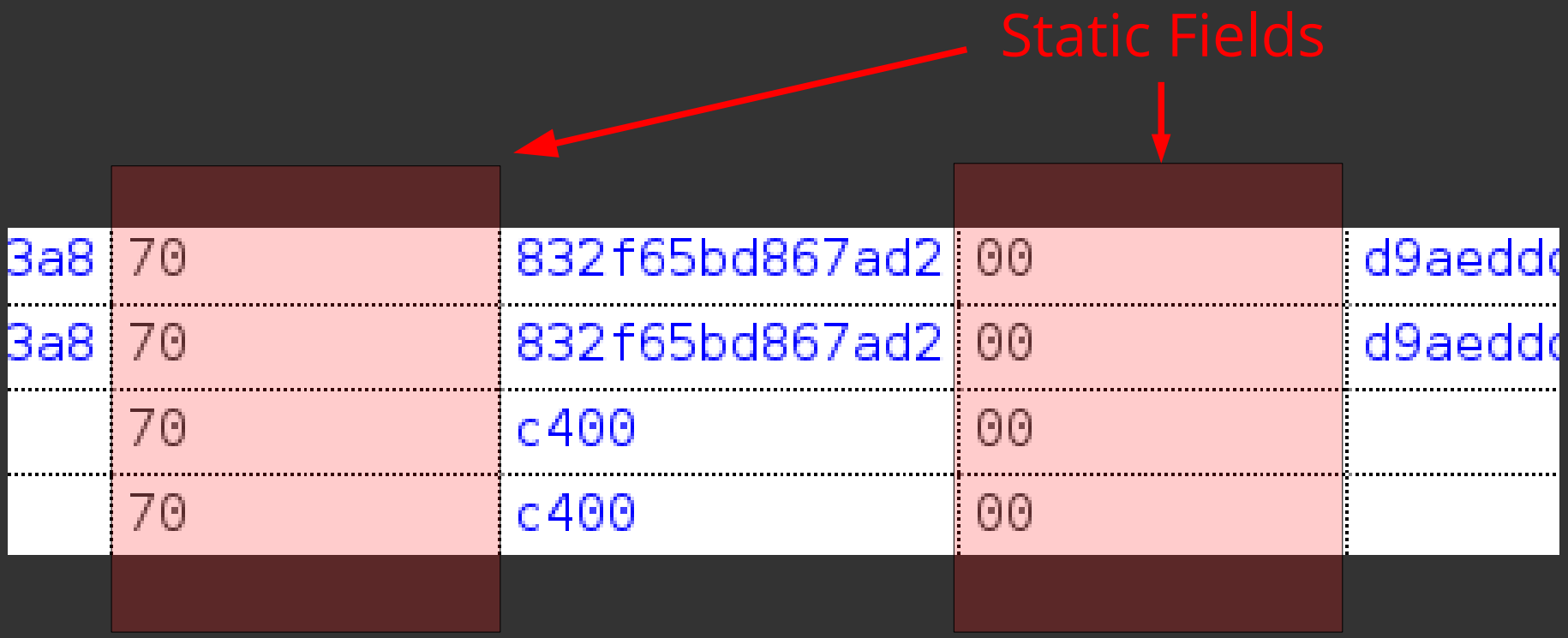
Messages are splitted in **Fields** using

- Simple Alignment
- Delimitor-based Alignment
- Sequence Alignment

Needleman & Wunsch

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

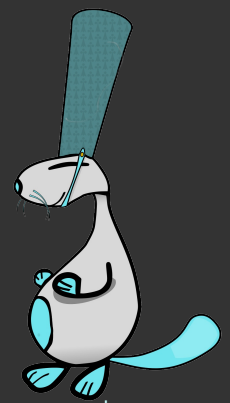



Static Fields
3a8 70 832f65bd867ad2 00 d9aeddc
3a8 70 832f65bd867ad2 00 d9aeddc
70 c400 00
70 c400 00

Static Fields

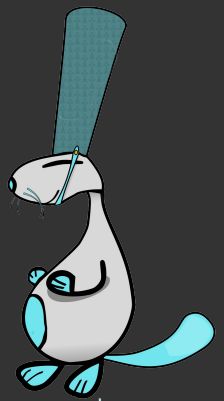
3a8	70	832f65bd867ad2	00	d9aeddc
3a8	70	832f65bd867ad2	00	d9aeddc
	70	c400	00	
	70	c400	00	

Dynamic Fields



Sequence alignment with Needleman-Wunsh

Idea proposed by Bedoe

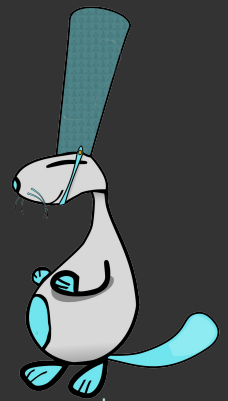


Sequence alignment with Needleman-Wunsh

70 83 2f 65 bd 86 7a d2 00

70 c4 00 00

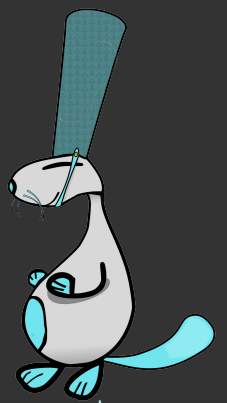
We start with 2 messages



Sequence alignment with Needleman-Wunsh

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

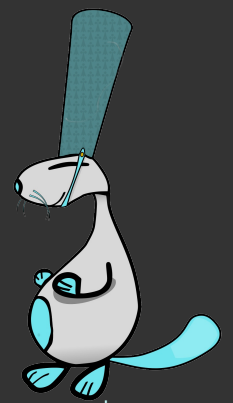
We build a distance matrix



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

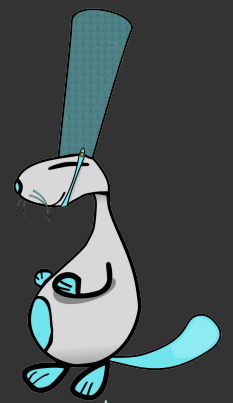
		70	83	2f	65	bd	86	7a	d2	00
	0	0	0	0	0	0	0	0	0	0
70	0	?								
c4	0									
00	0									
00	0									

We fill the matrix with to 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)

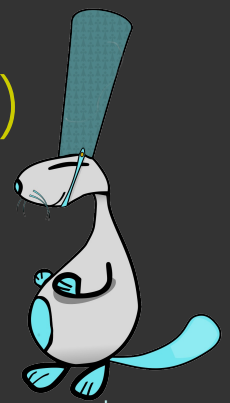


Sequence alignment with Needleman-Wunsh

		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 to the formula:

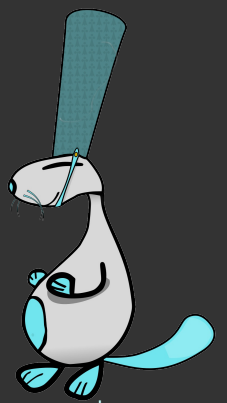
$$M(i,j) = \max(M(i-1, j-1) + S, M(i, j-1) + W, M(i-1, j) + W)$$



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	**30**

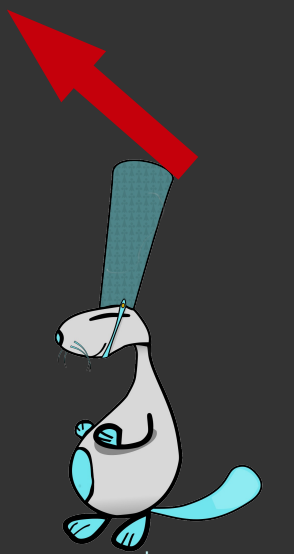
We fill the entire matrix



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	**30**

We do a traceback



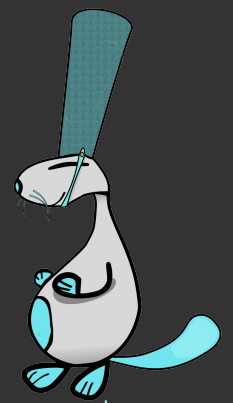
Sequence alignment with Needleman-Wunsh

		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**



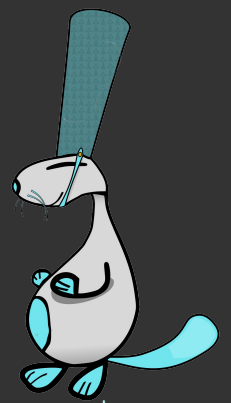
Sequence alignment with Needleman-Wunsh

We finally build a regex for the model

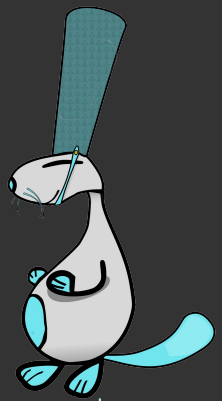
70 83 2f 65 bd 86 7a d2 00

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

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



How to evaluate messages similarities ?

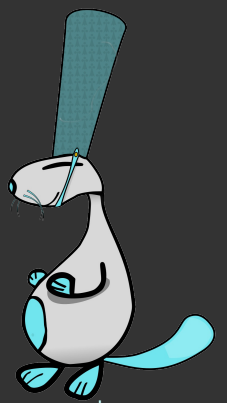


Measure of the **Quality of Symbols**

0 % < Similarity Score < 100 %

Messages have
Nothing in common

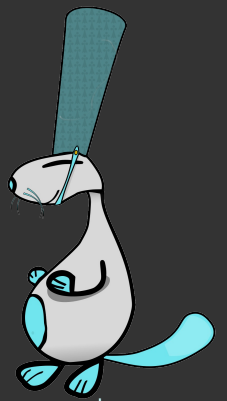
Messages are
identicals



Similarity factors between messages

Currently two factors are used

- F1: ratio of dynamic fields / static bytes
- F2: ratio of common dynamic bytes

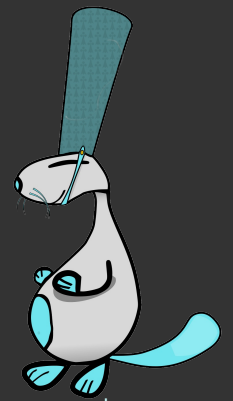


Similarity factors between messages

Currently two factors are used

- F1: ratio of dynamic fields / static bytes
- F2: ratio of common dynamic bytes

The design of Netzob allows for inclusion of future factors



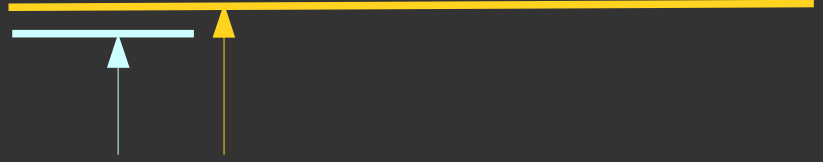
Similarity factors between messages

F1: ratio of dynamic fields / static bytes

F2: ratio of common dynamic bytes

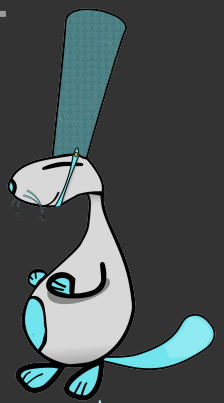
70 83 2f 65 bd 86 7a d2 00

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



F2 = 100/2*7

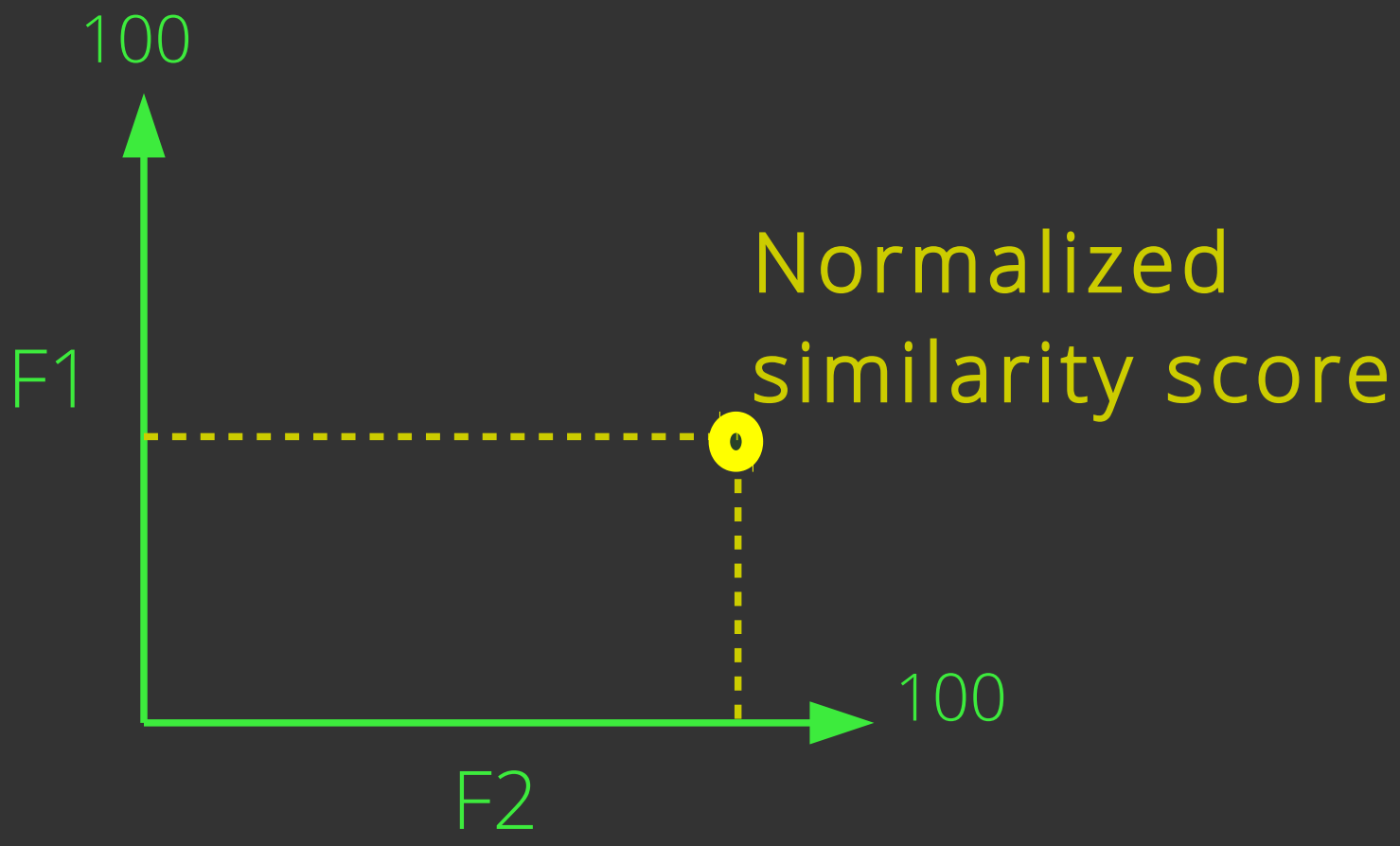
F1 = 100 / (1 + 2) * 2

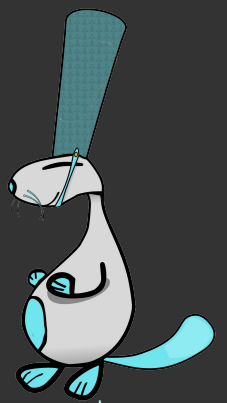


Similarity factors between messages

F1: ratio of dynamic fields / static bytes

F2: ratio of common dynamic bytes

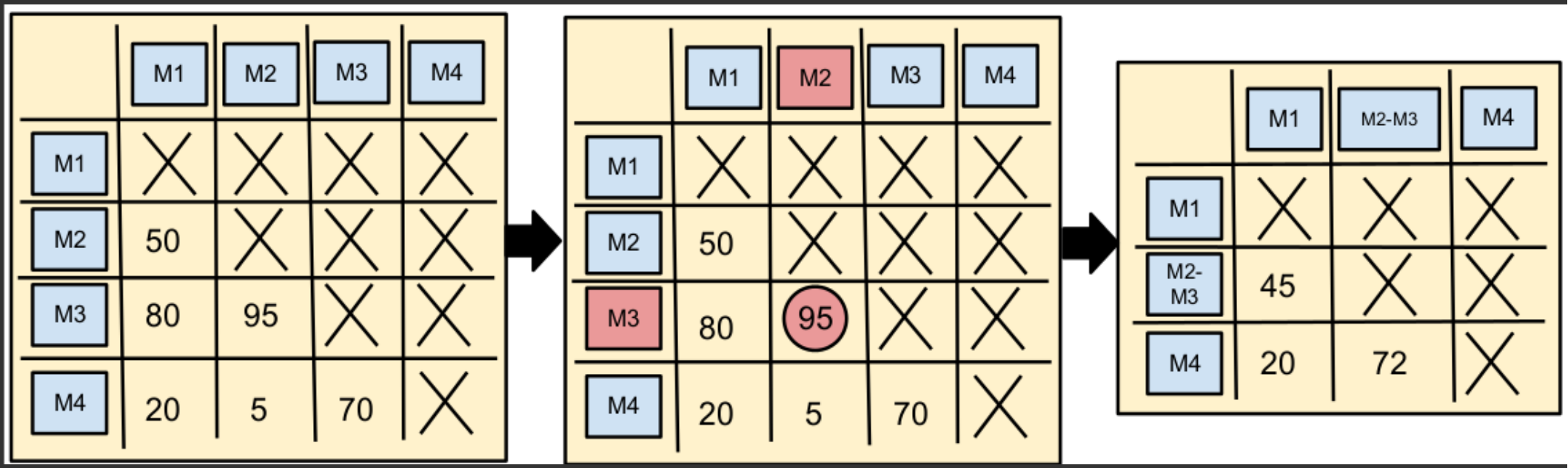




Hierarchical Clustering by similarities:

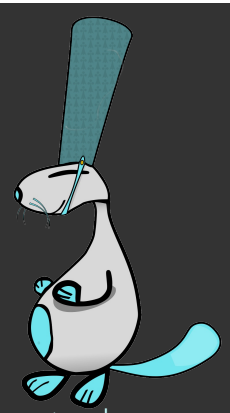
Similarity matrix

UPGMA

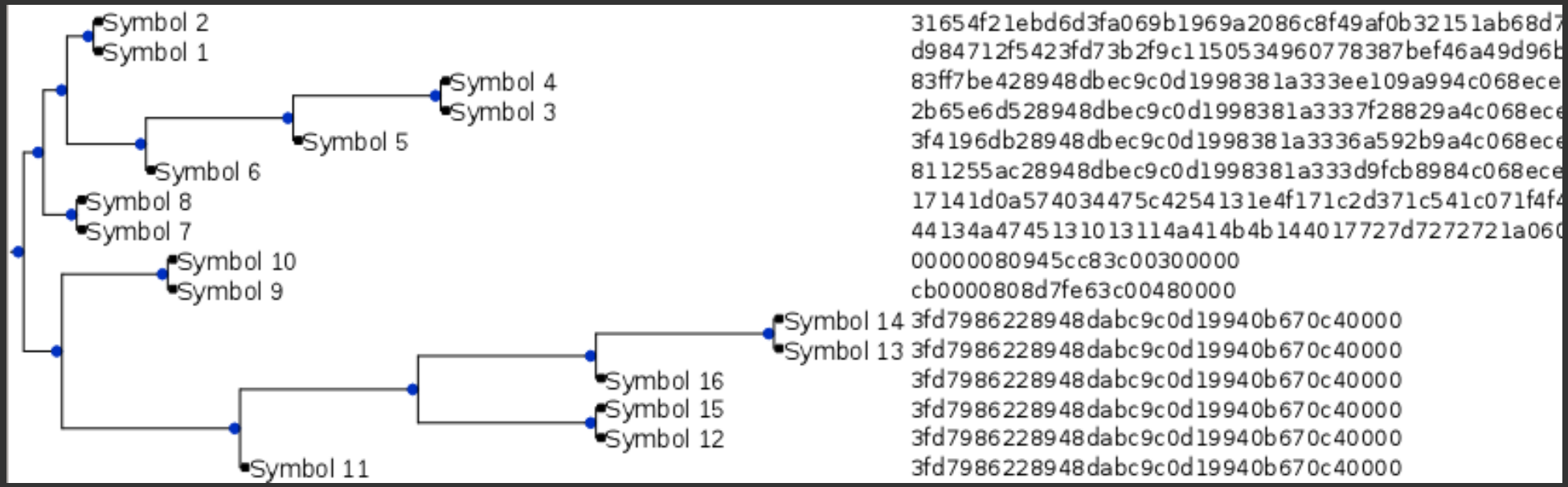


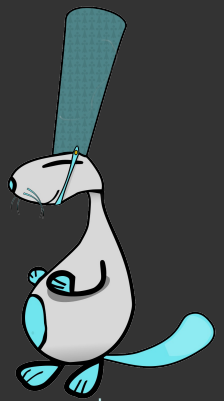
Fill of a similarity matrix

Iteratively merge the 2 most similar messages

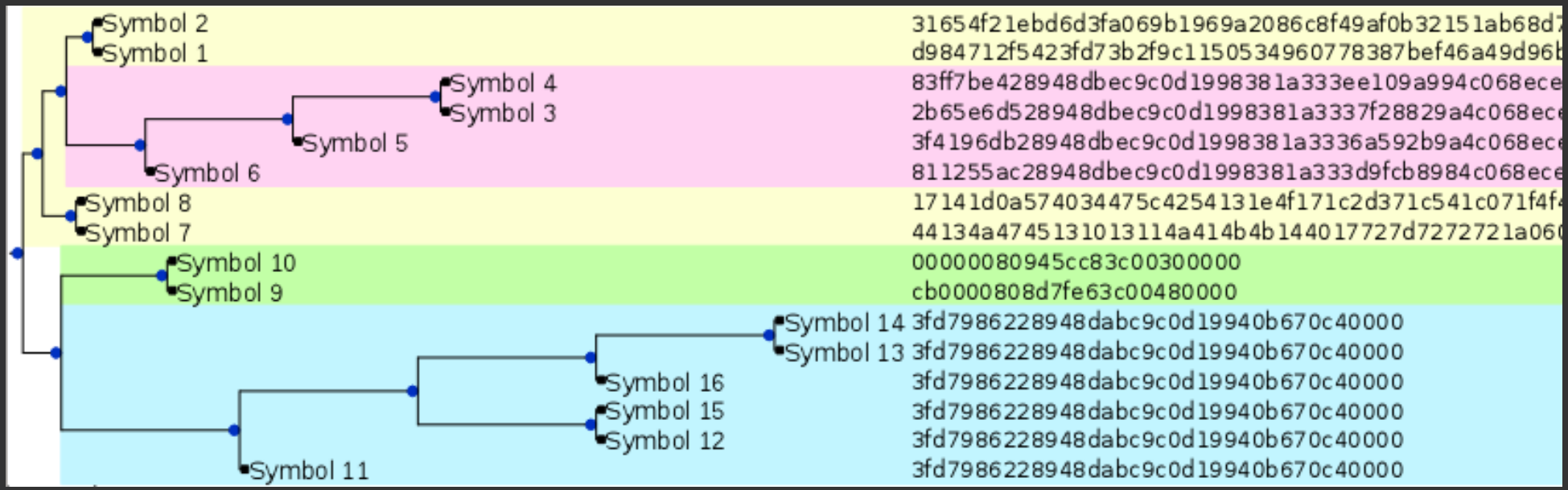


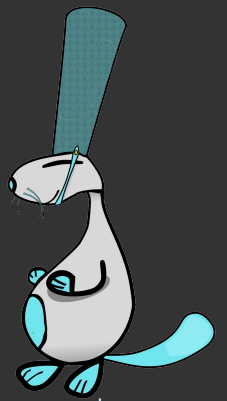
UPGMA creates a similarity tree





UPGMA creates a similarity tree and facilitates clustering





ZAccess Example

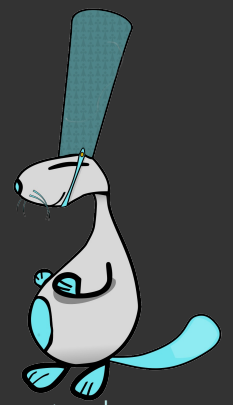
Results of Clustering and Sequence Alignment

Symboles

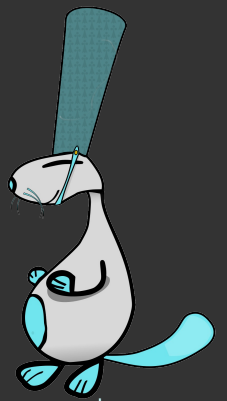
Nom	Message	Champ
Symbol 10	5	3
Symbol 25	8	3
Symbol 23	10	5
Symbol 5	8	12

Symbol 5

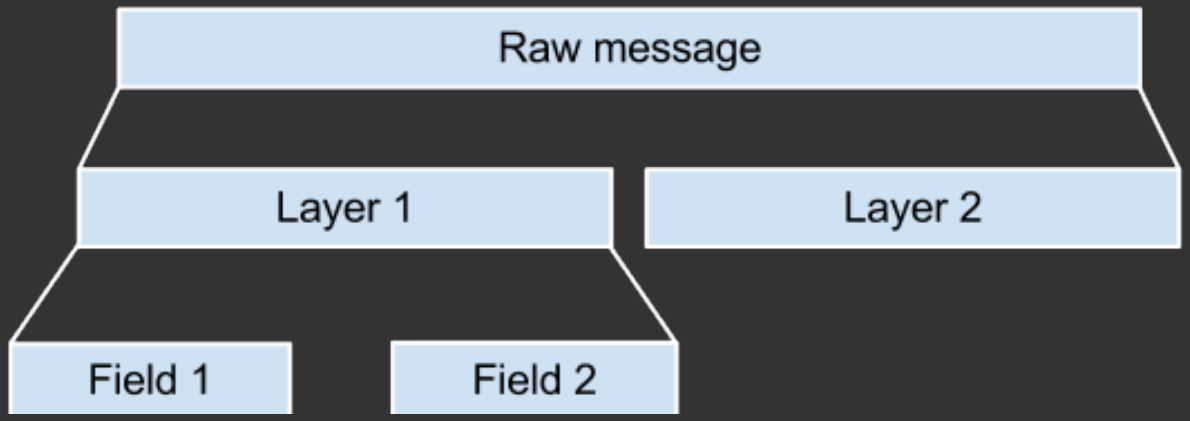
Field 0 (.{8}) hex	Field 1 (4c74657200000000) hex	Field 2 (.{2}) hex	Field 3 (000000) hex	Field 4 (.{8}) hex	Field 5 (00000000) hex	Fie
01a76c9f	4c74657200000000	03	000000	bd584ae8	00000000	d2
9782e4fb	4c74657200000000	06	000000	c725c5ca	00000000	b7
46b9c0a9	4c74657200000000	05	000000	d0b4d6d1	00000000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



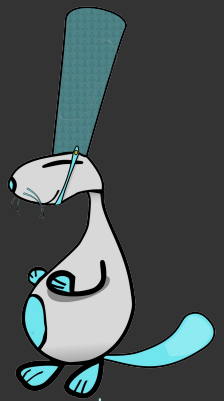
Abstract fields



Message format model

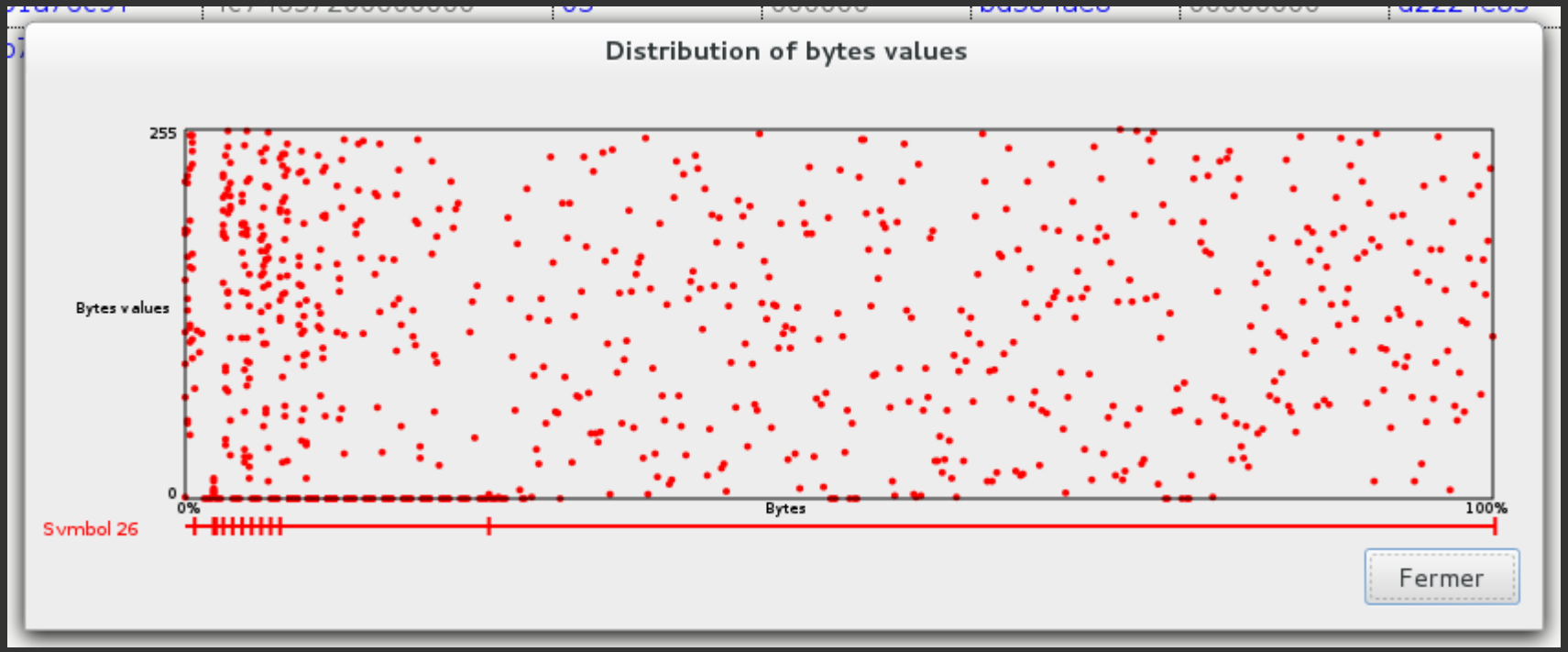


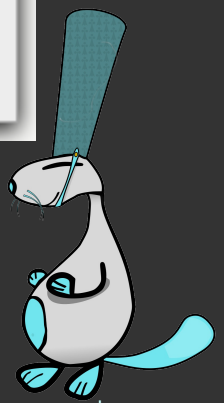
Allows multiple partitionment strategies per symbols



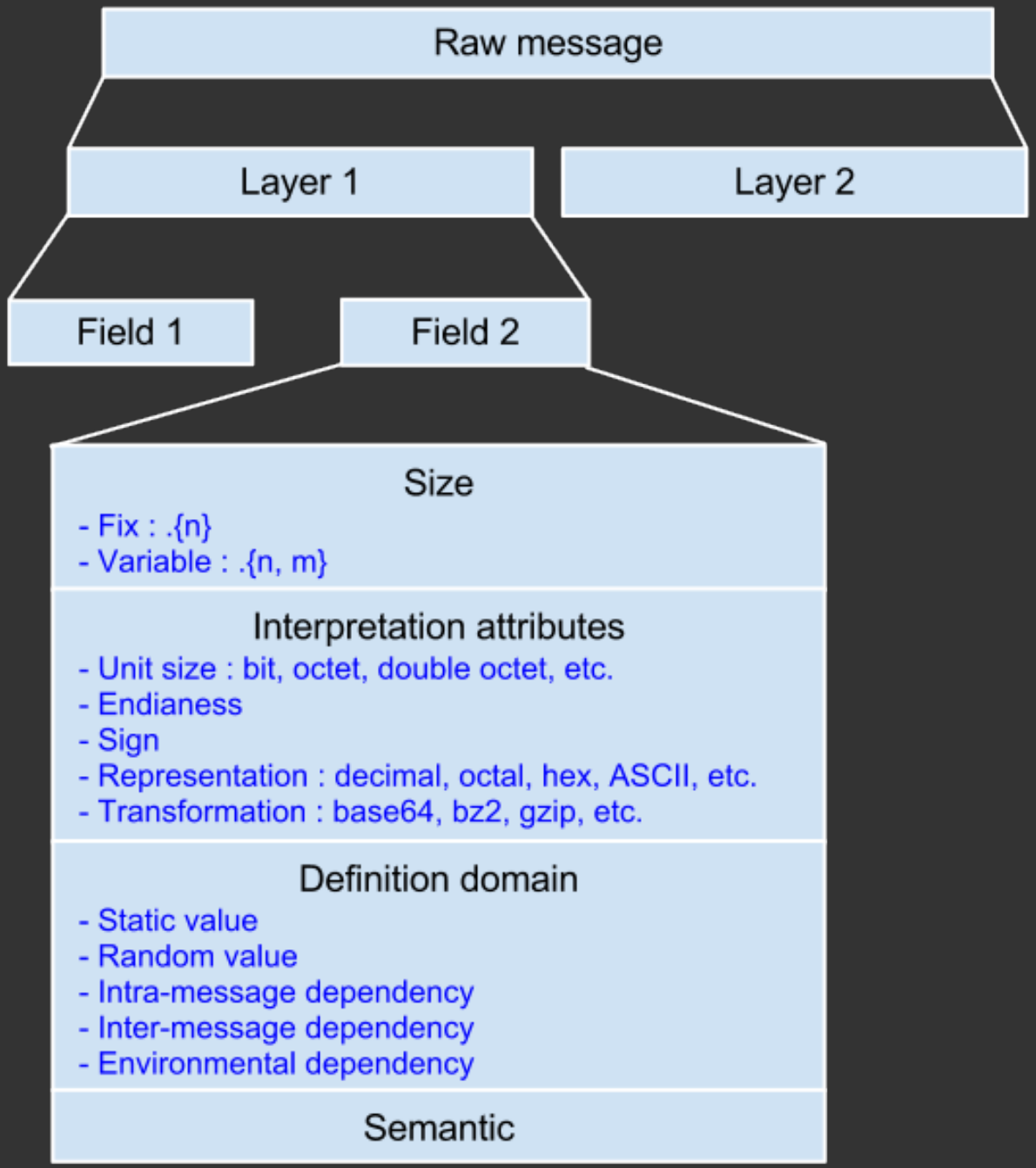
ZAccess Example

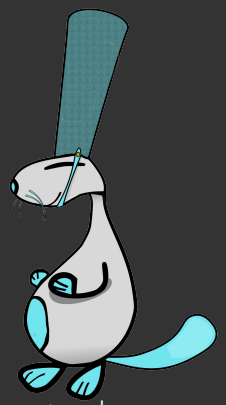
The bytes distribution helps to identify multiple partitionment strategies



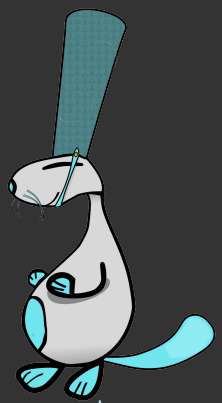


Full message format model



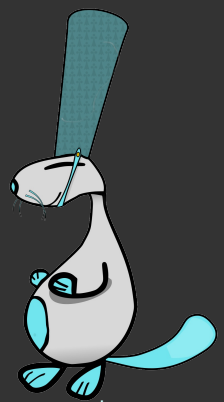


Transformations



The idea: transform raw bytes into application-level bytes

- Applied either on messages, layers or fields
- Examples of provided functions: base64, gzip, bz2



Adding a custom transformation function

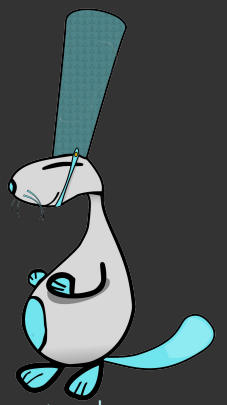
Ex: ZeroAccess XOR-based obfuscation

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)
```



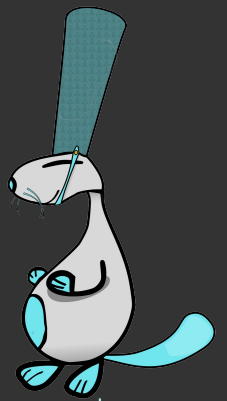
Adding a custom transformation function

Ex: ZeroAccess XOR-based obfuscation

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

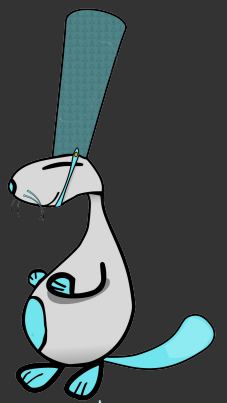
87ee1533	4c7465720000000010000000	8b4e2efc	00000000
78ab751d	4c7465720000000010000000	bd71e2fd	00000000
be33b34a	4c7465720000000010000000	ffffffff	00000000

Search for relations



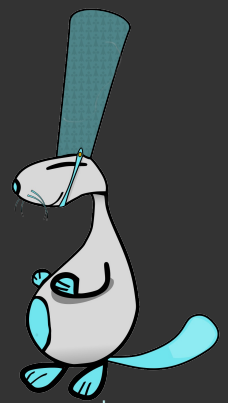
Binary ID, Affiliate ID,
Filenames, *etc.*

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

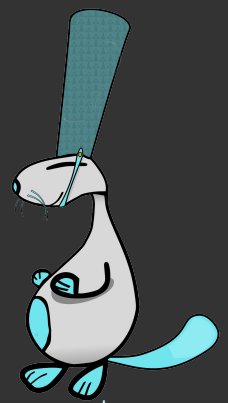


« Inter-Symbol » and « Intra-Symbol » relations

→ Size Fields, CRCs, *etc.*

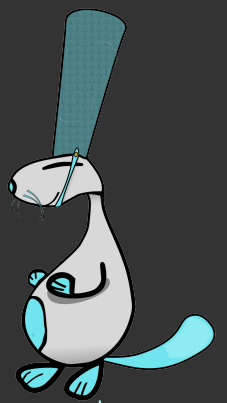


Correlate field's size, values (CRCs,...) with the
« Maximal Information Coefficient » (M.I.N.E. by M.I.T.)



Correlate field's size, values (CRCs,...) with the
« Maximal Information Coefficient » (M.I.N.E. by M.I.T.)

Qualify correlated fields

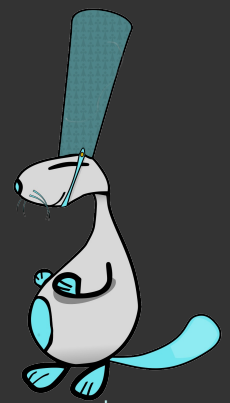


Search for closest pairs :

- Measure dependence between each pairs
- rank pairs by their score

Good Points :

- Fast
- Search for dependances
  - Detect linear, non-linear, periodic relations, ...
- Support Noisy datasets

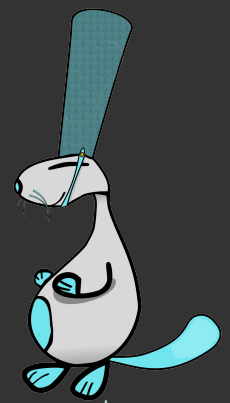


Search for closest pairs :

- Measure dependence between each pairs
- rank pairs by their score

Good Points :

- Fast
- Search for dependances
  - Detect linear, non-linear, periodic relations, ...
- Support noisy datasets



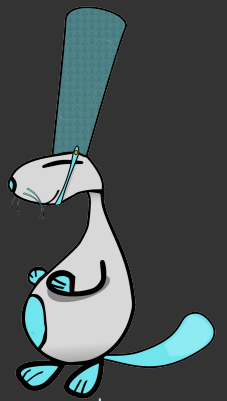
Generate Pairs of data :

Simple way :

- Value of each field
- Size of each dynamic field

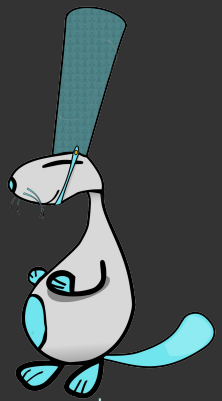
Add more :

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

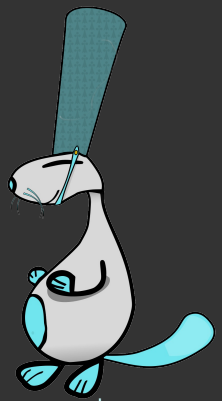


« Environmental » dependencies

Search for messages' meta-informations in data

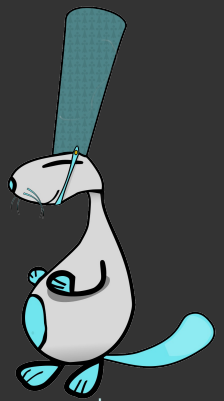


Step 3 : RE grammar



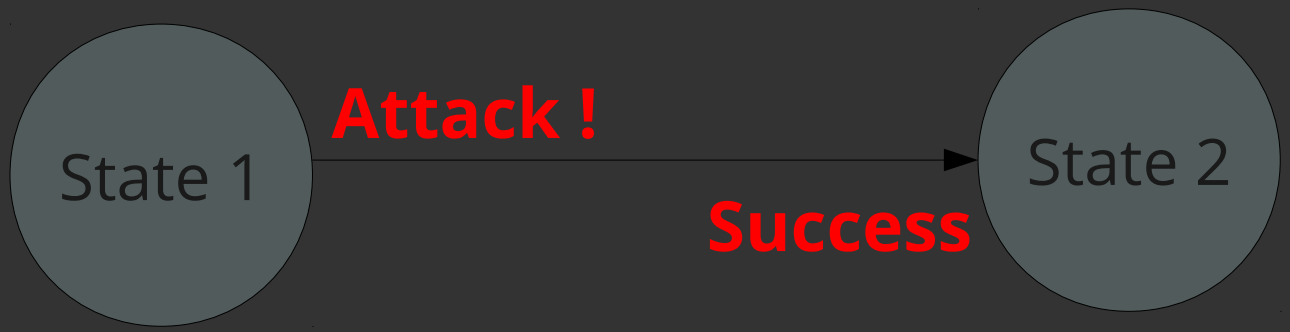
Sequence of valid exchanged symbols.

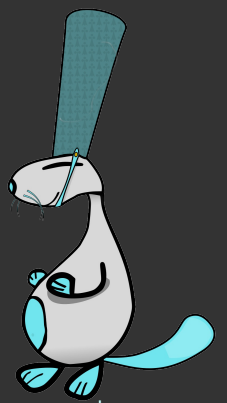
→ IO Automata



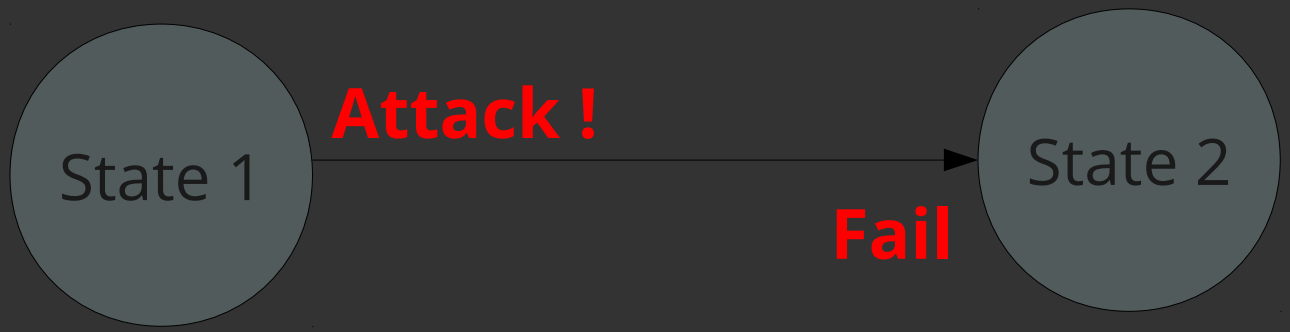
Sequence of valid exchanged symbols.

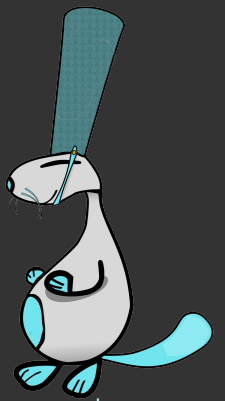
→ IO Automata





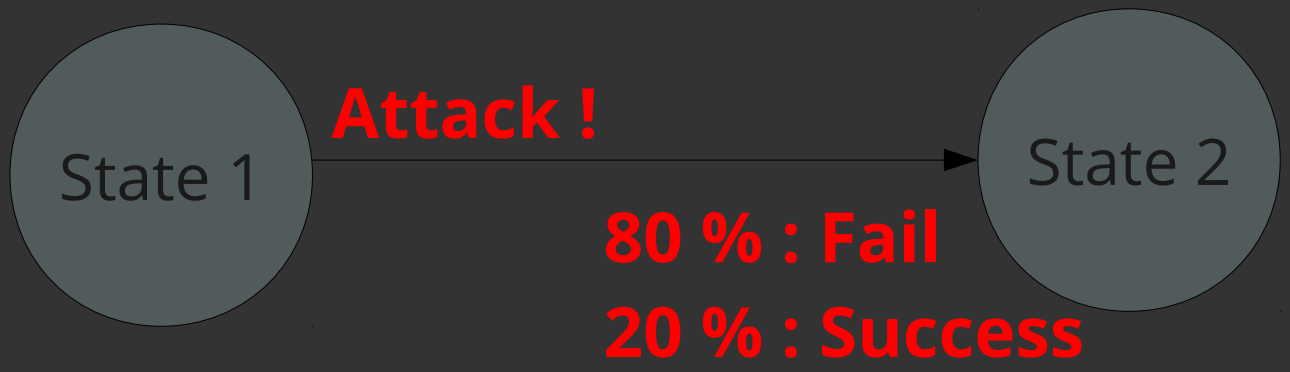
But answers depends on the environment

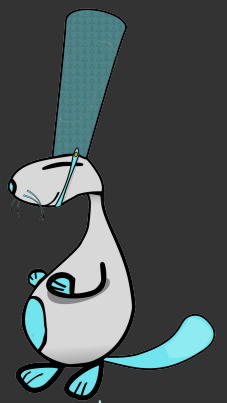




Our model (**SMMDT**)

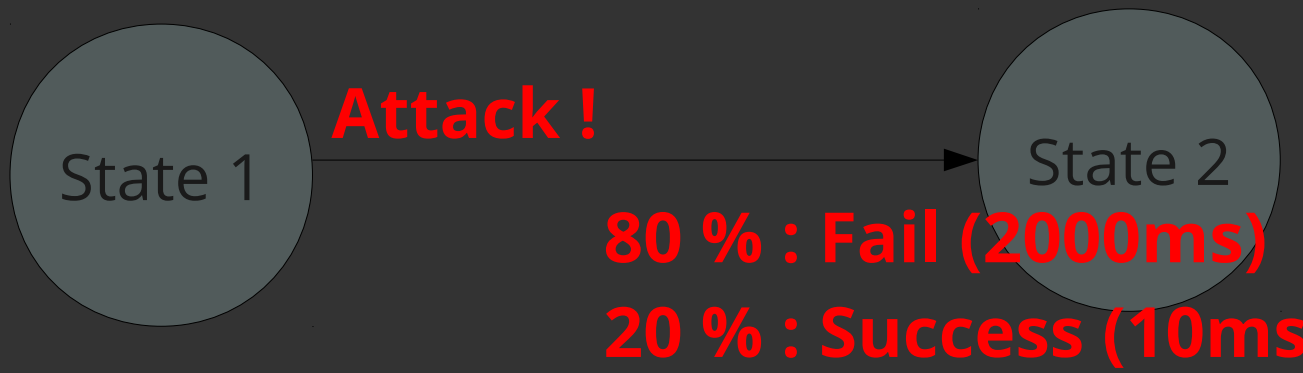
→ Add probabilities on output messages

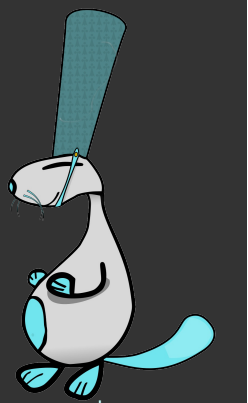




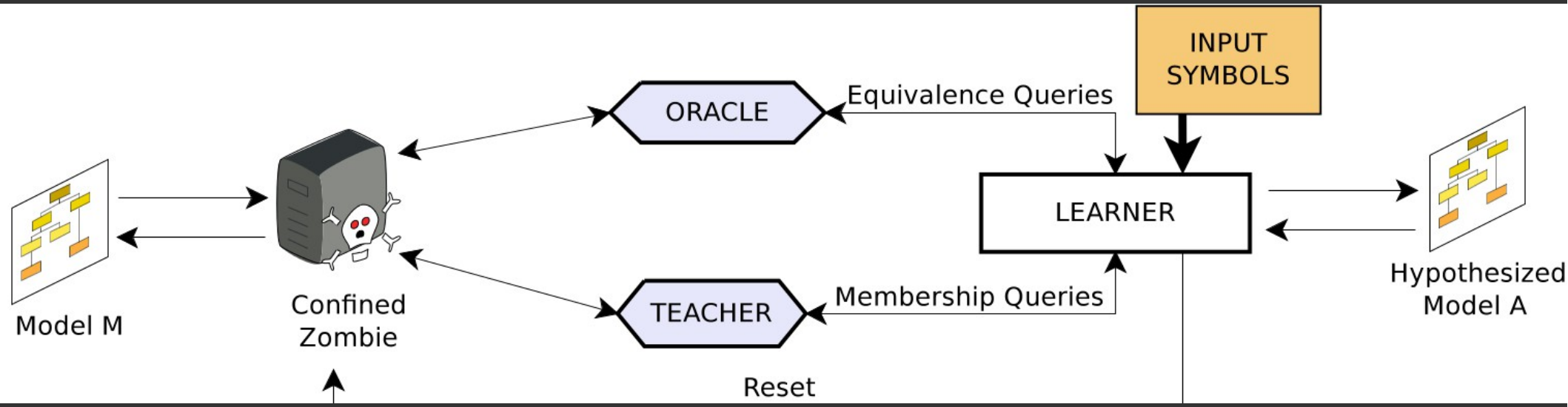
Our model (**SMMDT**)

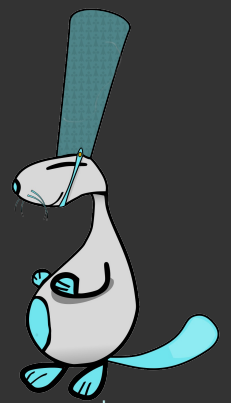
→ Add the « reaction time »





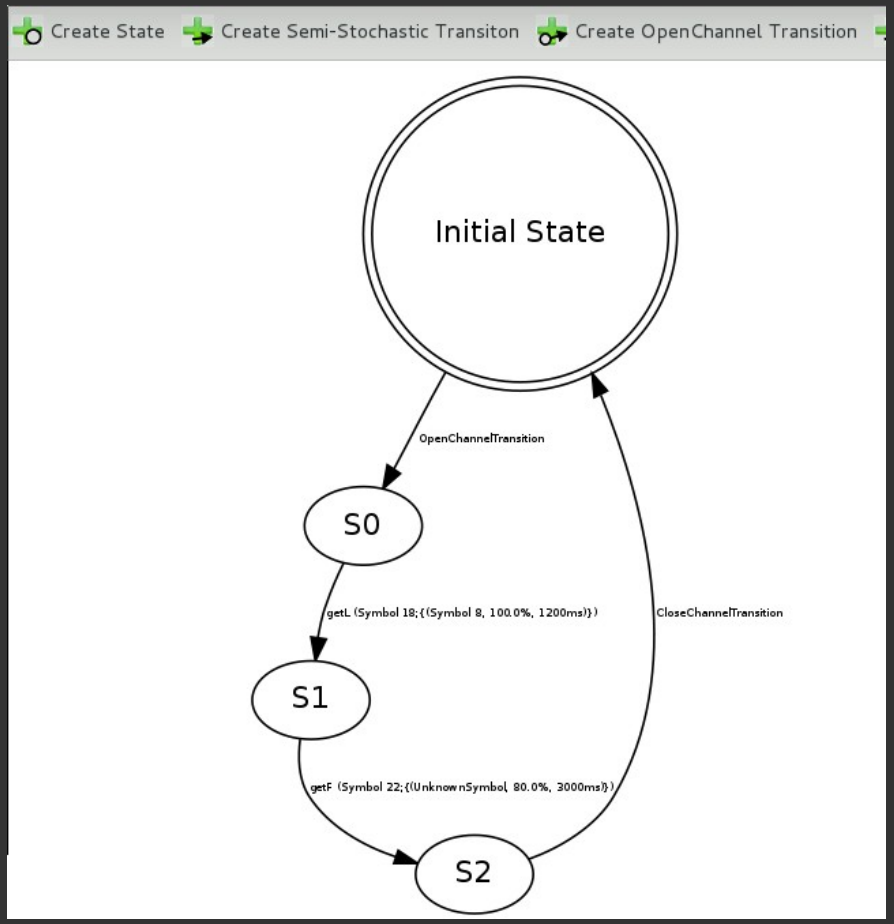
Active Grammatical Inference Process
→ Angluin L*a Algorithm

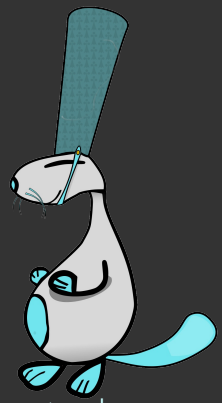




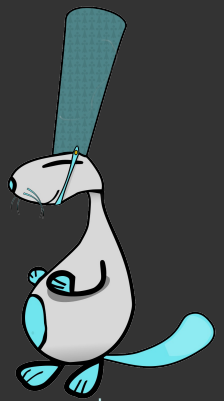
Active Grammatical Inference Process

→ Angluin L*a Algorithm



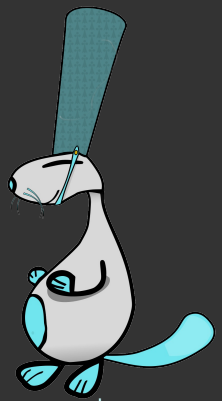


Recaps on ZeroAccess protocol



Recaps on ZA protocol

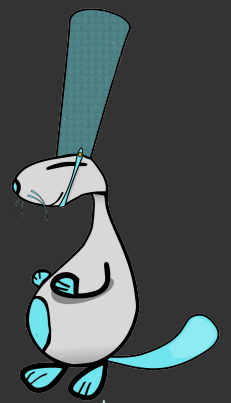
- At least 2 versions
- Multiple P2P management messages
  - « GetL », « RetL », « GetF », ...
  - Share common format
- Low-encryption
- UDP & TCP connections
  - UDP for messages (port 16464)
  - TCP for data
  - Hard coded Bootstrap Peers
    - Ex : 92.47.102.2, (...), 216.211.181.226



Generating traffic...

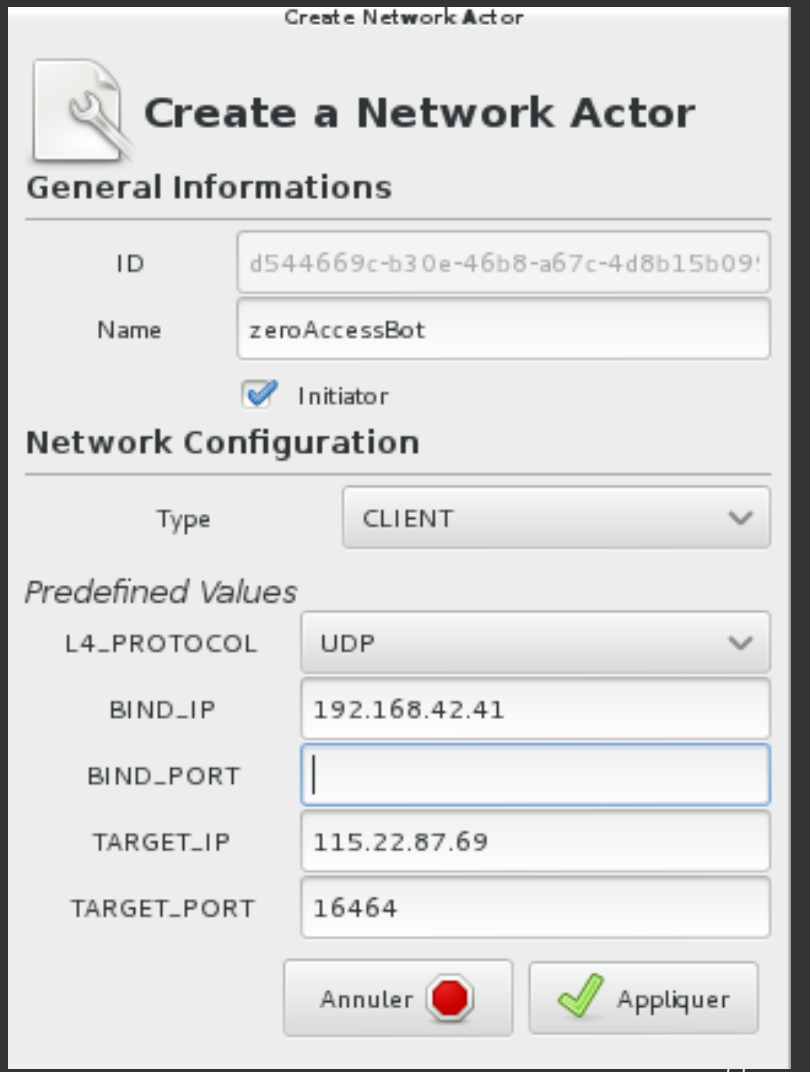
Netzob can generate traffic that:

- Follows the inferred message format
- Respects the state machine



Emulation of different kind of actors and flows

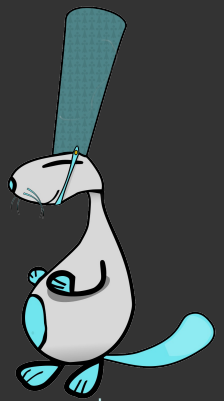
- Client ↔ Real server implementation
- Server ↔ Real client implementation
- Both client(s) and server



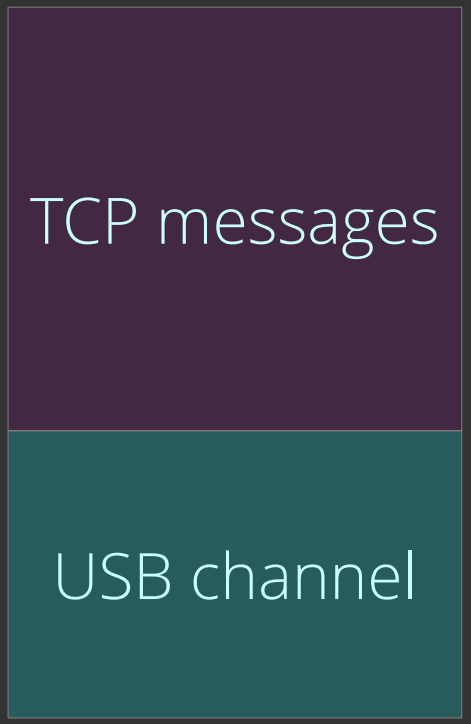
Distinction between

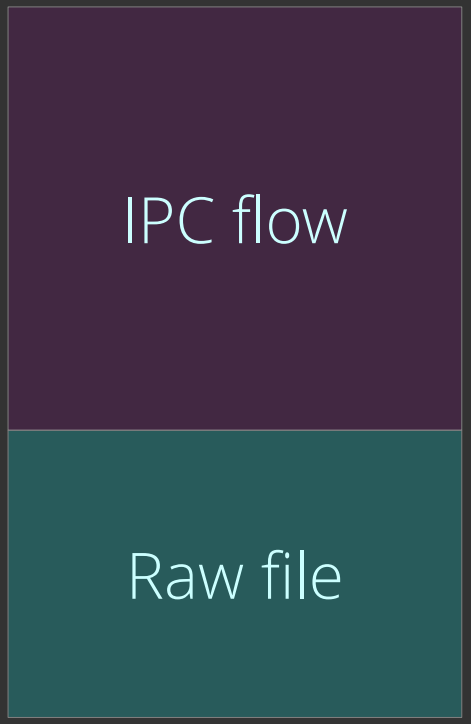
- Client / server
- Initiator / responder of the opening channel

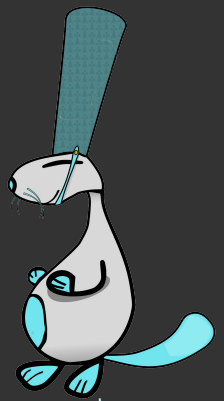
Ex : TLS with TCP session initiated by the server



Abstraction from the communication channel

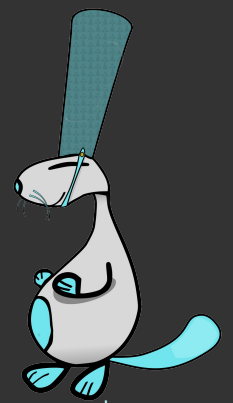




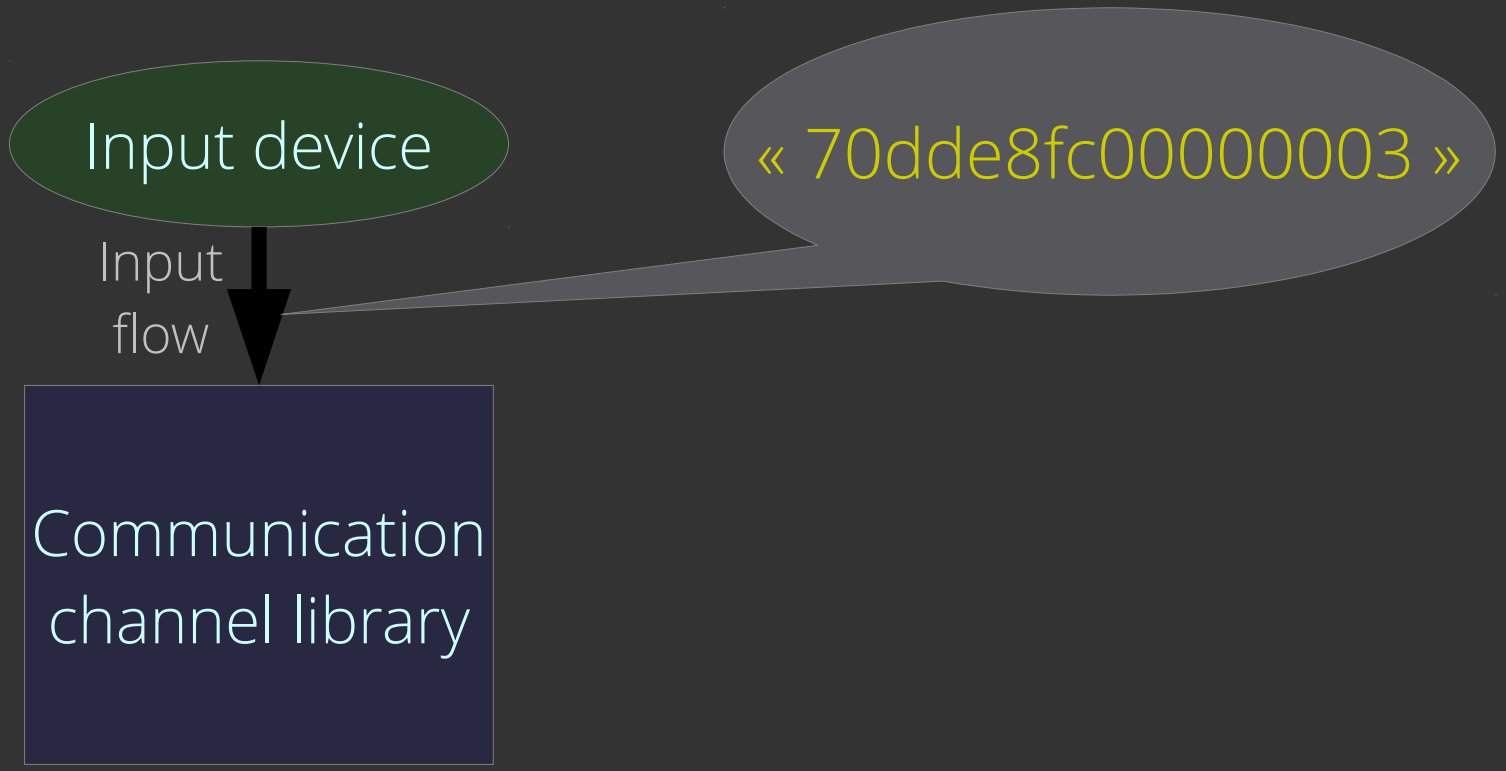


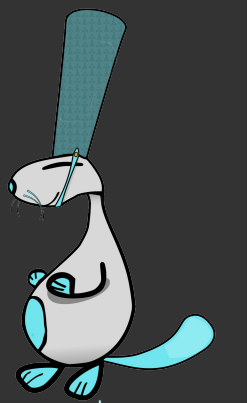
Memory mechanism

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

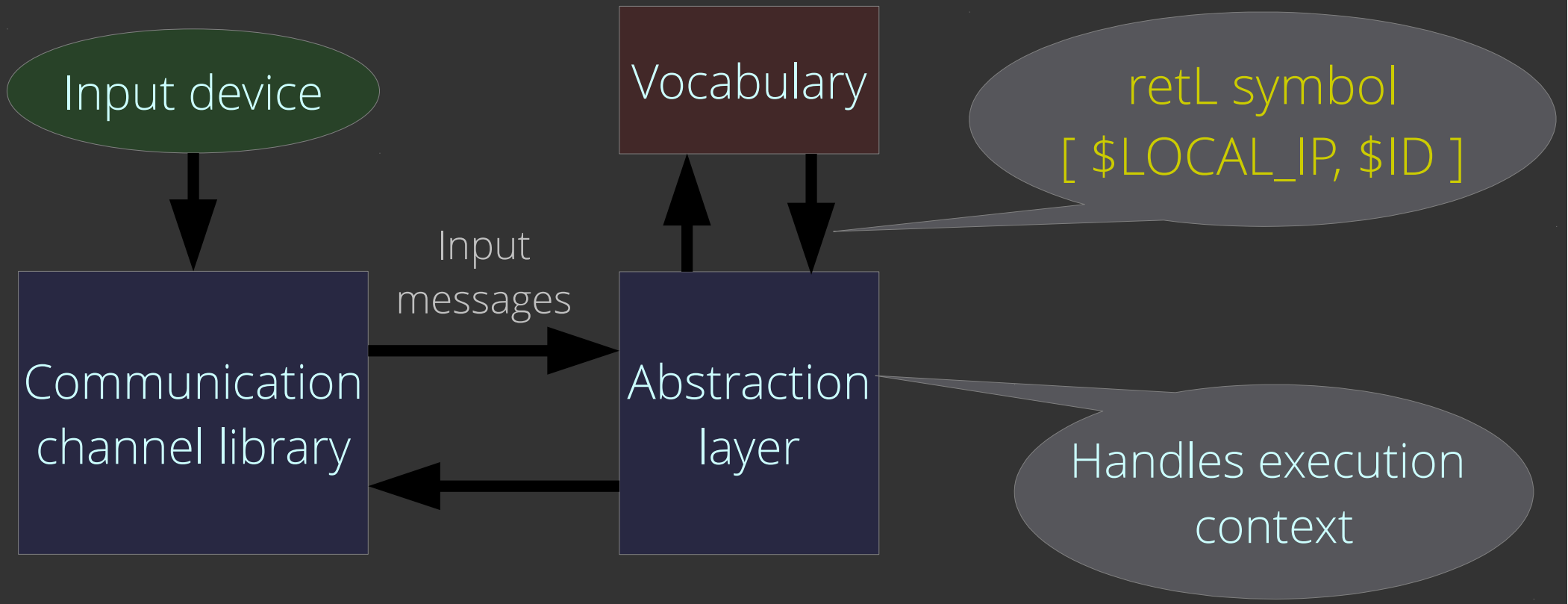


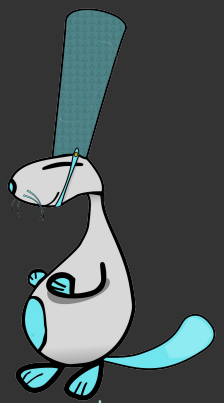
Abstraction and contextualization principles



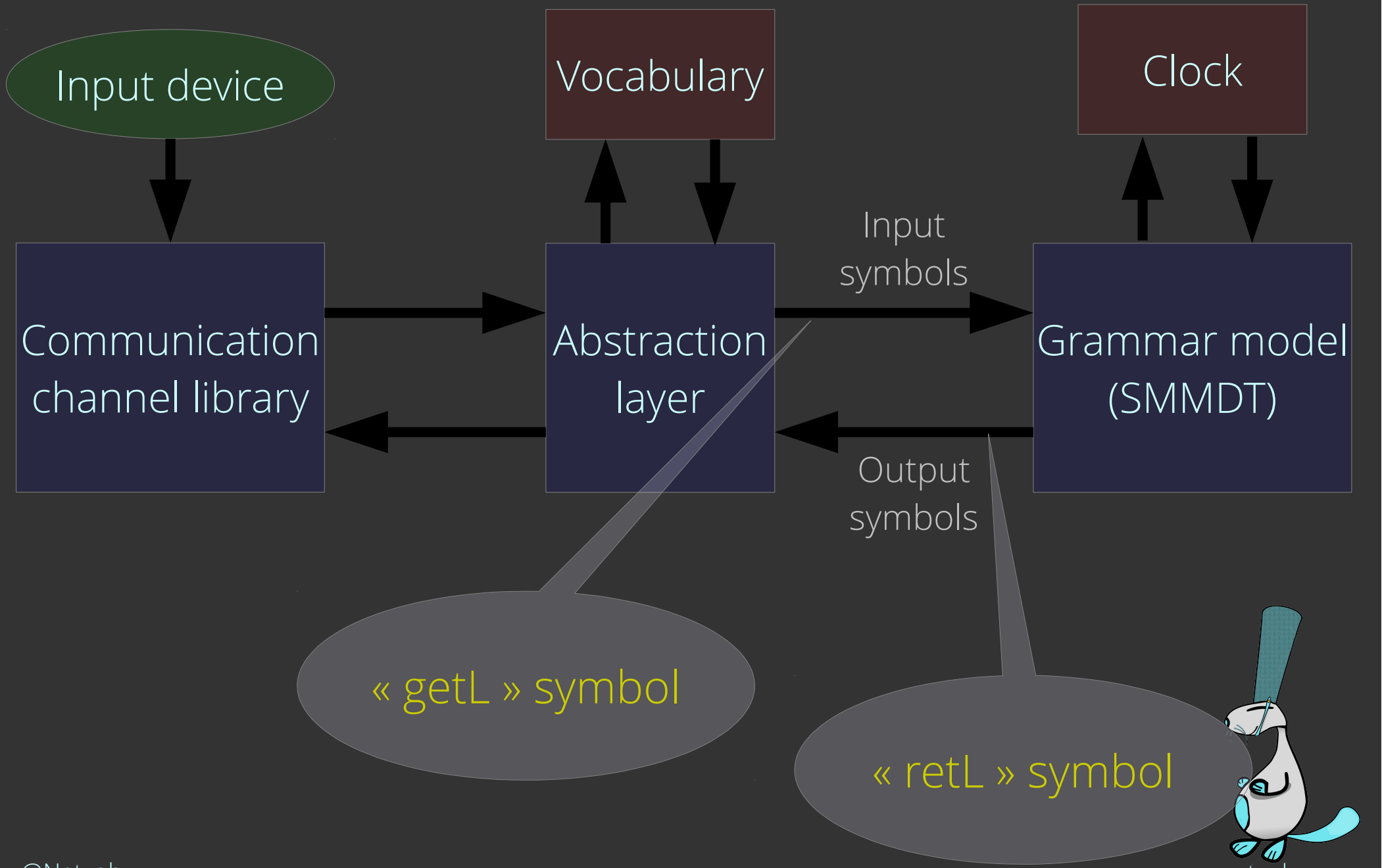


Abstraction and contextualization principles

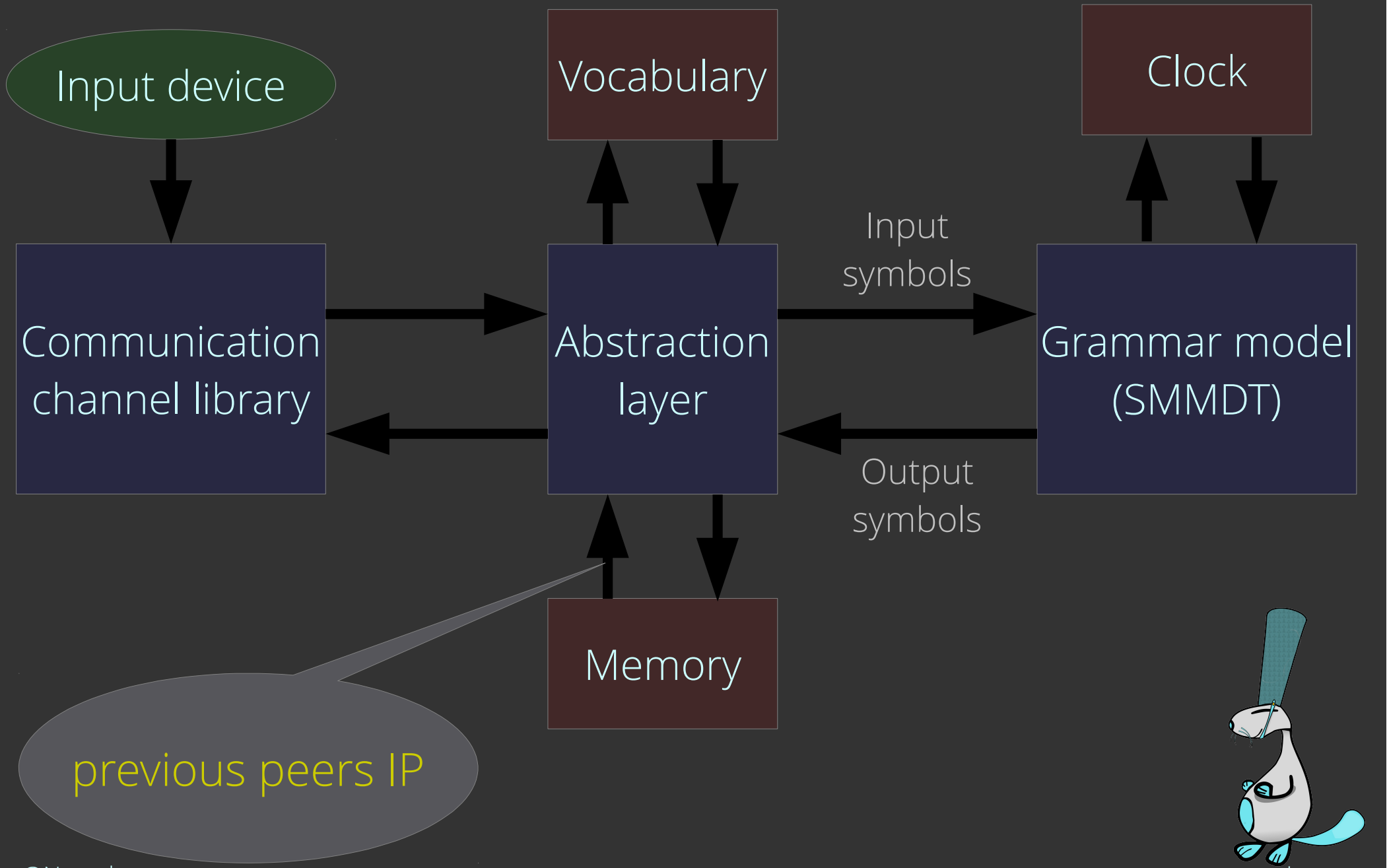




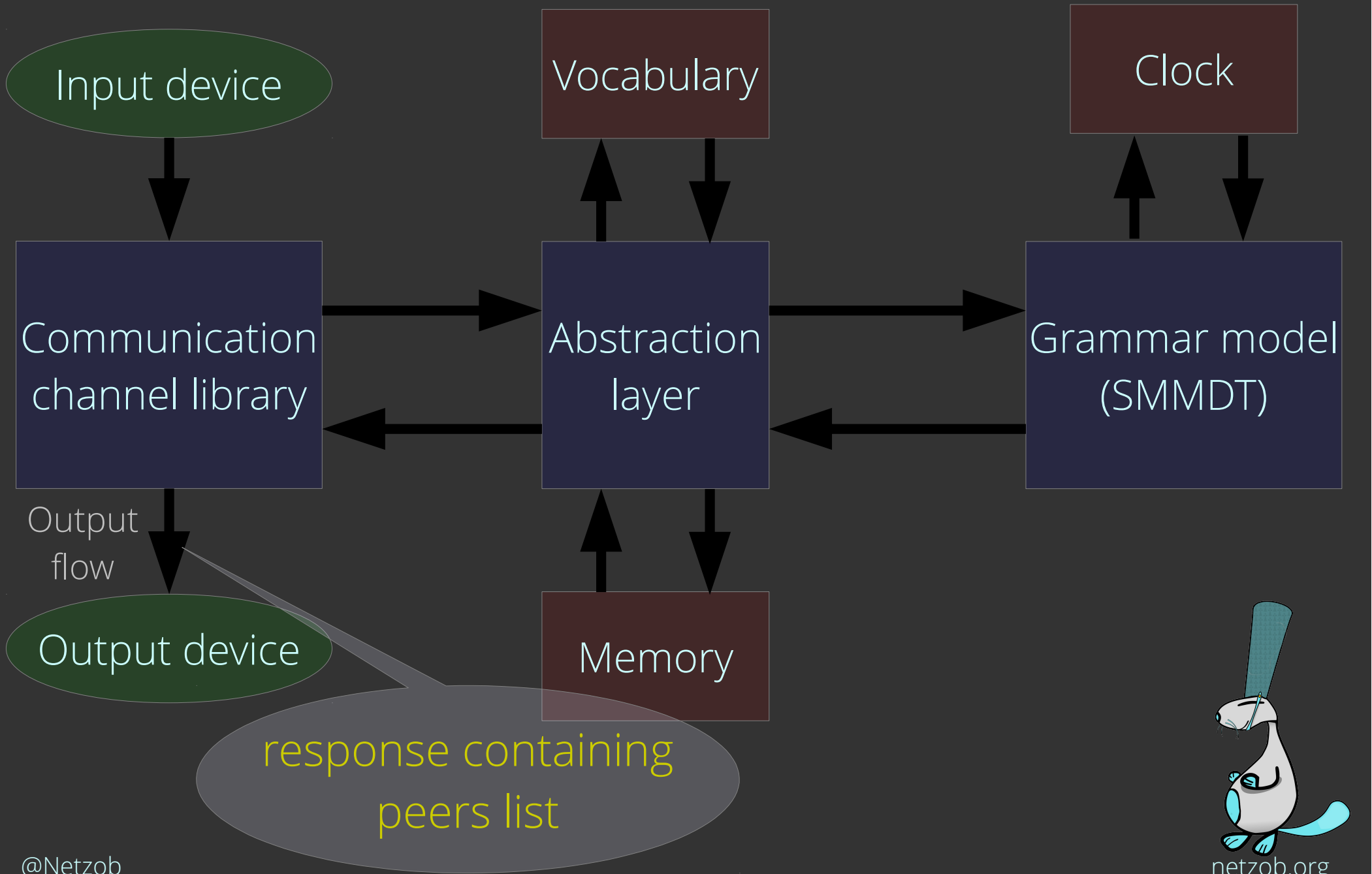
Abstraction and contextualization principles



Abstraction and contextualization principles



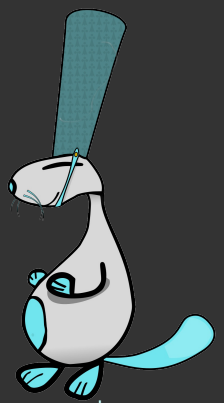
Abstraction and contextualization principles



Use cases of trafic simulation

Use case 1: simulation of a P2P botnet

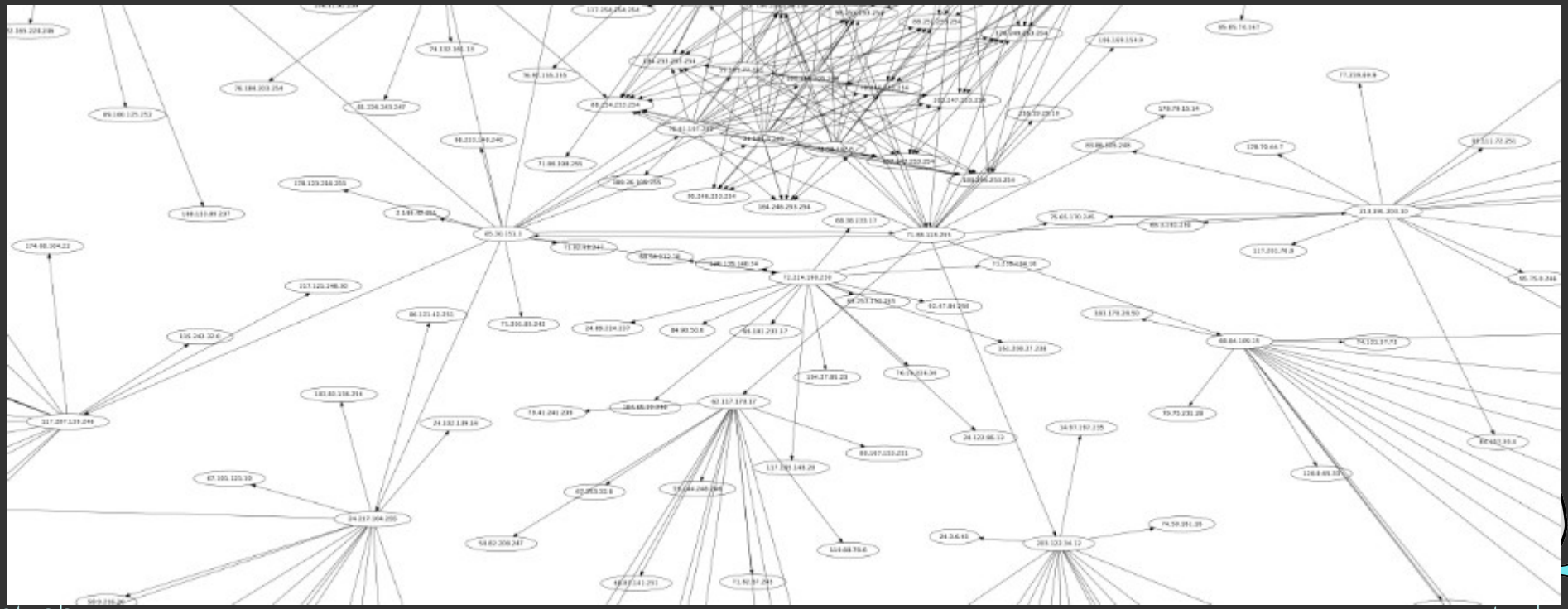
- Analyzing botnet scaling
- Studying botnet behavior at the network level
- Testing 3rd party products (IDS, ...)



Use cases of trafic simulation

Use case 2: retrieve the P2P zombie directory

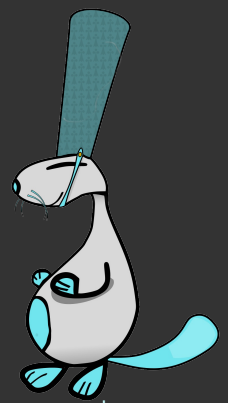
- Zero Access P2P map visualization
- Mapping the peers neighbours relations




Future improvments
of Netzob...

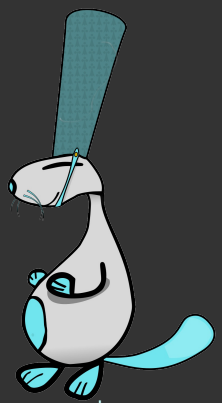
Integrated smart fuzzing, by leveraging the simulator engine

→ Allows to fuzz undocumented and propriatery protocols



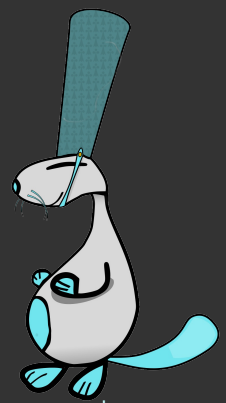
Integrated smart fuzzing, by leveraging the simulator engine

➞ Allows to fuzz undocumented and propriatery protocols



Support of more communication channels

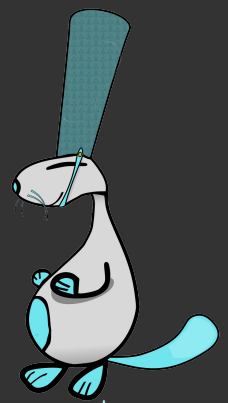
- USB
- IOCTL
- API (ssl_read, ssl_write, etc.)



Export protocol model in more 3rd party products

- Wireshark
- Scapy
- Peach Fuzzer

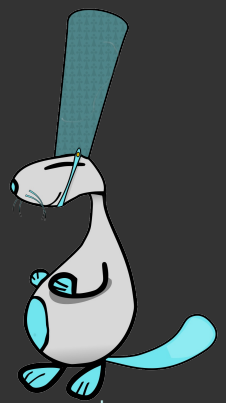
Allows protocol dissection with established tools



Export protocol model in more 3rd party products

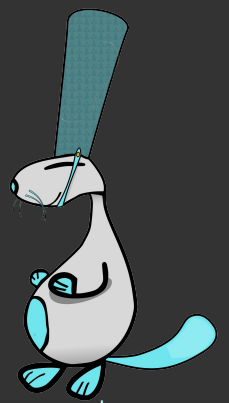
- Wireshark
- Scapy
- Peach Fuzzer

Allows fuzzing of unknown protocols with well-known tools

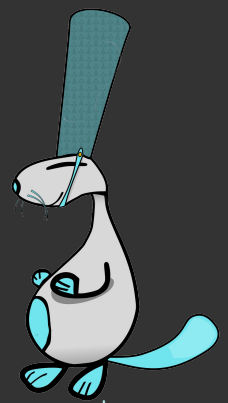



Conclusion...

- 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

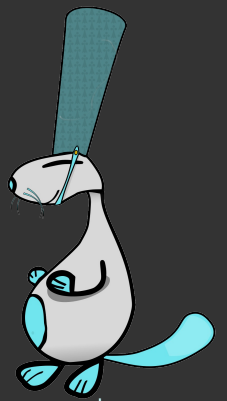


- Main current **sponsors**: AMOSSYS & Supélec
- Open to all kind of contributions
  - Feedback
  - Bug fix
  - Feature proposal / implementation
  - Translation
  - ...



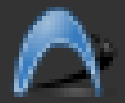
Netzob 0.4
« JumpingRhino »

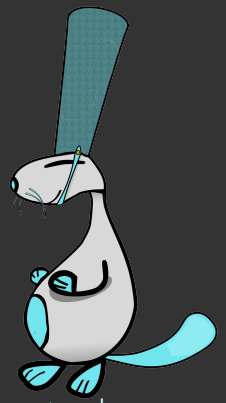
« It's alive ! It's alive ! »
Henry Frankenstein



- Towards a much more stable tool
- « Feature Release »
  - New User-Friendly Graphical Interface (GTK-3)
  - Upgrade the Vocabulary Inference process
    - Per-Layer RE Strategy
    - Add Transformation Functions ...
  - Extend Netzob with your own plugins
  - Upgraded Importers
    - Per-Layer Network importer
    - Ospy Importer ...

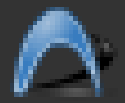




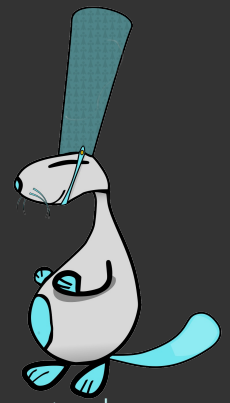


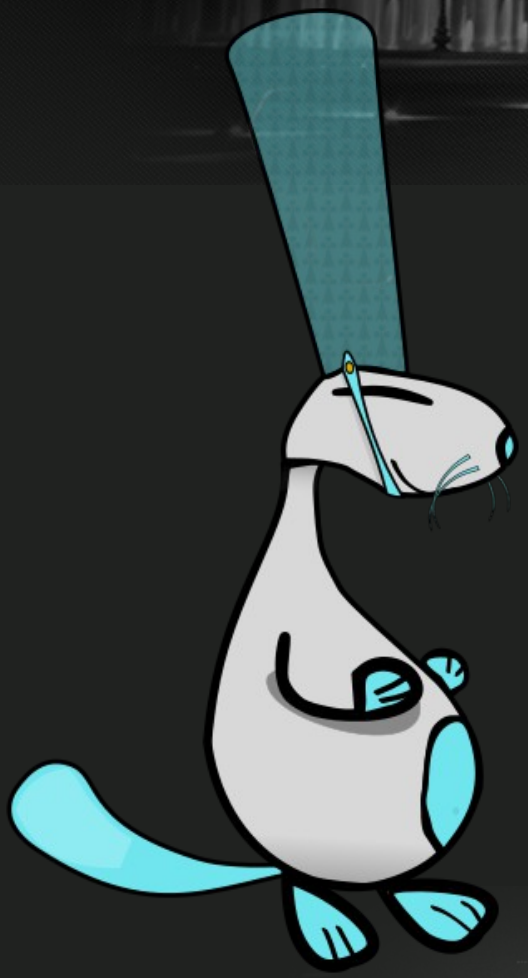
- Towards a much more stable tool
- « Feature Release »
  - New User-Friendly Graphical Interface (GTK-3)
  - Upgrade the Vocabulary Inference process
    - Per-Layer RE Strategy
    - Add Transformation Functions ...
  - Extend Netzob with your own plugins
  - Upgraded Importers
    - Per-Layer Network importer
    - Ospy Importer ...






Please download it and tell us what you think of it !





Thanks for you attention !

Any questions ?

Please complete the Speaker Feedback Surveys.

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