



# Netzob : un outil pour la rétro-conception de protocoles de communication

Georges Bossert <sup>1 2</sup>, Frédéric Guihéry <sup>1</sup>, Guillaume Hiet <sup>2</sup>

<sup>1</sup> AMOSSYS - Rennes, France

<sup>2</sup> Research team CIDre, Supélec

6 juin 2012



## Georges Bossert



Doctorant  
AMOSSYS / Supélec

## Frédéric Guihéry



Ingénieur Sécurité  
AMOSSYS

## Guillaume Hiet



Enseignant chercheur  
Supélec - CIDre

- **AMOSSYS** : Conseil et Expertise en Sécurité des Technologies de l'Information.
- **SUPELEC - CIDre** : Groupe de recherche focalisé sur la sécurité des systèmes d'informations distribués.



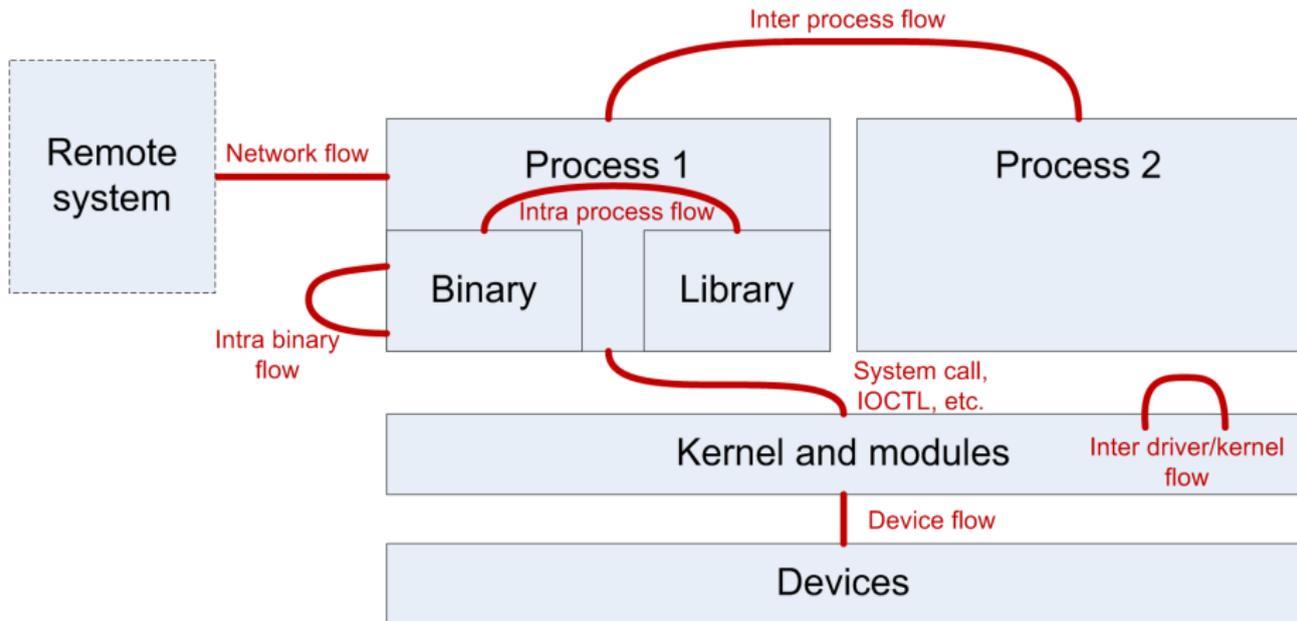
# Plan

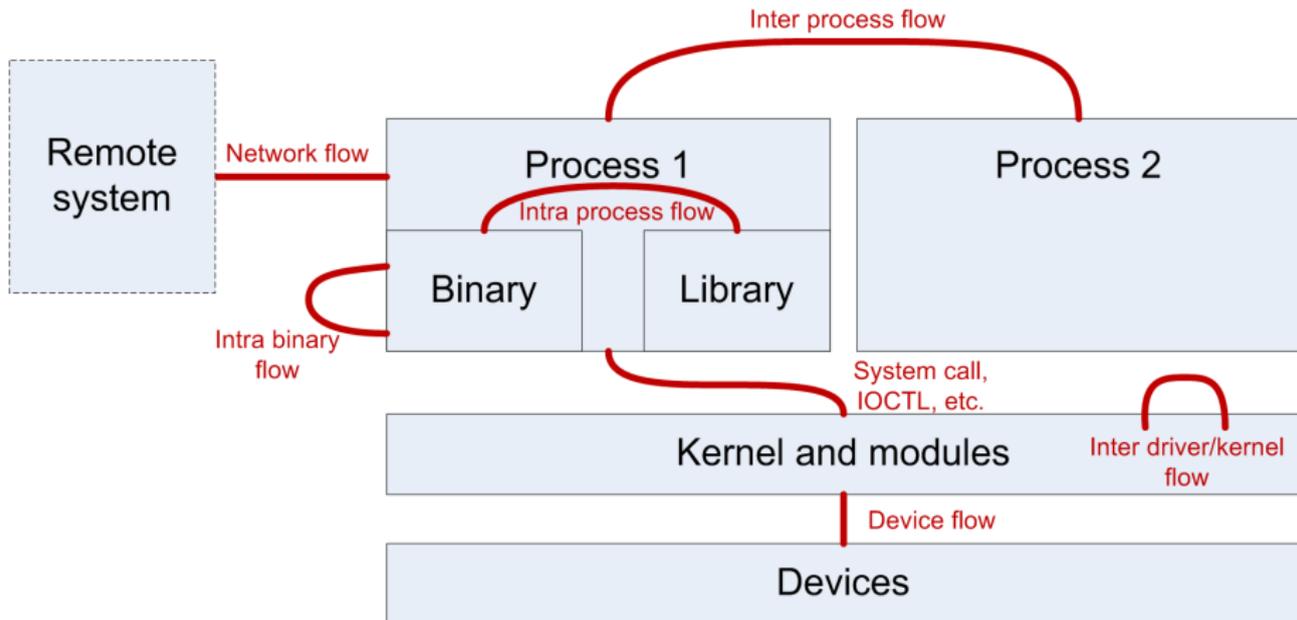
- 1 Le contexte
- 2 Notre modèle d'un protocole
- 3 L'inférence du modèle
- 4 Présentation de l'outil Netzob



# Le contexte

Omniprésence des protocoles de communication





Malheureusement, les spécifications ne sont pas toujours disponibles  
(protocoles propriétaires ou non documentés)



Pour **évaluer** la robustesse des implémentations

=> Fuzzer l'API de contrôle d'une centrifugeuse.

Pour **générer** un trafic réaliste et contrôlable.

=> Simuler la présence d'un botnet.

Mais également...

- Pour **analyser** le trafic et identifier d'éventuelles fuites de données.
- Pour **développer** une version libre d'une implémentation propriétaire.
- Pour **valider** l'implémentation d'un protocole vis-à-vis d'une spécification.



L'essentiel du temps de rétro-analyse d'un protocole ressemble à ça :



### L'essentiel du temps de rétro-analyse d'un protocole ressemble à ça :



- Processus complexe et laborieux
- Processus essentiellement manuel, voire visuel
- Communauté dépourvue d'outils d'analyse



### L'essentiel du temps de rétro-analyse d'un protocole ressemble à ça :



- Processus complexe et laborieux
- Processus essentiellement manuel, voire visuel
- Communauté dépourvue d'outils d'analyse

Ce constat, partagé, a mené à la création du projet Netzob



## Protocole informatique : spécification

- Vocabulaire : liste des messages et leur format.
- Grammaire : règles de procédure permettant d'assurer la cohérence des messages échangés.



## Protocole informatique : spécification

- Vocabulaire : liste des messages et leur format.
- Grammaire : règles de procédure permettant d'assurer la cohérence des messages échangés.

## Rétro-conception

- Retrouver la spécification de protocoles lorsqu'elle est inconnue
  - Par **inférence** (passive ou active) ou par analyse (manuelle)
  - A partir des **messages échangés** ou de l'exécutable

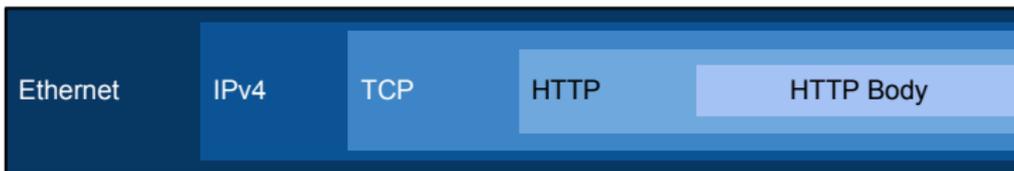
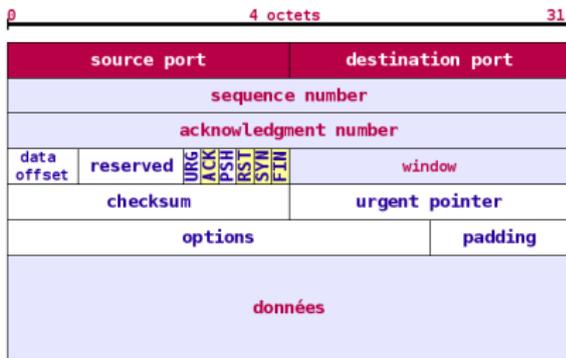
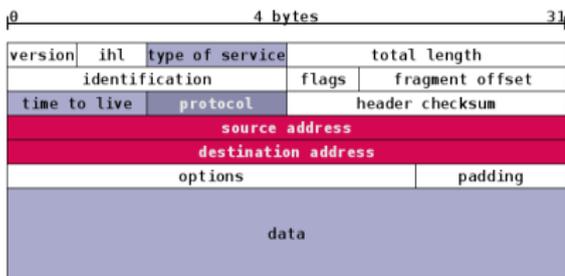


# Plan

- 1 Le contexte
- 2 Notre modèle d'un protocole**
- 3 L'inférence du modèle
- 4 Présentation de l'outil Netzob



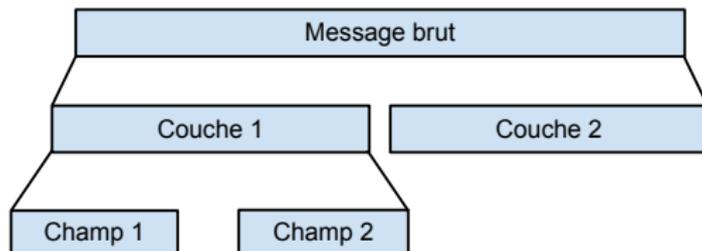
### Exemples que l'on souhaite vouloir modéliser

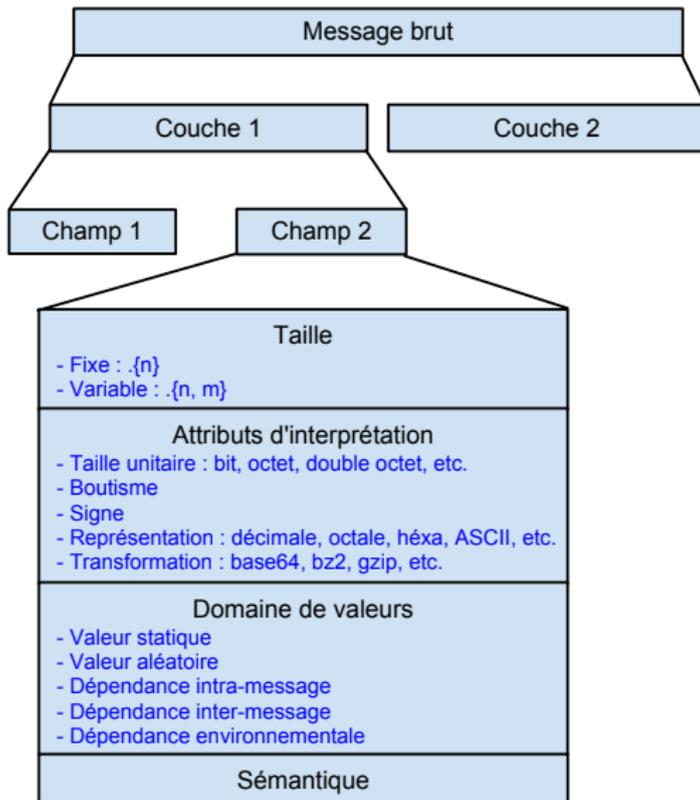




# Le modèle du vocabulaire

## Réponse aux besoins







**#1** : Traduire la relation entre un symbole en entrée et un symbole en sortie

- Répondre « *attack successful* » lorsque l'on reçoit « *attack* ».



**#1** : Traduire la relation entre un symbole en entrée et un symbole en sortie

- Répondre « *attack successful* » lorsque l'on reçoit « *attack* ».

**#2** : Définir plusieurs symboles de sortie pour le même symbole d'entrée

- Répondre « *attack successful* » ou « *attack failed* ».



**#1** : Traduire la relation entre un symbole en entrée et un symbole en sortie

- Répondre « *attack successful* » lorsque l'on reçoit « *attack* ».

**#2** : Définir plusieurs symboles de sortie pour le même symbole d'entrée

- Répondre « *attack successful* » ou « *attack failed* ».

**#3** : Associer une probabilité d'émission pour chaque symbole de sortie

- « *attack successful* » (90%) ou « *attack failed* » (10%).



**#1** : Traduire la relation entre un symbole en entrée et un symbole en sortie

- Répondre « *attack successful* » lorsque l'on reçoit « *attack* ».

**#2** : Définir plusieurs symboles de sortie pour le même symbole d'entrée

- Répondre « *attack successful* » ou « *attack failed* ».

**#3** : Associer une probabilité d'émission pour chaque symbole de sortie

- « *attack successful* » (90%) ou « *attack failed* » (10%).

**#4** : Prendre en compte le temps de réponse

- « *attack successful* » plus rapide que « *attack failed* ».



### Machine de Mealy Stochastique à Transitions Déterministes

$$MMSTD = \langle S, X, Y, T, q_0 \rangle$$

$q_0$  État initial

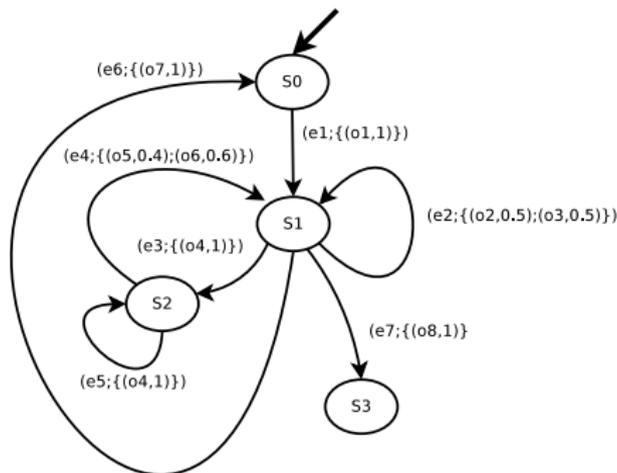
$S$  Ensemble des états

$X$  Alphabet des symboles d'entrées

$Y$  Alphabet des symboles de sorties

$T$  Ensemble de matrices de transitions

- $|T| = |X| \times |Y|$
- $T = \{A(x, y)\}$ ,  $a_{i,j}(x, y) = (p(s_j, y | s_i, x), t_{i,j}(x, y))$
- $\forall x \in X, \forall s_i, s_j \in S \quad p(s_j | s_i, x) = \sum_{y \in Y} p(s_j, y | s_i, x) \in \{0, 1\}$



### Alphabets :

#### Entrées

e1=".login 123"  
e2=".info"  
e3=".ddos"  
e4="\$IP"  
e5="!\$IP"  
e6=".logout"  
e7=".disconnect"

#### Sorties

o1="Welcome master"  
o2="Windows XP"  
o3="Linux"  
o4="IP of target ?"  
o5="Attack successfull"  
o6="Attack failed"  
o7="Goodbye"  
o8="Disconnected"

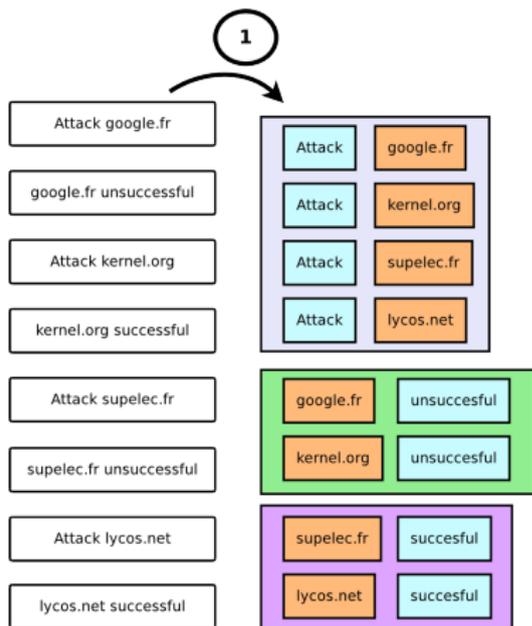
### Pour résumer :

- Automate symbolique
- **Transitions déterministes** mais symboles de sorties indéterministes
- Prise en compte du **temps de réaction**



# Plan

- 1 Le contexte
- 2 Notre modèle d'un protocole
- 3 L'inférence du modèle**
- 4 Présentation de l'outil Netzob



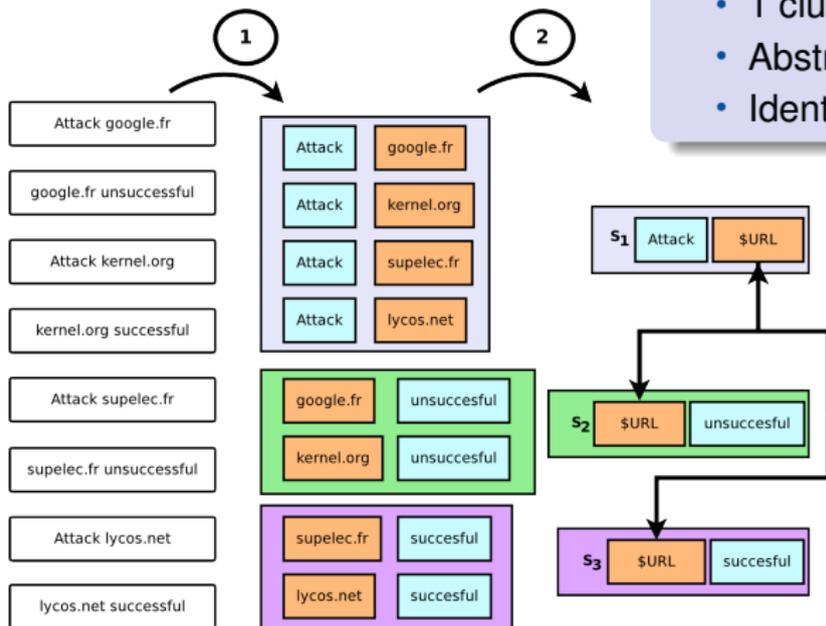
## #1 : Découpage et regroupement

- Découpage en champs
- Regroupement des messages par similarité
- Approche semi-automatique



## #2 : Abstraction en symboles

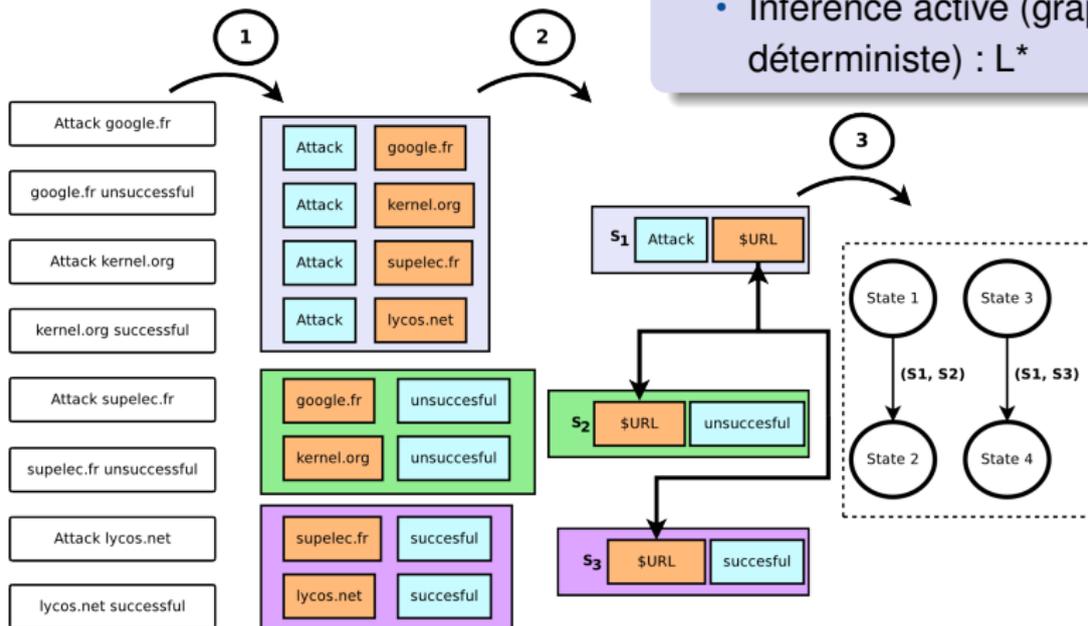
- 1 cluster = 1 symbole
- Abstraction des champs
- Identification des dépendances





## #3 : Inférence du graphe de transition

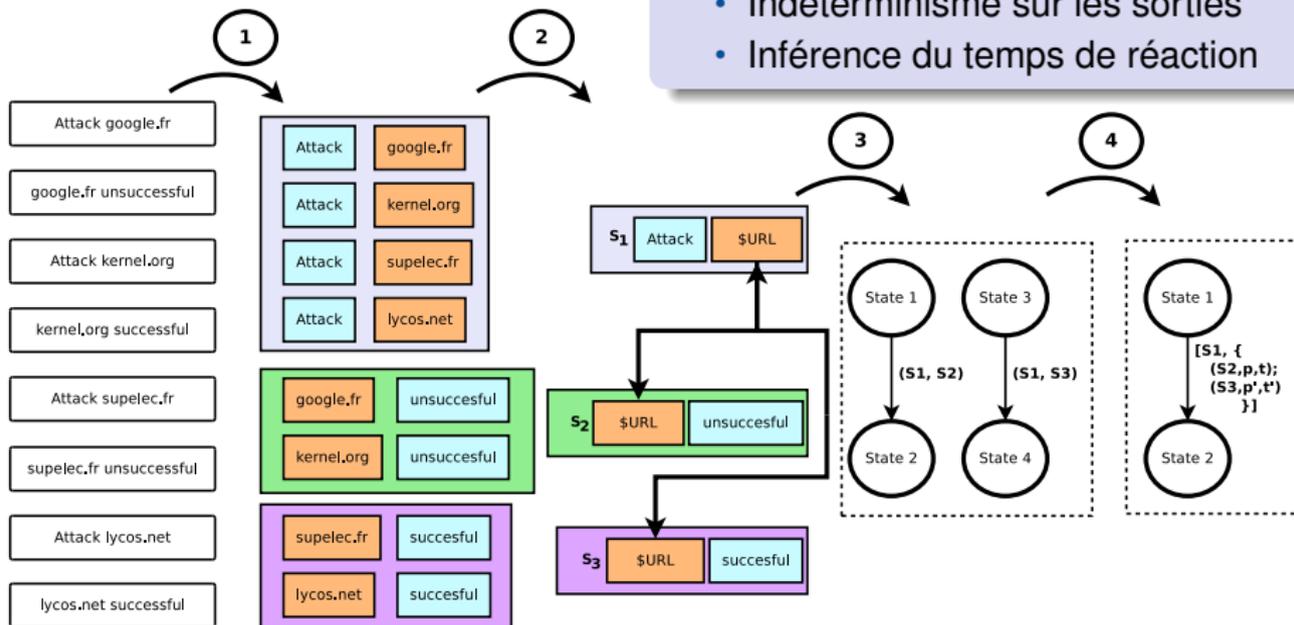
- Inférence active (graphe déterministe) :  $L^*$





## #4 : Généralisation de l'automate

- Indéterminisme sur les sorties
- Inférence du temps de réaction





### Le découpage en champs

- Partitionnement par délimiteur
- Partitionnement par variation

0	2	010600	00003d1e	0000000000000000	c0a8000a00000000	0000000000	0b8201fc42	00000000
0	2	010600	00003d1d	0000000000000000	c0a8000ac0a80001	0000000000	0b8201fc42	00000000
0	1	010600	00003d1e	0000000000000000	0000000000000000	0000000000	0b8201fc42	00000000
0	1	010600	fe089c15	0000000000000000	0000000000000000	0000000000	50ba1247cb	00000000
0	1	010600	9a5a2277	0000000000000000	0000000000000000	0000000000	1cc07e38c3	00000000
0	2	010600	fe089c15	0000000000000000	0a1414140a141404	0000000000	50ba1247cb	00000000
0	2	010600	fe089c15	0000000000000000	0a1414140a141404	0000000000	50ba1247cb	00000000
0	2	010600	9a5a2277	0000000000000000	c0a8000e00000000	0000000000	1cc07e38c3	00000000
0	2	010600	33dca406	0000000000000000	c0a8000e00000000	0000000000	1cc07e38c3	00000000

Champs dynamiques à taille fixe

- Alignement de séquence (Needleman-Wunsch)

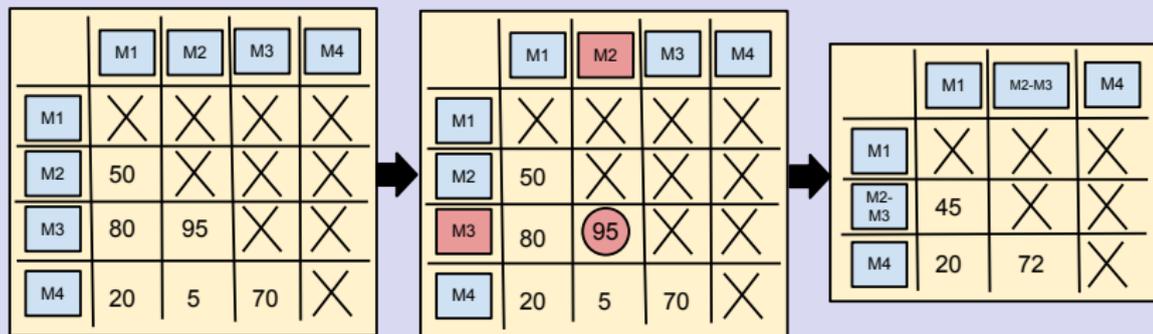
150 Opening BINARY mode data connection for	tb.php	(	5292	bytes)..
150 Opening BINARY mode data connection for	rss.php	(	3965	bytes)..
150 Opening BINARY mode data connection for	coords.txt	(	365	bytes)..
150 Opening BINARY mode data connection for	access_marsToMai.log	(	1392891	bytes)..

Champs dynamiques de taille variable



### Regroupement par clustering UPGMA

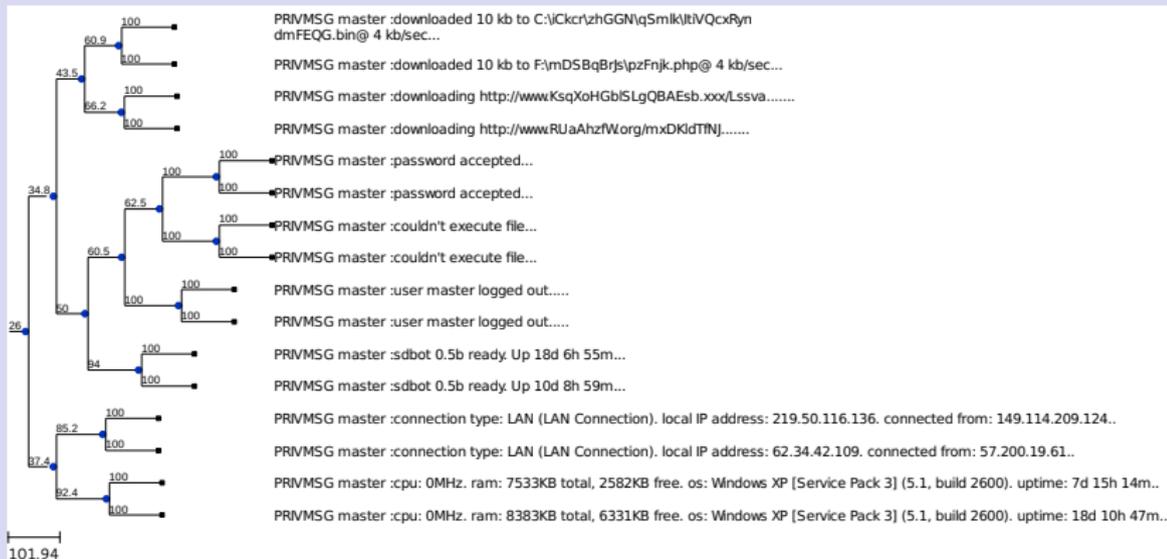
- Calcul de similarité en sortie de Needleman-Wunsh
- Remplissage d'une matrice de similarité
- Fusion des messages les plus similaires
- Puis on itère...





### Regroupement par clustering UPGMA

- Représentable sous la forme d'un dendrogramme
- Possibilité de choisir un seuil de regroupement





## Inférence du graphe de transition

- **Principe** : soumettre des séquences de **symboles** à un « professeur »
- Transmission des messages vers/depuis le binaire confiné
- Variante de l'algorithme  $L^*$  (Angluin).
- Ré-initialisation entre chaque soumission (virtualisation)

## Généralisation du modèle

- Ajout de l'indétermination en sortie :
  - rejeu des traces dans l'automate inféré
  - estimation des probabilités des symboles de sorties
- Ajout du temps de réaction
  - Mesure du temps entre un symbole d'entrée et de sortie.
  - Utilisation d'une loi normale.



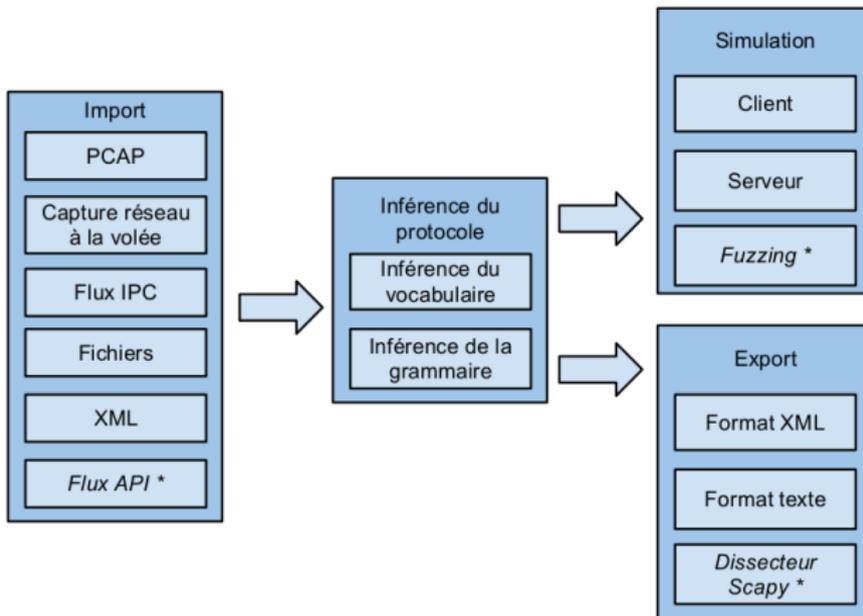
# Plan

- 1 Le contexte
- 2 Notre modèle d'un protocole
- 3 L'inférence du modèle
- 4 **Présentation de l'outil Netzob**



# Présentation de l'outil Netzob

## L'architecture fonctionnelle





## [ Démo ]

Netzob - Inferring communication protocols

Workspace Project View Help

Vocabulary inference Grammar inference Simulator

1 - Message format inference: Sequence alignment, Force partitioning, Simple partitioning, Smooth partitioning, Reset partitioning  
 2 - Field type inference: Freeze partitioning, Messages distribution, Data carving  
 3 - Dependencies inference: Find size fields, Environment dependencies  
 4 - Visualization: Format (hex), Unit size (none), Sign (unsigned), Endianess (big-endian)

- Search data: string

Search

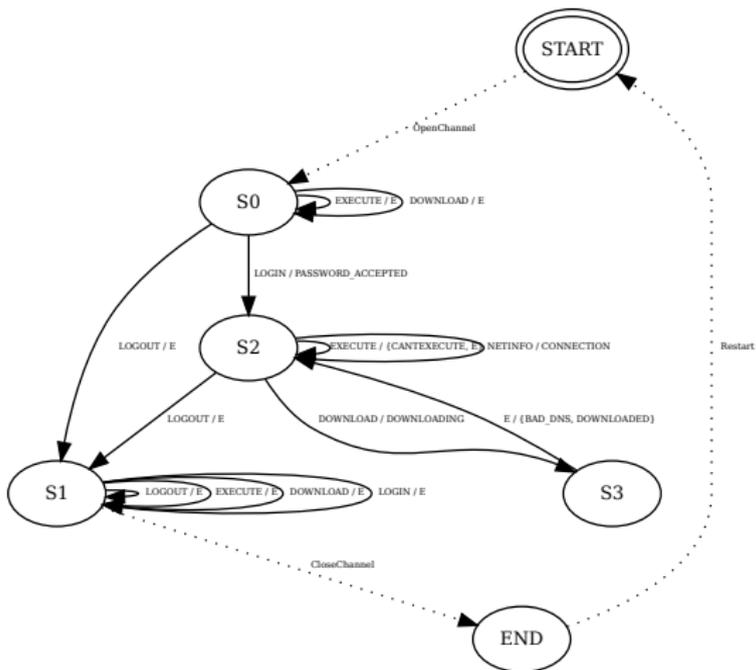
dhcp	Field 9	Field 10	Field 11	Field 12	Field 13
Symbol 7 (7)	863501 (...16)	04e0a800	(. (. 32))	zz000	(. (. 42))
Symbol 14 (10)	hex	hex	string	hex	hex
PCAP IMPORT (6)	863501 0132	4 c0 a8 0	0132cccccccc	ff000	cccccccccccc
PCAP IMPORT (4)	863501 0132	4 c0 a8 0	0132cccccccc	ff000	cccccccccccc
	863501 0132	4 c0 a8 0	0132cccccccc	ff000	cccccccccccc
	863501 0132	4 c0 a8 0	0132cccccccc	ff000	cccccccccccc
	863501 0132	4 c0 a8 0	0132cccccccc	ff000	cccccccccccc
	863501 0132	4 c0 a8 0	0132cccccccc	ff000	cccccccccccc
	863501 0132	4 c0 a8 0	0132cccccccc	ff000	cccccccccccc
	863501 0132	4 c0 a8 0	0132cccccccc	ff000	cccccccccccc
	863501 0132	4 c0 a8 0	0132cccccccc	ff000	cccccccccccc
	863501 0132	4 c0 a8 0	0132cccccccc	ff000	cccccccccccc
	863501 0132	4 c0 a8 0	0132cccccccc	ff000	cccccccccccc
	863501 0132	4 c0 a8 0	0132cccccccc	ff000	cccccccccccc

Properties

Timestamp 1334269827  
 Protocol UDP  
 IP source 192.168.0.14  
 IP target 192.168.0.254  
 Source port 68  
 Target port 67

Move to ...  
 Edit field  
 Field visualization  
 Concatenate field  
 Split field  
 Partitioning  
 Field's domain of definition  
 Configure variation of field  
 Copy to clipboard  
 Message properties  
 Delete message

Format  
 UnitSize  
 Sign  
 Endianess





- Développé en Python et en C
- Sous licence GPLv3
- Actuellement 7 contributeurs (AMOSSYS, Supélec, Bull)
- Disponibilité
  - Via dépôt git ou tar.gz
  - Paquet Debian (À la recherche d'un sponsor)
  - Paquet Gentoo et Windows (en cours)

Network protocol modelization by reverse engineering

# NETZOB

still in dev

LATEST AND GREATEST (VERSION 0.3.1)  
[changelog](#) | [past releases](#) |   

HOME ABOUT DOWNLOAD DOCUMENTATION CONTACT

## WHAT IS NETZOB ?

NETZOB SIMPLIFIES THE WORK FOR SECURITY AUDITORS by providing a complete framework for the REVERSE ENGINEERING OF COMMUNICATION PROTOCOLS. It handles different types of protocols : text protocols (like HTTP and IRC), fixed fields protocols (like IP and TCP) and variable fields protocols (like ASN.1 based formats).

Netzob is therefore suitable for REVERSING NETWORK PROTOCOLS, STRUCTURED FILES AND SYSTEM AND PROCESS FLOWS (IPC and communication with drivers). Netzob is provided with modules dedicated to capture data in multiple contexts (network, file, process and kernel data acquisition).

Besides being intrinsically adapted for classical reverse engineering of protocols, Netzob is capable of IDENTIFYING POTENTIAL VULNERABLE DATA FIELDS, like size fields and their associated payloads.

Furthermore, Netzob integrates a STOCHASTIC AND STATEFULL MODEL to represent any communication protocol through an XML declaration. This model declaration is then loaded in a dedicated component of Netzob, its network simulator. Therefore, it becomes easy to SIMULATE MULTIPLE ACTORS (servers and clients) which communicates according to the inferred protocol for advanced fuzzing processes or active inferring process.





## Bilan

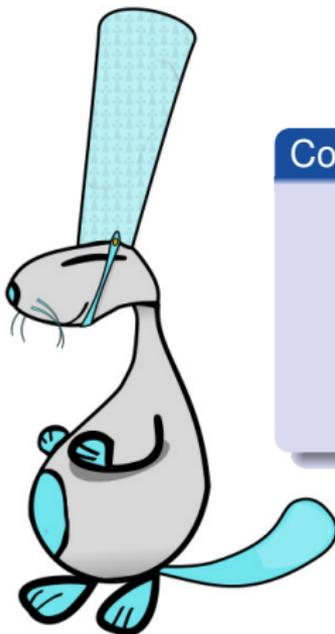
- Domaine assez actif au niveau académique ces dernières années
- Mais quasiment aucune retombée dans les outils publics
- Netzob vise à combler ce manque
  - Support de travaux académiques
  - Utilisable dans un **contexte opérationnel** (audit, évaluation, développement, ...)

## Travaux futurs

- Nouveaux capteurs (API, noyau, périphériques)
- Gestions des sessions
- Gestion d'autres formats d'encodages (ASN.1, TSN.1, EBML, etc.)
- Open « wishlist » : <https://dev.netzob.org>



- <http://www.netzob.org> , @Netzob
- Des questions à poser à Zoby ?



## Comment contribuer :

- Partage de PCAPs (malware, scada...)
- Participation au développement
- Retour d'expérience



$$M(i,j) = \text{MAX} [ M(i-1,j-1) + S(i,j) ; M(i,j-1) + W ; M(i-1,j) + W ]$$

$S(i,j)$  = SCORING SCHEME |  $W$  = GAP PENALTY

	G	A	A	T	T	C	A	G	T	T	A
G	0	0	0	0	0	0	0	0	0	0	0
G	0										
A	0										
T	0										
C	0										
G	0										
A	0										

	G	A	A	T	T	C	A	G	T	T	A
G	0	0	0	0	0	0	0	0	0	0	0
G	0	1									
A	0										
T	0										
C	0										
G	0										
A	0										

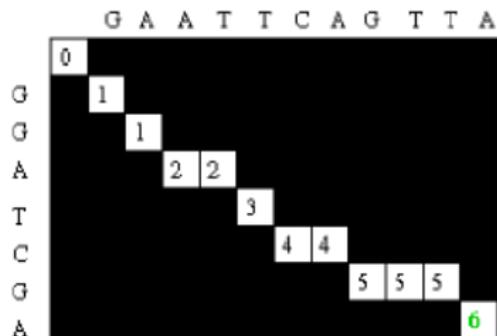
	G	A	A	T	T	C	A	G	T	T	A
G	0	1	1	1	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	1	1	1	1
A	0	1	2	2	2	2	2	2	2	2	3
T	0	1	2	3	3	3	3	3	3	3	3
C	0	1	2	3	3	3	4	4	4	4	4
G	0	1	2	3	3	3	4	4	5	5	5
A	0	1	2	3	3	3	4	5	5	5	6

	G	A	A	T	T	C	A	G	T	T	A
G	0	1	1	1	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	2	2	2	2
A	0	1	2	2	2	2	2	2	2	2	3
T	0	1	2	2	3	3	3	3	3	3	3
C	0	1	2	2	3	3	3	4	4	4	4
G	0	1	2	2	3	3	3	4	4	5	5
A	0	1	2	3	3	3	3	4	5	5	6



### Traceback step

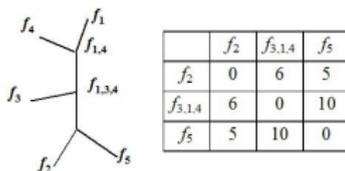
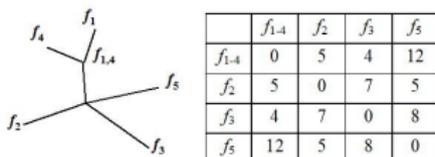
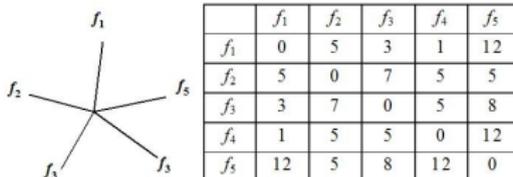
	G	A	A	T	T	C	A	G	T	T	A
0	0	0	0	0	0	0	0	0	0	0	0
G	0	1	1	1	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	2	2	2	2
A	0	1	2	2	2	2	2	2	2	2	3
T	0	1	2	2	3	3	3	3	3	3	3
C	0	1	2	2	3	3	3	4	4	4	4
G	0	1	2	2	3	3	3	4	4	5	5
A	0	1	2	3	3	3	3	4	5	5	5



G A A T T C A G T T A  
 | | | | | | | |  
 G G A \_ T C \_ G \_ \_ A



- Exécution itérative des opérations suivantes :
  - 1 Recherche dans la matrice le score le plus élevé.
  - 2 Regroupe les deux groupes associés.
  - 3 Mise à jour de la matrice (calcul de la moyenne des deux lignes).





- Tant que l'automate hypothèse n'est pas conforme
  - Tant que la table n'est pas close et consistente
    - Soumission d'une requête d'appartenance
  - Construction d'un automate hypothèse
  - Recherche d'un contre exemple (requête d'équivalence)

