



Inferring communication protocols with your NETZOB

(NETwork protocol modeliZatiOn By reverse engineering)

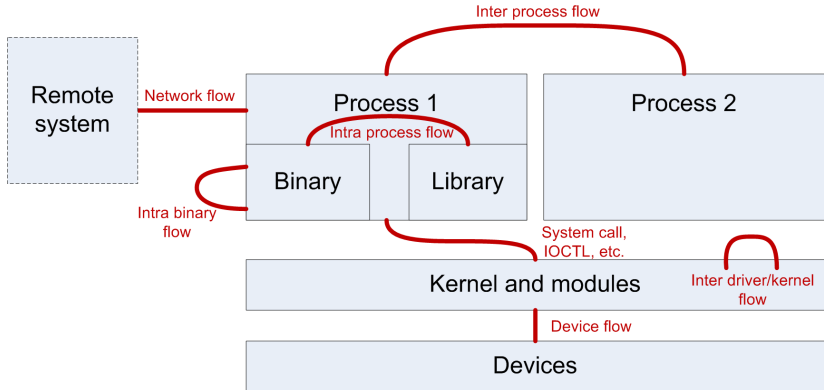
Georges Bossert ¹ ², Frédéric Guihéry ¹

¹ AMOSSYS - Rennes, France

² Research team CIDre, Supélec

10 nov 2011

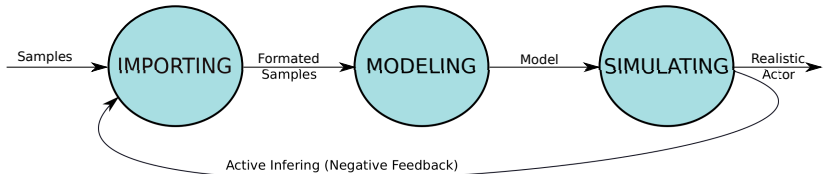




Why would you reverse engineer communication protocols ?

- ▶ To **assess the robustness** of a protocol handler implementation
- ▶ To **simulate** realistic traffic to test third-party products
- ▶ To **analyze** the traffic for potential information leakage
- ▶ To **develop** an open source version of a proprietary implementation
- ▶ ...

- ▶ Framework for the reverse engineering of communication protocols



1. **Import/capture** of samples of the protocol
2. Identify and extract a **model** which describes the protocol
3. **Simulate/tamper** traffic according to the model and its actors

Multiple accepted format

- ▶ PCAPs (network captured data)
- ▶ Files (binary and text)
- ▶ Raw (hexadecimal in a dedicated XML representation)

Multiple Importers

- ▶ **Live network** captor
- ▶ **Inter-process** communication captor (ex : unix socket, pipes, shared memory)
- ▶ **Intra-process** communication captor (ex : calls to OpenSSL library)

Protocol inference relies on

- ▶ Structure inference (based on sequence alignment)
- ▶ Regroupment of similar messages
- ▶ Field type inference
- ▶ Fields dependencies identification
- ▶ Field semantic inference
- ▶ Grammar inference (automata of the protocol)

Communication protocol

- ▶ A communication protocol can be defined through its :
 - ▶ vocabulary (set of valid words)
 - ▶ grammar (set of valid sentences)

Passive inferring process of the vocabulary

- ▶ “read-only” process (do not require a participation in the communication).
- ▶ **identify the fixed and dynamic fields** of all the messages.
- ▶ **regroups equivalent messages** depending of their fields structures.

Needleman and Wunsch algorithm

- ▶ Originally a bio-informatic algorithm (sequencing DNA)
- ▶ Align two messages and identify common patterns and field structure
- ▶ Computes an **alignment score** representing the efficiency of the alignment

01	1d6b815d385b4c6e13f72eb6c35f996c	00000000	c577b9d66edf5b73d29b5ae761e84463
01		00000000	326d078d656a9637a8ea7910d671374264

But

- ▶ Needleman and Wunsch only applies on 2 messages...

Clustering algorithm

- ▶ Identify equivalent messages based on their **alignment score**.
- ▶ Build a hierarchical organization of the messages with the **UPGMA algorithm** (Unweighted Pair Group Method with Arithmetic Mean)

► Sequence alignment for a set of similar messages

Name	Name	Name	Name	Name	Name	Name	Name
0500	{,2}	0310000000	{,2}	000000	{,78}	0000	{,743}
ascii, binary	ascii, binary	ascii, binary	binary	ascii, binary	binary	ascii, binary	binary
0500	00	0310000000	f4	000000	03000000dc000000000020b4135b9e9e5a9035a9a67ce348a73e4c80000004c148c0300000	0000	20000000300000040000000000000005
0500	02	0310000000	f4	000000	03000000dc0000000000060031323323200020000000000000000004c148c0300000	0000	2000000c000000400000000000000005
0500	00	0310000000	f4	000000	08000000dc000000000020b4135b9e9e5a9035a9a67ce348a73e4c80000004c148c0300000	0000	20000000300000040000000000000005
0500	02	0310000000	f4	000000	08000000dc00000000000600313233232000200000000000000000004c148c0300000	0000	2000000c000000400000000000000005
0500	00	0310000000	f4	000000	07000000dc000000000020b4135b9e9e5a9035a9a67ce348a73e4c80000004c148c0300000	0000	2000000030000004000000218f5a7505
0500	00	0310000000	f4	000000	06000000dc000000000020b4135b9e9e5a9035a9a67ce348a73e4c80000004c148c0300000	0000	2000000030000004000000218f5a7505
0500	00	0310000000	f4	000000	04000000dc000000000020b4135b9e9e5a9035a9a67ce348a73e4c80000004c148c0300000	0000	2000000030000004000000218f5a7505
0500	00	0310000000	f4	000000	05000000dc000000000020b4135b9e9e5a9035a9a67ce348a73e4c80000004c148c0300000	0000	2000000030000004000000218f5a7505
0500	02	0310000000	f4	000000	07000000dc00000000000600313233232000200000000000000000004c148c0300000	0000	2000000c0000004000000218f5a7505

► **The abstract structure is represented as a regex**

```
0007cb3e3d6b001cc07e38c30800
450000
(.{,2})
(.{,6})
00401100
(.{,2})
c0a8000ed41b28f0
(.{,4})
003500
(.{,2})
(.{,10})
00000100000000000000
(.{,46})
0000
(.{,2})
0001
```

Tune and adapt the inferring process with dedicated tools :

- ▶ Manual sequencing
- ▶ Fields type identification :
 - ▶ Primary types (binary, ascii, num, base64, ...)
 - ▶ Computes the definition domain of a field (unique elements)
- ▶ Semantic data identification
 - ▶ Emails, IP, ...
 - ▶ Data carving (images, tar gz, ...)
- ▶ Fields dependencies identification :
 - ▶ Length fields and associated payloads
 - ▶ Encapsulated messages identifications
- ▶ Fields statistical distribution

Select trace : reputation1 Current trace : reputation1 Save trace

Import **Modelization** Fuzzing Expert Simulator

Options during alignment process

Similarity threshold : 90 Orphan reduction

Start alignment Slick regexes

Analyses after alignment process

Visualization options

Protocol type : Fixed field

Groups	Col0	Col1	Col2	Col3
RESPONSES [22]	HEADER	ID_USER	IP_REQUESTED	REQUEST
REQUESTS [22]	01	1d6b815d385b4c6e13f72eb6c35f996c	((. {2}00. {4}))	((. {32}))
	01	1d6b815d385b4c6e13f72eb6c35f996c	00000000	c577b9d66edf5b79d29b5ae761e84463
	01	1d6b815d385b4c6e13f72eb6c35f996c	00000000	c577b9d66edf5b79d29b5ae761e84463
	01	1d6b815d385b4c6e13f72eb6c35f996c	00000000	c577b9d66edf5b79d29b5ae761e84463
	01	1d6b815d385b4c6e13f72eb6c35f996c	00000000	c577b9d66edf5b79d29b5ae761e84463
	01	1d6b815d385b4c6e13f72eb6c35f996c	00000000	c577b9d66edf5b79d29b5ae761e84463
	01	1d6b815d385b4c6e13f72eb6c35f996c	00000000	c577b9d66edf5b79d29b5ae761e84463
	01	1d6b815d385b4c6e13f72eb6c35f996c	00000000	c577b9d66edf5b79d29b5ae761e84463
	01	1d6b815d385b4c6e13f72eb6c35f996c	c7005e0a	73666ddcf1aaacd64014453c52d275
	01	1d6b815d385b4c6e13f72eb6c35f996c	1e005e0a	f4d4db0d1b213bf3e405ebcf0dff
	01	1d6b815d385b4c6e13f72eb6c35f996c	03005e0a	45395bcef6d7e25cacecc348e7bf113

Properties of message 90da8317-9106-420c-8bd3-568a4a8ef508

Property	Value
ID	90da8317-9106-420c-8bd3-568a4a8ef508
Type	Network
Timestamp	1319213147
Protocol	UDP
IP source	10.94.0.10
IP target	79.125.11.244
Source port	55952
Target port	8007
Data	011d6b815d385b4c6e13f72eb6c35f996c0000000c577b9d66edf5b79d29b5ae761e84463

Domain of definition for the column 3

```

1
2
3
4 45395bcef6d7e25cacecc348e7bf113
5 73666ddcf1aaacd64014453c52d275
6 984f36e031c4808b59f5c3a72bafadb
7 bbc21edaeadae4d3d379739c62ab4fc0
8 c577b9d66edf5b79d29b5ae761e84463
9 e98cd0c070455b69313493b7270d5d
10 f4d4db0d1b213bf3e405ebcf0dff

```

Name Value

HEADER: 01

ID_USER: 1d6b815d385b4c6e13f72eb6c35f996c

IP_REQUESTED: ((. {2}00. {4})) / c7005e0a

REQUEST: ((. {32})) / 73666ddcf1aaacd64014453c52d275

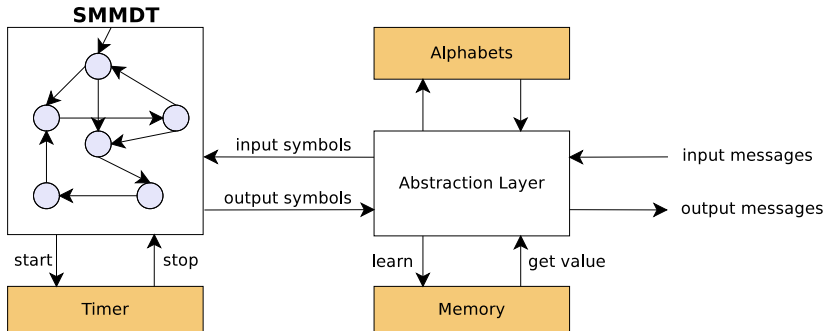
► Example of IP and UDP payloads extraction

```

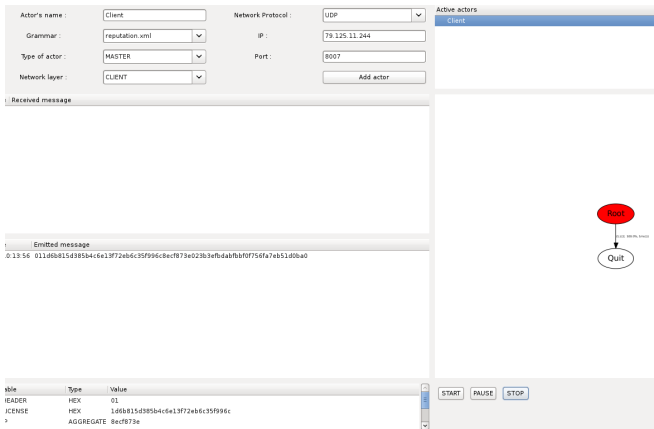
Name:      0007cb3e3d6b001cc07e38c30800
  Name:    450000                                Start of payload
  Name:    (.{2}) / 43                            Size field
  Name:    (.{6}) / 7c4140
  Name:    00401100
  Name:    (.{2}) / a7
  Name:    c0a8000ed41b28f0
    Name:  (.{4}) / d22d                            Start of payload
    Name:  003500
    Name:  (.{2}) / 2f                               Size field
    Name:  (.{10}) / be02f7aa01
    Name:  00000100000000000000
    Name:  (.{46}) / 0b6164736572766572616d7306616474656368026465
    Name:  0000
    Name:  (.{2}) / 1c
    Name:  0001
  
```

- ▶ **Simulate** any communication actors with NETZOB.
- ▶ Insert the inferred vocabulary in a **statefull** and **stochastic** dedicated model.
- ▶ Simulate clients **and** Masters.
- ▶ Supports Network (TCP/UDP) (Client/Server) communication channel.
- ▶ **Contextualized** input and output messages (included variables are set according to the environment)
- ▶ Highly extendable model

- **Simulate** any communication actors with NETZOB.



- **Simulate** any communication actors with NETZOB.



The screenshot displays the NETZOB simulation environment. On the left, the 'Actor configuration' panel includes fields for Actor's name (Client), Grammar (reputation.xml), Type of actor (MASTER), Network layer (CLIENT), Network Protocol (UDP), IP (79.125.11.244), and Port (8007). An 'Add actor' button is located below these fields. The 'Received message' section is currently empty. The 'Emitted message' section shows a log entry: '0:13:56 011d6b815d385b4c6e13f72eb6c35f996c8ecf873e023b3efbdabfbf0756fa7eb51d0ba0'. At the bottom left, a table displays the message details:

file	Type	Value
HEADER	HEX	01
LICENSE	HEX	1d6b815d385b4c6e13f72eb6c35f996c
0	AGGREGATE	8ecf873e

On the right, the 'Active actors' panel shows 'Client' as the active actor. Below it, a state transition diagram features a red oval labeled 'Root' with a downward arrow pointing to a white oval labeled 'Quit'. The diagram is titled '0:13:56:13:56:562'.

At the bottom right of the interface, there are three control buttons: 'START', 'PAUSE', and 'STOP'.

- ▶ License GPLv3 (AMOSSYS and SUPELEC as main supporters)
- ▶ Google Project : <http://code.google.com/p/netzob/>
- ▶ Official website : <http://www.netzob.org>



netzob
NETZOB - Network protocol modelization by reverse engineering

Project Home | Downloads | Wiki | Issues | Source | About Us

Summary | Updates | People

[Tip: Discuss and then document your ideas!](#)

Project Information

★ Starred by 17 users
Activity: [Low](#) | [Medium](#) | [High](#)

Code Source
[GITHUB](#)

Labels
Protocol, Networking, ReverseEngineering, Cipher, Security, Sequencing

Members
[test.guhey](#), [glossier](#)

Your role
[Owner](#)

Links

External links
[Amossys.fr](#)
[Supélec - AMOSSYS.fr](#)
[Amossys.fr](#)

Netzob simplifies the work for security auditors by providing a complete framework for the reverse engineering of all handles different types of protocols : text protocols (like HTTP and IRC), fixed fields protocols (like IP and TCP) and ASN.1 based formats. Netzob is therefore suitable for reversing network protocols, structured files and systems and communication with drivers. Netzob is provided with modules dedicated to capture data in multiple contexts (network, data acquisition).

Description of the functionalities

- Handle the following inputs as initial data :
 - PCAP
 - Network capturing (with Scapy)
 - Structured files with unknown format
 - Inter Process Communication (IPC) calls
 - Inter Processes Communication pipes, shared memory and sockets
 - Kernel Memory (with a dedicated module)
- Metabola representation of inputs
- Clustering (Program equivalent messages using an LPSIMD Algorithm to regroup similar messages
- an `ispdM2` and `ispdM1` implementations
- Sequencing, Alignment, Identification of fields in messages
- `libBerserk` & `libWorsch` implementations
- Fields dependencies identification
- `isearch` tools and associated outputs

Network-protocol modelization by reverse engineering

NETZOB

still in dev.

LATEST AND GREATEST (VERSION 0.3.1)
[CHANGES](#) | [EMAIL](#) | [GITHUB](#) | [ISSUES](#) | [FORUM](#) | [NEWS](#)

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 STATEFUL 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.

WHY WOULD YOU USE NETZOB ?



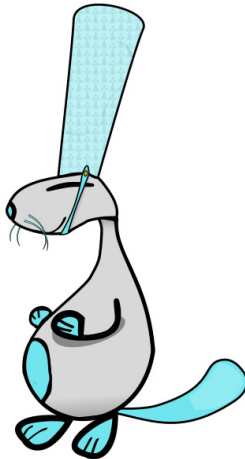
Current work in progress

- ▶ Debian package
- ▶ Windows Runtime DLL hijacker
- ▶ Windows package (MSI) and official Windows version
- ▶ Releasing an almost **first stable version** (version 0.3)

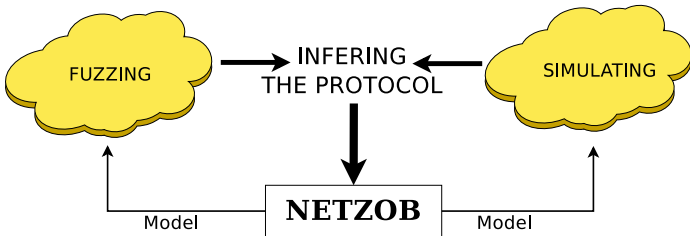
Futur work

- ▶ Active inferring
- ▶ Extending captors (kernel and intra-binary captor, etc.)
- ▶ Support the inference of more data encoding (ASN.1, TSN.1, EBML, etc.)
- ▶ Open “wishlist” (do not hesitate !)

- ▶ Want to ask questions to Zoby ?



- ▶ Originally to support **Fuzzing** and **Simulating** processes.



- ▶ **“FUZZERS”** need to reduce the processing time.
- ▶ **“SIMULATORS”** need to infer a realistic model.

SVN hosted source code

- ▶ Mostly Object Oriented Python.
- ▶ Some in C (even some in OpenMP for parrallelization).
- ▶ 15193 lines of code in 149 files (#80).

Source code quality

- ▶ **Continuous Integration** over Hudson
 - ▶ too few unit tests
 - ▶ “to be released” version : Netzob-0.3
- ▶ Dedicated Redmine **bug tracker**.
- ▶ Self-evaluated quality : **low/medium** (beta with bugs) few months before stable !

SVN hosted source code

- ▶ Mostly Object Oriented Python.
- ▶ Some in C (even some in OpenMP for parralelization).
- ▶ 15193 lines of code in 149 files (#80).

Source code quality

- ▶ **Continuous Integration** over Hudson
 - ▶ too few unit tests
 - ▶ “to be released” version : Netzob-0.3
- ▶ Dedicated Redmine **bug tracker**.
- ▶ Self-evaluated quality : **low/medium** (beta with bugs) few months before stable !

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

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

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

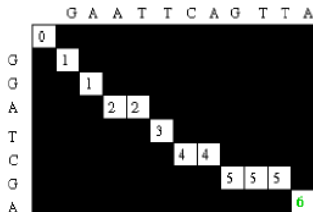
	G	A	A	T	T	C	A	G	T	T	A
0	0	0	0	0	0	0	0	0	0	0	0
G	0	1									
G	0										
A	0										
T	0										
C	0										
G	0										
A	0										

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

	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	6

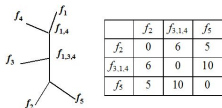
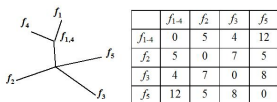
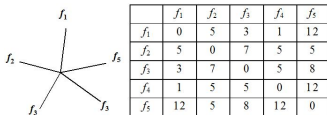
Traceback step

	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	6



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

- ▶ Iterative execution :
 1. Search in the matrix the highest score
 2. Regroup the two groups
 3. Update the matrix (compute the average of the two lines)



► Capture IPC flows with NETZOB

NETZOB: Network protocol modelization By reverse engineering

Select trace : Current trace : ... Save trace

Import Modelization Fuzzing Export Simulator

Network Capturing Update processes list 2559 /usr/lib/iceweasel/firefox-bin

IPC Capturing File system Network Interprocess Update flows

API capturing

PCAP import

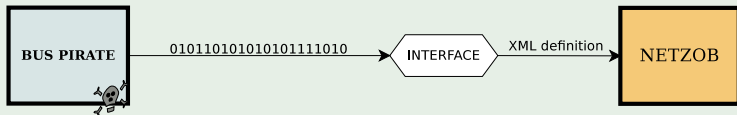
File import

FD	Type	Name
23u	FIFO	pipe
24u	FIFO	pipe
25u	FIFO	pipe
26u	FIFO	pipe
27u	CHR	/dev/urandom
28r	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/extensions.sqlite
29w	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/content-prefs.sqlite
30u	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/permissions.sqlite
31wr	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/addons.sqlite
32u	unix	socket
33ur	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/cookies.sqlite
34u	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/cookies.sqlite-w
35u	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/search.sqlite
36u	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/formhistory.sqlite
37u	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/cert8.db
38u	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/key3.db
39r	FIFO	pipe
40w	FIFO	pipe
41ur	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/cookies.sqlite-s
42ur	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/cookies.sqlite
43u	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/Cache_CACHE
44ur	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/places.sqlite
45u	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/places.sqlite-w
46u	REG	/home/gbt/mozilla/firefox/3gm1v04j.default/places.sqlite

FD	Direction	Data
25	read	fa
25	read	fa
25	read	fa
25	read	fa
4	read	1c00dac7af000008501000085dd0b000000000000000000000000000000
25	read	fa
25	read	fa
4	read	0f00dac78c04003000
4	read	0102dec700000000400a00300000000000000000000000000000000000000
4	read	1c00dec7af0000085010000e5de0b0000000000000000000000000000000000
25	read	fa
4	read	1c0030ca8c0400032300000005df0b0000000000000000000000000000000000
4	read	1c0031caaf0000085010000ef0b00000000000000000000000000000000000000
4	read	011832ca0000000af00000000014055605b902000000000000000000000000
4	read	010133ca0000000e11540010000240000000000000000000000000000000000
4	read	1c0033caaf000003e0100001cdf0b000000000000000000000000000000000000
4	read	1c0081ccaf00000850100003ff0b00000000000000000000000000000000000000
4	read	1c0081ccaf000003e0100004df0b00000000000000000000000000000000000000
4	read	1c0081ccaf000003e0100004df0b00000000000000000000000000000000000000
4	read	1c0081ccaf000003e01000041df0b00000000000000000000000000000000000000
25	read	fa
25	read	fa
25	read	fa
25	read	fa
25	read	fa
25	read	fa
25	read	fa

Sniff on every flows Save selected packets

Importing from exotic communication channel



- ▶ Supports hexadecimal flows
- ▶ Simple XML representation of the data