The Future of Protocol Reversing and Simulation Applied on ZeroAccess

29C3, Hamburg
December 29 2012

@Netzob
Authors...

« You talkin' to me? »
Travis Bickle
Yes, we're French!
Frédéric GUIHERY (@_sygus)

➢ IT security engineer
  ➢ Reverse engineering
  ➢ System analysis and hardening
  ➢ Trusted Computing
Georges BOSSERT (@Lapeluche)

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

Supelec CIDRe
Research team

Advisers:
- Guillaume Hiet
- Ludovic Mé
AMOSSYS, France

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

www.amossys.fr
Topics...

« Go ahead, make my day »
Harry Callahan
Reverse protocols
Simulate endpoints
Map botnets
Why reverse engineering of protocols?
Protocols are everywhere
Protocols are everywhere
(even within Tamagotchis)
Assess the robustness of implementations

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

- Ex: Fuzz the control API of a centrifuge

- 29C3 Ex:
  - « Many Tamagotchis Were Harmed in the Making of this Presentation »
  - « EXSi Beast »
Analyze traffic and identify potential data leakage

- **Ex**: Are you sure your « IP Reputation Appliance » doesn't leak your emails?
Compare the implementation of a protocol with its official specifications

- Ex: CC evaluations of crypto products
To develop a free version of a proprietary implementation

- Ex: Drew Fisher's talk @ 28C3 on Kinect RE
Current reverse engineering approach...
Have you ever *(tried to)* RE a protocol?
Did it looked like this?
Did it looked like this?

- Complex
- Time-consuming
- Mostly Manual
Did it looked like this?

- Complex
- Time-consuming
- Mostly Manual

MOSTLY VISUAL
No available tool to reverse a proprietary protocol...

Should we create one?
Let's see if we can **automate** some RE tasks...
Some reminders about protocols
Let's examine the TCP protocol
Different **types** of messages

- SYN message
- ACK message
- PUSH message
- FIN message
- RST message
- ...
Concept of encapsulation layers

- Ethernet
- IPv4
- TCP
- HTTP
- HTTP Body
Fields partitioning

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgement Number</td>
<td></td>
</tr>
<tr>
<td>Data Offset</td>
<td>Reserved</td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>Options and Padding</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>
**Relations**

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgement Number</td>
<td></td>
</tr>
<tr>
<td>Data Offset</td>
<td>Reserved</td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>Options and Padding</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

**Intra-message relations**
Relations

**Inter-message relations**

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgement Number</td>
<td></td>
</tr>
<tr>
<td>Data offset</td>
<td>Reserved</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent</td>
</tr>
<tr>
<td>Options and Padding</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgement Number</td>
<td></td>
</tr>
<tr>
<td>Data offset</td>
<td>Reserved</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent</td>
</tr>
<tr>
<td>Options and Padding</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>
## Contextual values

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgement Number</td>
<td></td>
</tr>
<tr>
<td>Data offset</td>
<td>Reserved</td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>Options and Padding</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>
## Application-level values

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgement Number</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Data offset</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Options and Padding</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>
Sequence of valid messages
Let's find a **model** that covers protocol attributes
Academics are very good with * models :) 

* sometimes useless
A Communication Protocol is made of 5 distinct parts ...
a service (1/5)
a service (1/5)
some **assumptions** about the environment (2/5)
a vocabulary of messages (3/5)
the **encoding** (format) of each message (4/5)
the procedure rules (5/5)
Yes, that was an academic model
Reduced model for a Protocol

- a vocabulary → a list of **Message Format**
- a grammar → **State Machine**
Introducing Netzob ...
Goals of Netzob

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

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

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

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

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

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

Netzob's Sponsors

@Netzob
« State of the art » boundaries

- Fuzzing
- Language Theory
- Reverse Engineering
- Grammar Inference
- Botnet Behavioural Analysis

Netzob

Sum of human knowledge

The unknown

@Netzob
NEW « State of the art » boundaries

Netzob

Fuzzing

Language Theory

Reverse Engineering

Grammar Inference

Botnet Behavioural Analysis

New sum of human knowledge

The unknown

Based on an original idea of Matt Might
RE Zero Access C&C protocol with Netzob
Zero Access (aka Sirefef)

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

Based on Sophos and Kindsight Reports
Zero Access (aka Sirefef)

- Multiple **P2P management messages**
  - Peers directory retrieval
  - Files directory retrieval
- **UDP & TCP connections**
  - UDP for messages (udp:16464)
  - TCP for data
- **Hard coded Bootstrap Peers**
  - Ex: 68.51.108.245, (...), 216.211.181.226
Let's play with its P2P protocol
Requirements
Few real communication traces

ZAccess: some traces were provided by

Kevin McNamee

an infected machine

Contagio (http://contagiodump.blogspot.com)
A **confined** environment and **the binary**

Adapted Virtual Machines + Firewalls + Torify + management system
A confined environment and the binary

Adapted Virtual Machines + Firewalls + Torify + management system

**Warning**: Consider legal issues before dealing with this!
Step 1 : Get messages
Capture **dataflows**

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

@Netzob
Capture dataflows

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

@Netzob
Split dataflows in messages
(sub protocol knowledge, time based, delimiter...)

Message 1

Message 2

Message 3

Message 4
Netzob framework

Filter imported messages
Choose layer of import

@Netzob
Step 2 : RE vocabulary
Abstract messages
1 message = a sorted received or sent sequence of bits

0101101010010001010101101001011101010001010010
1 message = a sorted **received or sent** sequence of bits

specific to a context

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

010110101001000101010110100101011101010001010010
We have to decontextualize messages.

The IDEA:

Regroup messages by similarity and find contextual variations.
We consider similar messages based on their common partitioning
Messages are splitted in Fields using
Messages are splitted in **Fields** using

- Simple Alignment
- Delimiter-based Alignment
- Sequence Alignment

| 3e341eb5ce | 4c068e | c2d5baed3a | 331938 | 3b108271e8 |
| dc18fcb8ce | 4c068e | 2da8f3e33a | 331938 | cf48cd8fe8 |
| dc18fcb8ce | 4c068e | 2da8f3e33a | 331938 | cf48cd8fe8 |
Messages are splitted in **Fields** using

- Simple Alignment
- Delimiter-based Alignment
- Sequence Alignment
Messages are splitted in **Fields** using

- Simple Alignment
- Delimiter-based Alignment
- Sequence Alignment

**Needleman & Wunsch**

<table>
<thead>
<tr>
<th>3a8</th>
<th>70</th>
<th>832f65bd867ad2</th>
<th>00</th>
<th>d9aeddcd</th>
</tr>
</thead>
<tbody>
<tr>
<td>3a8</td>
<td>70</td>
<td>832f65bd867ad2</td>
<td>00</td>
<td>d9aeddcd</td>
</tr>
<tr>
<td>70</td>
<td></td>
<td>c400</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
<td>c400</td>
<td>00</td>
<td></td>
</tr>
</tbody>
</table>
But WTF is Needleman & Wunsch?
Sequence alignment with Needleman-Wunsh applied to RE of protocols (c.f. Marshall Bedoe)

We start with 2 messages

70 83 2f 65 bd 86 7a d2 00
70 c4 00 00
Sequence alignment with Needleman-Wunsh

<table>
<thead>
<tr>
<th></th>
<th>70</th>
<th>83</th>
<th>2f</th>
<th>65</th>
<th>bd</th>
<th>86</th>
<th>7a</th>
<th>d2</th>
<th>00</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We build a distance matrix
### Sequence alignment with Needleman-Wunsch

<table>
<thead>
<tr>
<th></th>
<th>70</th>
<th>83</th>
<th>2f</th>
<th>65</th>
<th>bd</th>
<th>86</th>
<th>7a</th>
<th>d2</th>
<th>00</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**We initialize the matrix**
Sequence alignment with Needleman-Wunsh

We fill the matrix with the formula:

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

- **S**: Match/Mismatch score (+/- 10)
- **W**: Gap score (0)
Sequence alignment with Needleman-Wunsh

We fill the matrix with the formula:

\[ M(i,j) = \text{Max}(M(i-1, j-1) + S, M(i, j-1) + W, M(i-1, j) + W) \]
Sequence alignment with Needleman-Wunsh

<table>
<thead>
<tr>
<th></th>
<th>70</th>
<th>83</th>
<th>2f</th>
<th>65</th>
<th>bd</th>
<th>86</th>
<th>7a</th>
<th>d2</th>
<th>00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>0</td>
<td><strong>10</strong></td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>c4</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td><strong>20</strong></td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td><strong>20</strong></td>
</tr>
</tbody>
</table>

We fill the matrix
Sequence alignment with Needleman-Wunsh

<table>
<thead>
<tr>
<th></th>
<th>70</th>
<th>83</th>
<th>2f</th>
<th>65</th>
<th>bd</th>
<th>86</th>
<th>7a</th>
<th>d2</th>
<th>00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>c4</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

We do a traceback
Sequence alignment with Needleman-Wunch

<table>
<thead>
<tr>
<th></th>
<th>70</th>
<th>83</th>
<th>2f</th>
<th>65</th>
<th>bd</th>
<th>86</th>
<th>7a</th>
<th>d2</th>
<th>00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>c4</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

We compute the common pattern

```
70 83 2f 65 bd 86 7a d2 00
70 c4 00 -- -- -- -- -- 00
```
Sequence alignment with Needleman-Wunsh

We finally build a regex

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

```
(70) (.*{2,7}) (00)
```
Symbol x $\rightarrow \left[ \ldots, 0x70, (.*\{4,14\}), 0x00, \ldots \right]$
### Static Fields

<table>
<thead>
<tr>
<th>Symbol</th>
<th>0x70</th>
<th>(832f65bd867ad2)</th>
<th>00</th>
<th>(d9aeddc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Dynamic Fields

Symbol \(x \rightarrow [..., 0x70, (.*\{4,14\}, 0x00, ...] \)
How to measure **similarity** between two messages?
Measure the **Quality of Fields**

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

- **Messages have Nothing in common**
- **Messages are identicals**
Similarity scores between messages

- S1: ratio of **dynamic fields / bytes**
- S2: ratio of **common dynamic bytes**
Similarity scores between messages

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

The design of Netzob allows for inclusion of future factors
Similarity scores between messages

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

\[ S1 = \frac{1}{1 + 2} \]
Similarity scores between messages

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

\[
\begin{align*}
S2 &= \frac{2}{7} \\
70\ 83\ 2f\ 65\ bd\ 86\ 7a\ d2\ 00 \\
70\ c4\ 00\ --\ --\ --\ --\ --\ 00
\end{align*}
\]
Similarity scores between messages

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

Normalized similarity score: \( S \)
How to retrieve *groups* of similar messages?
Hierarchical Clustering by similarities:

**Similarity matrix**

**UPGMA**

- **Filling of a similarity matrix**
- **Iteratively merge the 2 most similar messages**
UPGMA creates a similarity tree
**UPGMA** creates a similarity tree and facilitates clustering.
## ZAccess Example

### Results of Clustering and Sequence Alignment

<table>
<thead>
<tr>
<th>Nom</th>
<th>Message</th>
<th>Champ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol 10</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Symbol 25</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>Symbol 23</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Symbol 5</td>
<td>8</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field 0</th>
<th>Field 1</th>
<th>Field 2</th>
<th>Field 3</th>
<th>Field 4</th>
<th>Field 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(.8)</td>
<td>(4c74657200000000)</td>
<td>(.2)</td>
<td>(000000)</td>
<td>(.8)</td>
<td>(00000000)</td>
</tr>
<tr>
<td>hex</td>
<td>hex</td>
<td>hex</td>
<td>hex</td>
<td>hex</td>
<td>hex</td>
</tr>
<tr>
<td>01a76c9f</td>
<td>4c74657200000000</td>
<td>03</td>
<td>000000</td>
<td>bd584ae8</td>
<td>00000000</td>
</tr>
<tr>
<td>9782e4fb</td>
<td>4c74657200000000</td>
<td>06</td>
<td>000000</td>
<td>c725c5ca</td>
<td>00000000</td>
</tr>
<tr>
<td>46o9c0a9</td>
<td>4c74657200000000</td>
<td>05</td>
<td>000000</td>
<td>d0b4d6d1</td>
<td>00000000</td>
</tr>
<tr>
<td>b7dadb61</td>
<td>4c74657200000000</td>
<td>0e</td>
<td>000000</td>
<td>deed851e</td>
<td>00000000</td>
</tr>
<tr>
<td>db34786e</td>
<td>4c74657200000000</td>
<td>0c</td>
<td>000000</td>
<td>e029f3c8</td>
<td>00000000</td>
</tr>
<tr>
<td>738a2cf6</td>
<td>4c74657200000000</td>
<td>07</td>
<td>000000</td>
<td>b6d28e36</td>
<td>00000000</td>
</tr>
<tr>
<td>badfa0e7</td>
<td>4c74657200000000</td>
<td>05</td>
<td>000000</td>
<td>b85b8fda</td>
<td>00000000</td>
</tr>
<tr>
<td>5d2676f9</td>
<td>4c74657200000000</td>
<td>04</td>
<td>000000</td>
<td>6541e62e</td>
<td>00000000</td>
</tr>
</tbody>
</table>
Abstract fields

to decontextualize messages
Message format model

Allows multiple partitionment strategies per symbols
Full message format model

Raw message

Layer 1
- Field 1
  - Size
    - Fix: \{n\}
    - Variable: \{n, m\}
  - Interpretation attributes
    - Unit size: bit, octet, double octet, etc.
    - Endianess
    - Sign
    - Representation: decimal, octal, hex, ASCII, etc.
    - Transformation: base64, bz2, gzip, etc.
  - Definition domain
    - Static value
    - Random value
    - Intra-message dependency
    - Inter-message dependency
    - Environmental dependency
- Field 2

Layer 2
Interlude
'
cauze +/- 60 slides left
Let's do 5 minutes of knitting
Transformations
How to handle

Encoded values

(XOR, ASN.1) ?
How to handle

**Encrypted values**

(symmetric, asymmetric, ...)?
Messages include **Transformed Fields**

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

Transform raw bytes into application-level bytes

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

Transform raw bytes into application-level bytes

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

Transform raw bytes into application-level bytes

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

Transform raw bytes into application-level bytes

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

**Ex: ZeroAccess XOR-based obfuscation**

```python
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*

<table>
<thead>
<tr>
<th></th>
<th>28948dbec9c0d1998381a333</th>
<th></th>
<th>28948dbec9c0d1998381a333</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b59e6155</td>
<td>ad4d699b</td>
<td>4c068ece</td>
<td>4adb017b</td>
<td>9b72a59a</td>
</tr>
<tr>
<td>4adb017b</td>
<td>d9fcb898</td>
<td>4c068ece</td>
<td>8c43c72c</td>
<td></td>
</tr>
<tr>
<td>8c43c72c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>4c7465720000000001000000</th>
<th></th>
<th>8b4e2efc</th>
<th>00000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>87ee1533</td>
<td>bd71e2fd</td>
<td>00000000</td>
<td>78ab751d</td>
<td></td>
</tr>
<tr>
<td>78ab751d</td>
<td>fffffffffff</td>
<td>00000000</td>
<td>be33b34a</td>
<td></td>
</tr>
<tr>
<td>be33b34a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Search for relations
How to handle when the value depends on something?
Binary ID, Affiliate ID, Filenames, etc.

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

Size Fields, CRCs, etc.
The idea

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

- **Correlate** field's size and values with the « Maximal Information Coefficient » (M.I.N.E.)
- **Qualify** correlated fields
Generate Pairs of data for each field:

Simple way:

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

Add more:

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

- Measure **dependences** between each pairs
  - Support noisy datasets
- Rank pairs by their score
\[ \text{MINE}(\text{Value}(F1), \text{Size}(F2)) = 1 \]

→ Typical Size Field Relation

\[ \text{MINE}(\text{Value}(F7), \text{CRC32}(\text{Value}(F1), \text{Value}(F2))) = 1 \]

→ Typical CRC Relation
\[
\text{MINE}(\text{Value}(F1), \text{Size}(F2)) = 1
\]
→ Typical Size Field Relation

\[
\text{MINE}(\text{Value}(F7), \text{CRC32}(\text{Value}(F1), \text{Value}(F2))) = 1
\]
→ Typical CRC Relation
« Environmental » dependencies
During packets capture:

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

During vocabulary inference:

- **Search for meta-data** in messages
Step 3 : RE grammar
Sequence of valid exchanged symbols.

→ IO Automata
Sequence of valid exchanged symbols.

→ IO Automata

State 1 \(\xrightarrow{\text{Attack !}}\) State 2

Success
But answers depend on the environment

State 1 \rightarrow \text{Attack!} \rightarrow \text{Fail} \rightarrow \text{State 2}
Our model (SMMDT)

→ Add probabilities on output messages

State 1 → State 2

80 % : Fail
20 % : Success
Our model (SMMDT)
→ Add the « reaction time »

State 1 ➔ State 2

- **Attack !**
- 80% : Fail (2000ms)
- 20% : Success (10ms)
Active Grammatical Inference Process

→ Angluin L*a Algorithm
Active Grammatical Inference Process

→ Angluin L*a Algorithm
Generating traffic...
Netzob can generate traffic that:

- Follows the **inferred message format**
- Respects the **state machine**
Emulation of different kind of actors and flows

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

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

Ex: TLS with TCP session initiated by the server
Abstraction from the communication channel

- TCP messages
- USB channel
- IPC flow
- Raw file
Memory mechanism

› Some received values are **memorized**...
› ...and **reinjected** in future messages
› Also handles contextual values (IP, time, etc.)
Abstraction and contextualization principles

Input device

Input flow

Communication channel library

« 70dde8fc00000003 »
Abstraction and contextualization principles

Input device

Communication channel library

Input messages

Vocabulary

Abstraction layer

getL symbol

[ $LOCAL_IP, $ID ]

Handles execution context
Abstraction and contextualization principles

Input device

Communication channel library

Vocabulary

Abstraction layer

Grammar model (SMMDT)

Clock

« getL » symbol

« retL » symbol

Input symbols

Output symbols
Abstraction and contextualization principles

- Input device
- Communication channel library
- Abstraction layer
- Grammar model (SMMDT)
- Clock
- Memory
- Vocabulary
- Input symbols
- Output symbols

previous peers IP

@Netzob
Abstraction and contextualization principles

Input device -> Communication channel library

Communication channel library -> Abstraction layer

Abstraction layer -> Memory

Memory -> Grammar model (SMMDT)

Grammar model (SMMDT) -> Clock

Clock -> Vocabulary

Vocabulary -> Abstraction layer

Abstraction layer -> Output device

Output device -> Output flow

Output flow -> response containing peers list
Demo 1: **reversing the protocol of ZA**

- Import samples of communications
- De-obfuscate exchanges
- Find similar messages
- Split messages in fields
- Abstract Fields
- Search for relations
Demo 2: retrieve the P2P zombie directory

- Simulation of a realistic zombie
- Map the peers neighbours relations
Future improvements of Netzob...
Integrated smart fuzzing, by leveraging the simulator engine

→ Allows to fuzz undocumented and proprietary protocols
Integrated smart fuzzing, by leveraging the simulator engine

→ Allows to fuzz undocumented and propriatery protocols
Support of more communication channels

- USB
- IOCTL
- API (ssl_read, ssl_write, etc.)
Export protocol model in more 3rd party products *(coming soon)*

- Wireshark
- Scapy
- Peach Fuzzer

Allows protocol dissection with established tools
Export protocol model in more 3rd party products *(coming soon)*

- Wireshark
- Scapy
- Peach Fuzzer

Allows fuzzing of unknown protocols with well-known tools
Conclusion...
‣ Protocol RE automation domain is quite active at the academic level

‣ But no real tool available...

‣ Netzob tries to fill this lack by
  ▶ Supporting academic researches
  ▶ Being usable in operational context
- Open to all kind of contributions
  - Feedback
  - Bug fix
  - Feature proposal / implementation
  - Translation
  - ...

@Netzob
Thanks for you attention!

Any questions?

www.netzob.org
@netzob
Image licences

http://www.flickr.com/photos/arne-halvorsen/3346841686/in/pool-1093738@N24
  CC-BY-NC / aha42 | tehaha
http://www.flickr.com/photos/torek/3280152297/sizes/l/in/pool-1093738@N24/
  CC-BY-ND ar kirainet
http://www.flickr.com/photos/jdawg/295956572/in/pool-security_theater/
  CC-BY-NC r lawgeek
http://www.flickr.com/photos/tjblackwell/7324060440/sizes/l/in/pool-36004471@N00/
  CC-BY-NC r tj.blackwell
http://www.flickr.com/photos/sterlinginely/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