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
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
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
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

SECURITY WILL BE RIGHT BACK
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

```
<table>
<thead>
<tr>
<th>data offset</th>
<th>reserved</th>
<th>URG</th>
<th>ACK</th>
<th>PSH</th>
<th>RST</th>
<th>SYN</th>
<th>FIN</th>
<th>window</th>
</tr>
</thead>
<tbody>
<tr>
<td>checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>options</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>padding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>données</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
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
« State of the art » boundaries

Fuzzing
Language Theory
Reverse Engineering
Grammar Inference
Botnet Behavioural Analysis

Sum of human knowledge

Netzob

The unknown
NEW « State of the art » boundaries

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
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, ...)

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

Message 1

Message 2

Message 3

Message 4
Netzob framework

Import

Capture

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
- Delimiter-based Alignment
- Sequence Alignment

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3e341eb5ce</td>
<td>4c068e</td>
<td>c2d5baed3a</td>
<td>331938</td>
<td>3b108271e8</td>
</tr>
<tr>
<td>dc18fcb8ce</td>
<td>4c068e</td>
<td>2da8f3e33a</td>
<td>331938</td>
<td>cf48cd8fe8</td>
</tr>
<tr>
<td>dc18fcb8ce</td>
<td>4c068e</td>
<td>2da8f3e33a</td>
<td>331938</td>
<td>cf48cd8fe8</td>
</tr>
</tbody>
</table>
Messages are split into Fields using:

- Simple Alignment
- Delimiter-based Alignment
- Sequence Alignment

<table>
<thead>
<tr>
<th>Address</th>
<th>Checksum</th>
<th>Data</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cfa0</td>
<td>4c</td>
<td>5519e43e2e27fa6916313dc89ac77569a00d</td>
<td>4c</td>
</tr>
<tr>
<td>Cfa0</td>
<td>4c</td>
<td>5519e43e2e27fa6916313dc89ac77569a00d</td>
<td>4c</td>
</tr>
<tr>
<td>Cfa0</td>
<td>4c</td>
<td>5519e43e2e27fa6916313dc89ac77569a00d</td>
<td>4c</td>
</tr>
</tbody>
</table>
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>d9aeddc</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>c400</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field 1</td>
<td>Field 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3a8</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3a8</td>
<td>832f65bd867ad2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>c400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>c400</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Static Fields
<table>
<thead>
<tr>
<th>Static Fields</th>
<th>Dynamic Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>3a8 70</td>
<td>832f65bd867ad2 00 d9aedddc</td>
</tr>
<tr>
<td>3a8 70</td>
<td>832f65bd867ad2 00 d9aedddc</td>
</tr>
<tr>
<td>70</td>
<td>c400 00</td>
</tr>
<tr>
<td>70</td>
<td>c400 00</td>
</tr>
</tbody>
</table>
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

<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-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>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>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>c4</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>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>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>
</tbody>
</table>

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

We fill the matrix with to 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

<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>?</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 fill the matrix with to 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>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>30</td>
</tr>
</tbody>
</table>

We fill the entire 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>30</td>
</tr>
</tbody>
</table>

We do a traceback
Sequence alignment with Needleman-Wunsh

We compute the common pattern

```plaintext
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

\[
\begin{align*}
70 & 83 \ 2f \ 65 \ bd \ 86 \ 7a \ d2 \ 00 \\
70 & \ c4 \ 00 \ -- \ -- \ -- \ -- \ -- \ 00 \\
\hline
(70) \ & \ (.*\{2,7\}) \ & \ (00)
\end{align*}
\]
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

\[
F1 = \frac{100}{(1 + 2) \times 2}
\]

\[
F2 = \frac{100}{2 \times 7}
\]
Similarity factors between messages

F1: ratio of dynamic fields / static bytes

F2: ratio of common dynamic bytes

Normalized similarity score
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

<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>Symbol 5</th>
</tr>
</thead>
</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>
<th>Field 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(.8)</td>
<td>(4c74657200000000)</td>
<td>(.2)</td>
<td>(000000)</td>
<td>(.8)</td>
<td>(0000000000)</td>
<td>hex</td>
</tr>
<tr>
<td></td>
<td>hex</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01a76c9f</td>
<td>4c74657200000000</td>
<td>03</td>
<td>000000</td>
<td>bd584ae8</td>
<td>0000000000</td>
<td>d2</td>
</tr>
<tr>
<td>9782e4fb</td>
<td>4c74657200000000</td>
<td>06</td>
<td>000000</td>
<td>c725c5ca</td>
<td>0000000000</td>
<td>b7</td>
</tr>
<tr>
<td>4609c0a9</td>
<td>4c74657200000000</td>
<td>05</td>
<td>000000</td>
<td>d0b4d6d1</td>
<td>0000000000</td>
<td>b6</td>
</tr>
<tr>
<td>b7dadb61</td>
<td>4c74657200000000</td>
<td>0e</td>
<td>000000</td>
<td>deed851e</td>
<td>0000000000</td>
<td>aa</td>
</tr>
<tr>
<td>db34786e</td>
<td>4c74657200000000</td>
<td>0c</td>
<td>000000</td>
<td>e029f3c8</td>
<td>0000000000</td>
<td>bd</td>
</tr>
<tr>
<td>738a2cf6</td>
<td>4c74657200000000</td>
<td>07</td>
<td>000000</td>
<td>b6d28e36</td>
<td>0000000000</td>
<td>8f</td>
</tr>
<tr>
<td>badfa0e7</td>
<td>4c74657200000000</td>
<td>05</td>
<td>000000</td>
<td>b85b8fda</td>
<td>0000000000</td>
<td>cd</td>
</tr>
<tr>
<td>5d9c756f9</td>
<td>4c74657200000000</td>
<td>04</td>
<td>000000</td>
<td>854af26f</td>
<td>0000000000</td>
<td>6f</td>
</tr>
</tbody>
</table>
Abstract fields
Message format model

- Raw message
- Layer 1
  - Field 1
- Layer 2
  - Field 2

Allows multiple partitionment strategies per symbols
ZAccess Example

The bytes distribution helps to identify multiple partitionment strategies
Full message format model

- Raw message
  - Layer 1
    - Field 1
    - Field 2
  - Layer 2

- 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

- Semantic
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

```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("<l", binMessage[i:i+4])[0]
        xoredSubData = subData ^ key
        result.append(struct.pack("<l", 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>Value</th>
<th>XORed Value</th>
<th>CRC32 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>b59e6155</td>
<td>28948dbec9c0d1998381a333</td>
<td>ad4d699b 4c068ece</td>
</tr>
<tr>
<td>4adb017b</td>
<td>28948dbec9c0d1998381a333</td>
<td>9b72a59a 4c068ece</td>
</tr>
<tr>
<td>8c43c72c</td>
<td>28948dbec9c0d1998381a333</td>
<td>d9fc898 4c068ece</td>
</tr>
<tr>
<td>87ee1533</td>
<td>4c74657200000000010000000</td>
<td>8b4e2efc 000000000</td>
</tr>
<tr>
<td>78ab751d</td>
<td>4c74657200000000010000000</td>
<td>bd71e2fd 000000000</td>
</tr>
<tr>
<td>be33b34a</td>
<td>4c74657200000000010000000</td>
<td>ffffffff 000000000</td>
</tr>
</tbody>
</table>
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 pair
- 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. \[ \rightarrow \text{IO Automata} \]

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

\[ \text{State 1} \rightarrow \text{State 2} \]

Success
But answers depends on the environment
Our model (SMMDT)

→ Add probabilities on output messages

State 1 → Attack ! → State 2

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

State 1 → State 2

- 80 % : Fail (2000ms)
- 20 % : Success (10ms)
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
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

[Image of network actor configuration]
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

Vocabulary

Abstraction layer

retL symbol
[$LOCAL_IP, $ID$]

Handles execution context

Input messages
Abstraction and contextualization principles

Input device

Communication channel library

Vocabulary

Abstraction layer

Grammar model (SMMMDT)

Input symbols

Output symbols

« getL » symbol

« retL » symbol

@Netzob
Abstraction and contextualization principles

- Input device
- Communication channel library
- Abstraction layer
- Memory
- Vocabulary
- Grammar model (SMMMDT)
- Clock

Input symbols to Output symbols

previous peers IP
Abstraction and contextualization principles

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

Output flow

Response containing peers list
Use cases of traffic simulation

Use case 1: simulation of a P2P botnet

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

@Netzob
netzob.org
Use cases of traffic simulation

Use case 2: retrieve the P2P zombie directory

- Zero Access P2P map visualization
- Mapping 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 proprietary 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
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
  • Osyp 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

@Netzob