

Fuddly: un *framework* de fuzzing et de manipulation de données

Eric Lacombe¹

¹eric.lacombe@{airbus.com, security-labs.org}
Airbus Operations S.A.S.



Limites constatées avec les *frameworks* de fuzzing actuels

- **Flexibilité insuffisante** dans la représentation des données
 - Niveau d'**expressivité** insatisfaisant
 - **Modifications** opaques des données modélisées
 - Pas de **représentations multiples** possible au sein d'un même modèle
- Frameworks évolués souvent **incompatibles avec nos contraintes** :
 - **Indisponibilité** du **code source** des cibles
 - Architectures **non-x86** (voire exotiques)
 - Moyens d'observabilité **spécifiques** à l'avionique

Plan

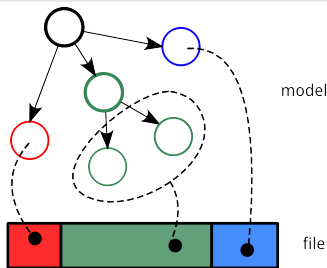
- 1 Modélisation des données
- 2 Manipulation des données
- 3 Automatisation

Plan

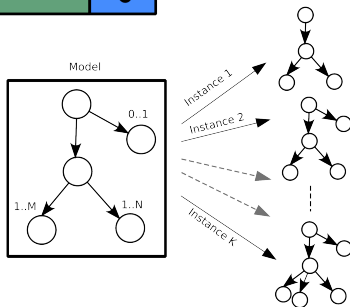
- 1 Modélisation des données
- 2 Manipulation des données
- 3 Automatisation

En bref

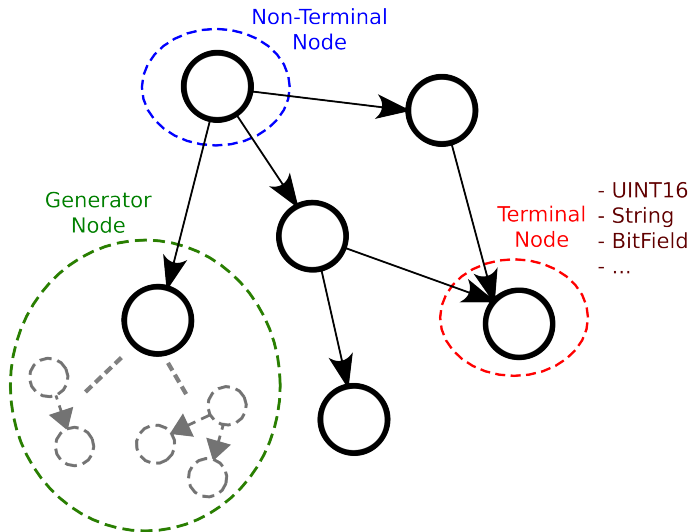
- Modélisation via **graphe orienté acyclique**



- Génération de données
- Absorption de données brutes
 - Modifications ultérieures
 - Analyse

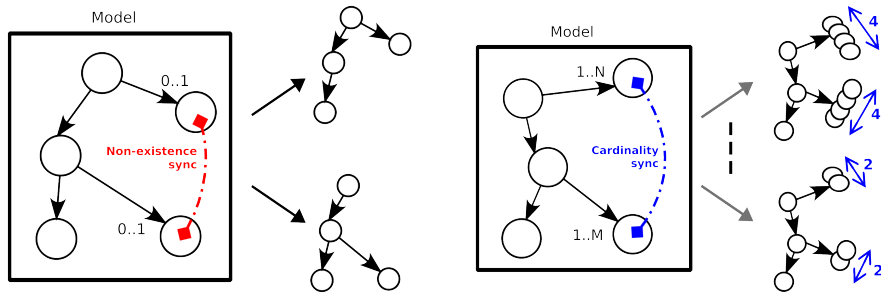


Types de nœuds



Types de liens entre nœuds

- Liens de parenté qui ordonnent les nœuds
- Liens entre Générateurs et ses nœuds paramètres
- Liens de contraintes¹



1. Évite le recours à des Générateurs pour des critères relationnels simples. »

Mais encore ?

- Configurations multiples possible pour chaque nœuds
 - Vues différentes d'un même bloc de données
 - Choix adaptées à la génération vs. l'absorption
- Abstractions pour regrouper des nœuds (ex : page d'un PDF)
 - Autorise des opérations de plus haut-niveau
 - Tout en conservant la structure du modèle
- Attributs divers sur les nœuds (Mutable, Determinist, Finite, ...)
- Ajout possible de méta-informations quelconques aux nœuds
- ...

Plan

- 1 Modélisation des données
- 2 Manipulation des données
- 3 Automatisation

Manipulation de données

Primitives aidant à la manipulation des données :

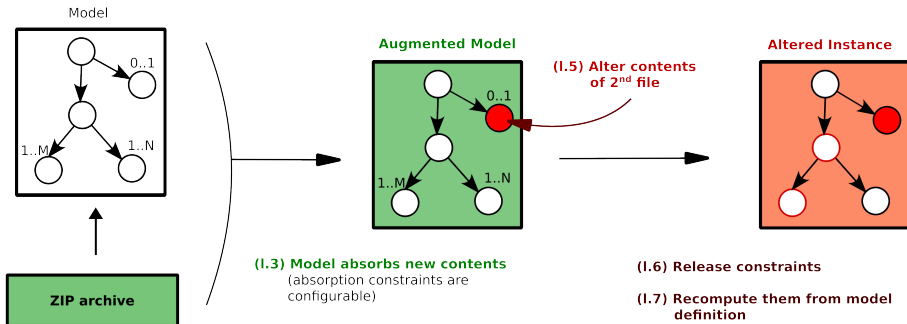
- Recherche évoluée au sein du graphe représentant la donnée
- Modification avec ou sans préservation des contraintes

Regroupement des traitements sur les données :

- Disrupteurs spécifiques aux modèles
- Disrupteurs génériques qui génèrent de nouvelles données :
 - Par analyse des spécificités des nœuds terminaux (cas limites)
 - En itérant sur les potentielles configurations alternatives des nœuds
 - Par troncature de la donnée en entrée, ...

Exemple – Absorption et modification d'une archive ZIP

```
1 zip = dm.get_data( 'ZIP' )
2 zip.set_current_conf( 'ABS', recursive=True )
3 zip.absorb( zip_buff, constraints=AbsNoCsts(size=True, struct=True) )
4
5 zip[ 'ZIP/file_list/file:2/data' ].absorb( b'TEST', constraints=AbsNoCsts() )
6 zip.unfreeze( only_generators=True )
7 zip.get_value()
```



Plan

- 1 Modélisation des données
- 2 Manipulation des données
- 3 Automatisation

En bref

Automatisation des traitement de données :

- Altération pas à pas du graphe via pattern *“visiteur-modificateur”*
- Disrupteurs à état
- Chaînage des disrupteurs

Automatisation des interactions avec la cible :

- Opérateurs virtuels
- Infrastructure de **monitoring** (sondes)

Conclusion

fuddly TODO list :

- Documentation ;)
- Support de nouveaux formats de données
- Ajout de nouveaux *Disrupteurs* génériques
- Définition de *template* de nœuds *Générateur*
- Intégration du DBI Pin (Intel) dans l'infrastructure de *monitoring*
- etc.

One last thing...

- `fuddly` est disponible sous licence GPL à l'adresse :

<https://github.com/k0retux/fuddly>

- `fuddly` supporte² actuellement les formats :

ZIP, PDF, PNG, JPG, descripteurs USB, ...

(mais aussi des protocoles avioniques qui n'intéressent pas grand monde ici ;))

2. de façon plus ou moins fine et/ou complète

Merci de votre attention ...

... Des questions ?

Backup - Modélisation basique d'un PNG

```
1 png_desc = \  
2 {'name': 'PNG_model',  
3  'contents': [  
4      {'name': 'sig',  
5       'contents': String(val_list=[b'\x89PNG\r\n\x1a\n'], size=8)},  
6      {'name': 'chunks',  
7       'qty': (2,200),  
8       'contents': [  
9          {'name': 'len',  
10           'contents': UINT32_be()},  
11          {'name': 'type',  
12           'contents': String(val_list=['IHDR', 'IEND',  
13                                     'IDAT', 'PLTE'], size=4)},  
14          {'name': 'data_gen',  
15           'type': MH.Generator,  
16           'contents': lambda x: Node('data', value_type= \  
17                                   String(size=x[0].get_raw_value())),  
18           'node_args': ['len']},  
19          {'name': 'crc32_gen',  
20           'type': MH.Generator,  
21           'contents': g_crc32,  
22           'node_args': ['type', 'data_gen'],  
23           'clear_attrs': [NodeInternals.Freezable]}  
24      ]}  
25 ]}
```

Backup - Visualisation des données

```
[TestNode]
-----
[0] TestNode [NonTerm]
  \_ (1) TestNode/val1 [BitField] size=3B
      \_ (+|2: 1 |1: 10 |0: 00111000001111100110 |-) 13043686
          \_ raw: b'\xc7\x07\xe6'
      \_ (1) TestNode/val1:2 [BitField] size=3B
          \_ (+|2: 0 |1: 10 |0: 000100110010100100011 |-) 4351267
              \_ raw: b'Be#'
      \_ (1) TestNode/val2 [String] size=2B --> M
          \_ raw: b'OK'
  \_ [1] TestNode/middle [NonTerm]
      \_ (2) TestNode/middle/val22 [Func | node_args: TestNode/val1, TestNode/val3] size=13B
          \_ raw: b'\xc7\x07\xe6.vHddW.7Kt'
      \_ [2] TestNode/middle/USB_desc [NonTerm]
          \_ [3] TestNode/middle/USB_desc/bLength [GenFunc | node_args: TestNode/middle/USB_desc/contents]
              \_ (4) TestNode/middle/USB_desc/bLength/dyn [UINT8] size=1B
                  \_ 20 (0x14)
                  \_ raw: b'\x14'
              \_ (3) TestNode/middle/USB_desc/bDescType [UINT8] size=1B
                  \_ 3 (0x3)
                  \_ raw: b'\x03'
              \_ (3) TestNode/middle/USB_desc/contents [String] size=18B
                  \_ raw: b'b\x00l\x00a\x00b\x00l\x00a\x00.\x00.\x00.\x00'
          \_ (2) TestNode/middle/val21 [BitField] size=3B
              \_ (+|2: 1 |1: 10 |0: 100011010001110001100 |-) 13738892
                  \_ raw: b'\xd1\xa3\x8c'
              \_ (2) TestNode/middle/val2 [String] size=2B --> M
                  \_ raw: b'OK'
  \_ (1) TestNode/val3 [String] size=10B
      \_ raw: b'.vHddW.7Kt'
  \_ (1) TestNode/val3:2 [String] size=10B
      \_ raw: b'.vHddW.7Kt'
  \_ (1) TestNode/val21 [String] size=4B
      \_ raw: b'PLIP'
```