Caradoc ou l'analyse de la structure de fichiers PDF

Conférence ESSI

Travaux de stage de Guillaume Endignoux
encadré par Olivier Levillain et Jean-Yves Migeon

13 septembre 2017

Motivation initiale

PDF, un format bien connu ?

- PDF signifie *Portable Document Format*
- un format classique pour l'échange de documents

PDF, un format bien connu ?

- PDF signifie *Portable Document Format*
- un format classique pour l'échange de documents

- une idée courante est que les fichiers PDF sont *passifs*
- parallèle possible avec le format PostScript

PDF, un format bien connu ?

- PDF signifie *Portable Document Format*
- un format classique pour l'échange de documents

- une idée courante est que les fichiers PDF sont *passifs*
- parallèle possible avec le format PostScript

- en réalité, les deux formats permettent plus que la description d'un contenu visuel
- *PostScript est un véritable langage de programmation complet*
- PDF permet notamment l'inclusion de scripts (actions sur des formulaire, JavaScript)

Pourquoi s'intéresser à PDF ?

- ~~c'est amusant~~

Pourquoi s'intéresser à PDF ?

- ~~c'est amusant~~
- ~~pour écrire un *parser* en OCaml~~

Pourquoi s'intéresser à PDF ?

- ~~c'est amusant~~
- ~~pour écrire un *parser* en OCaml~~
- le format est très utilisé et est un vecteur d'attaques courant

Un sujet déjà bien couvert

État de l'art (partiel) sur la sécurité de PDF

- Raynal et al. — *Malicious origami in PDF* (2001)
- Albertini et ses nombreux fichiers polyglottes (depuis 2013)
- divers travaux de classification de PDF malveillants
- d'autres travaux démontant les outils de classification précédents
- la recherche de vulnérabilités sur les lecteurs PDF classiques

Un sujet déjà bien couvert

État de l'art (partiel) sur la sécurité de PDF

- Raynal et al. — *Malicious origami in PDF* (2001)
- Albertini et ses nombreux fichiers polyglottes (depuis 2013)
- divers travaux de classification de PDF malveillants
- d'autres travaux démontant les outils de classification précédents
- la recherche de vulnérabilités sur les lecteurs PDF classiques

Notre approche

- écrire des *parsers* maison pour comprendre de première main la réalité du terrain
- s'intéresser à la structure du fichier avant d'envisager des analyses de plus haut niveau sur le contenu
- analyser des fichiers issus d'internet
- créer des fichiers facétieux

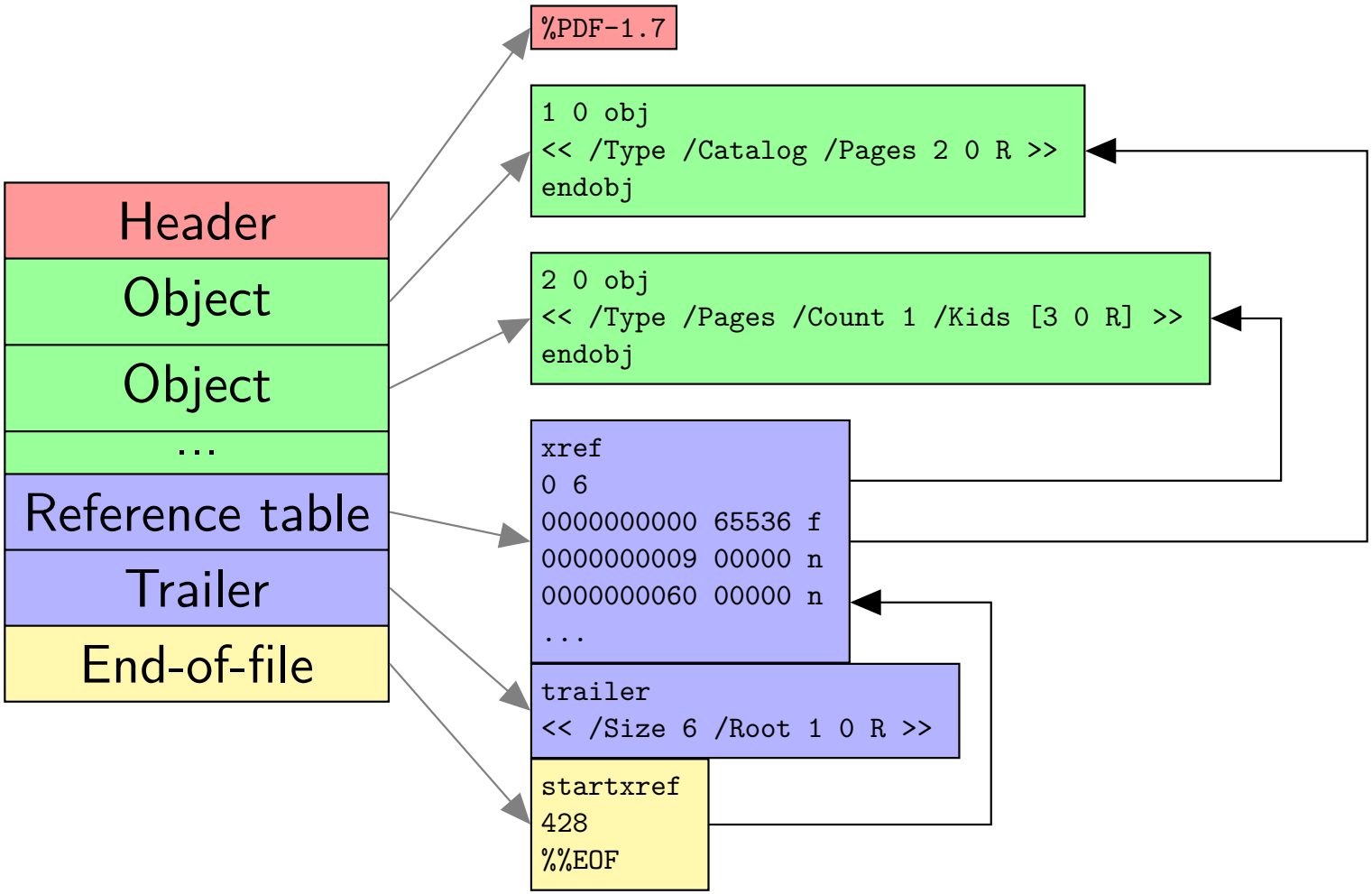
Description de PDF

PDF syntax 101

Un document PDF contient des objets :

- `null`
- des booléens : `true`, `false`
- des nombres : `123`, `-4.56`
- des chaînes de caractères : `(foo)`
- des *noms* : `/bar`
- des tableaux : `[1 2 3]`, `[(foo) /bar]`
- des *dictionnaires* : `<< /key (value) /foo 123 >>`
- des références : `1 0 obj ... endobj` et `1 0 R`
- des flux : `<< ... >> stream ... endstream`

Structure d'un fichier PDF



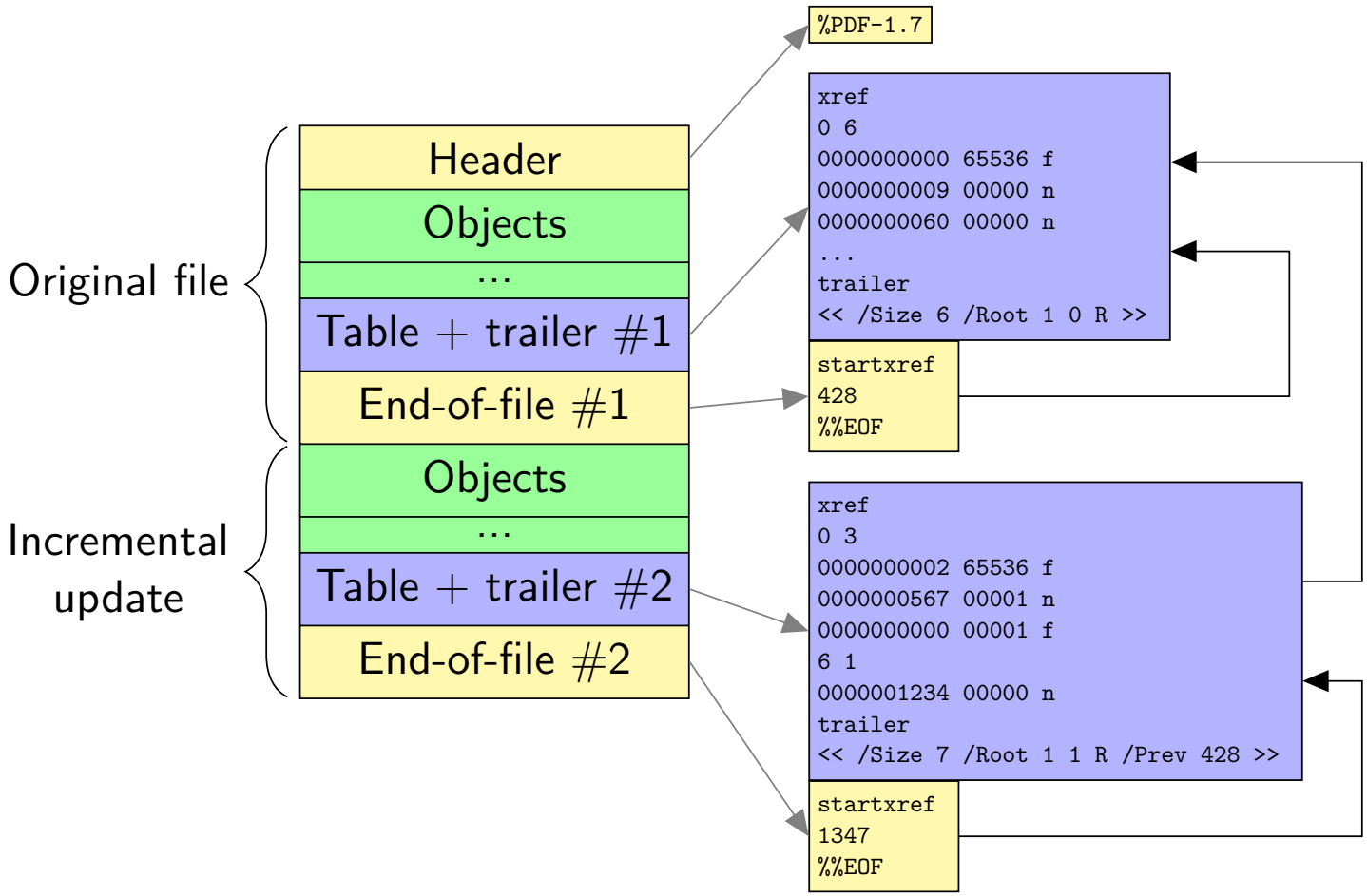
Source : `https://gendignoux.com/blog/`

Mises à jours incrémentales

Comment gérer *simplement* les modifications d'un PDF ?

Mises à jours incrémentales

Comment gérer *simplement* les modifications d'un PDF ?



Source : `https://gendignoux.com/blog/`

Compression des contenus

Afin d'optimiser la place prise par les données, ajout de *filtres* pour compresser les flux

Compression des contenus

Afin d'optimiser la place prise par les données, ajout de *filtres* pour compresser les flux

```
6 0 obj
<<
  /Filter /FlateDecode
  /Length 43
>>
stream
[ Contenu compressé ]
endstream
endobj
```

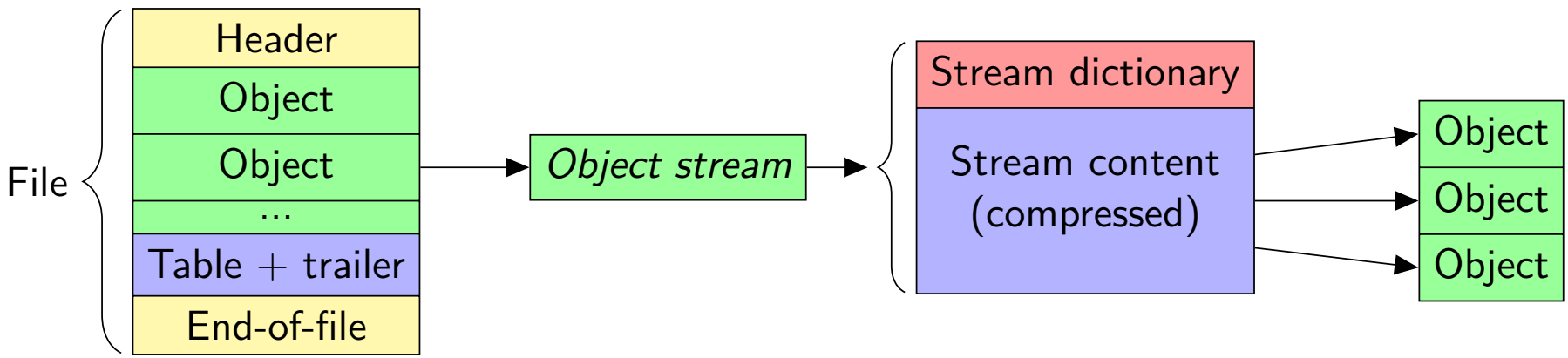
Compression de la structure

Comme le format reste très verbeux, on souhaite compresser aussi les objets

Compression de la structure

Comme le format reste très verbeux, on souhaite compresser aussi les objets

C'est l'apparition des *object streams*

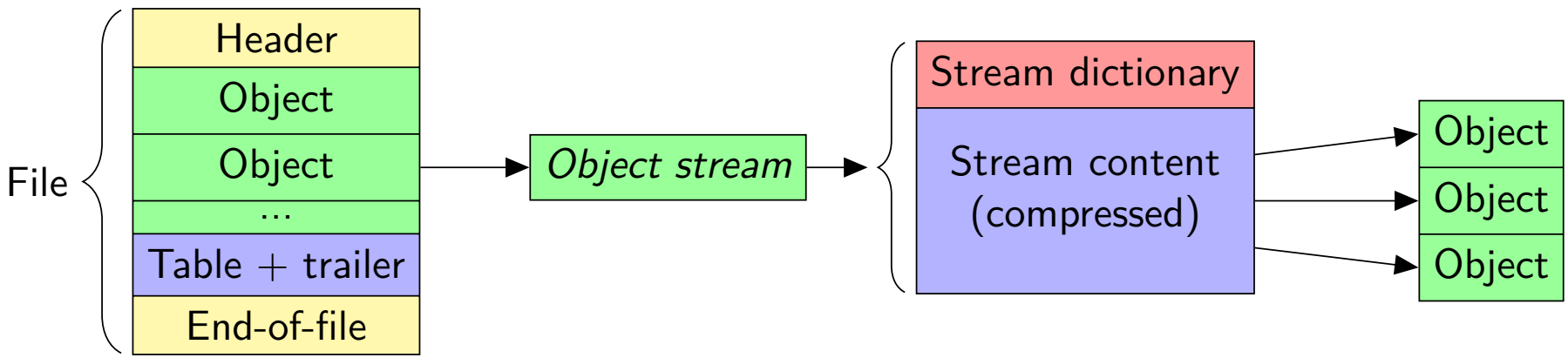


Source : `https://gendignoux.com/blog/`

Compression de la structure

Comme le format reste très verbeux, on souhaite compresser aussi les objets

C'est l'apparition des *object streams*



Source : `https://gendignoux.com/blog/`

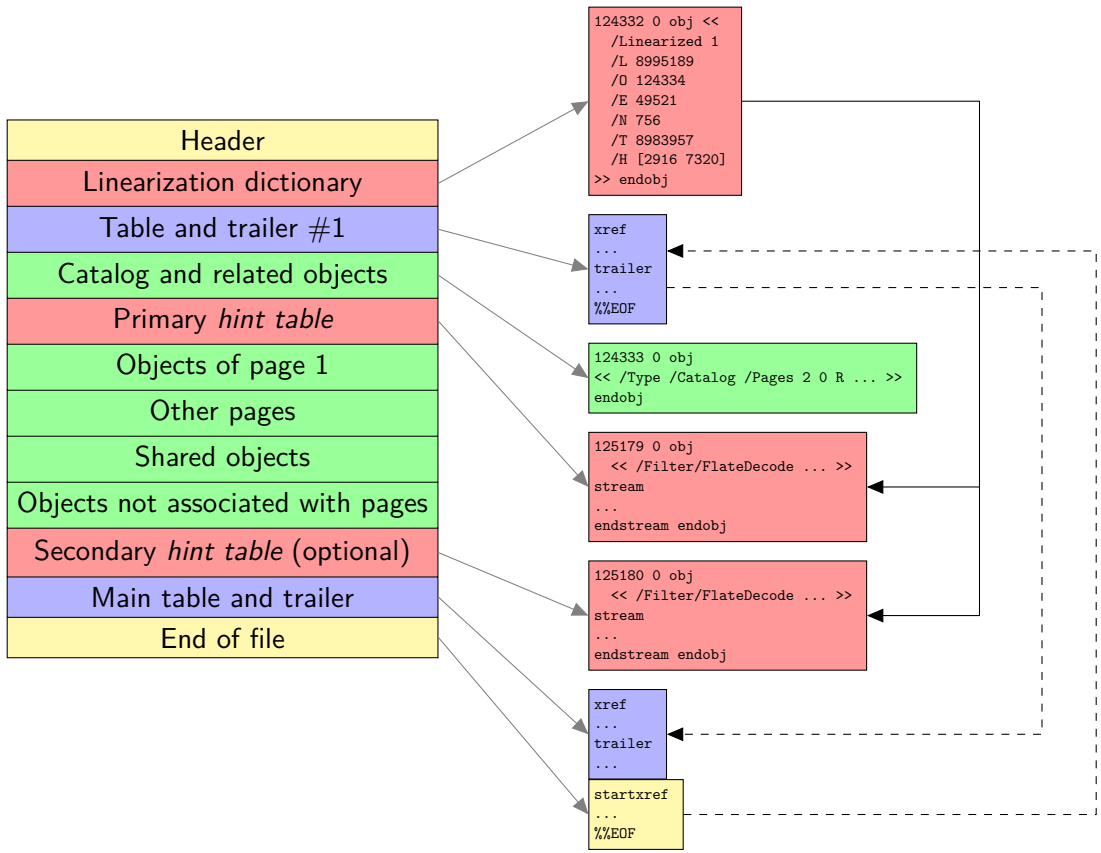
Afin d'éviter les blagues, les objets contenus dans un *object stream* ne peuvent pas être eux mêmes des flux

Et l'expérience utilisateur ?

Comment afficher un fichier avant de l'avoir téléchargé complètement ?

Et l'expérience utilisateur ?

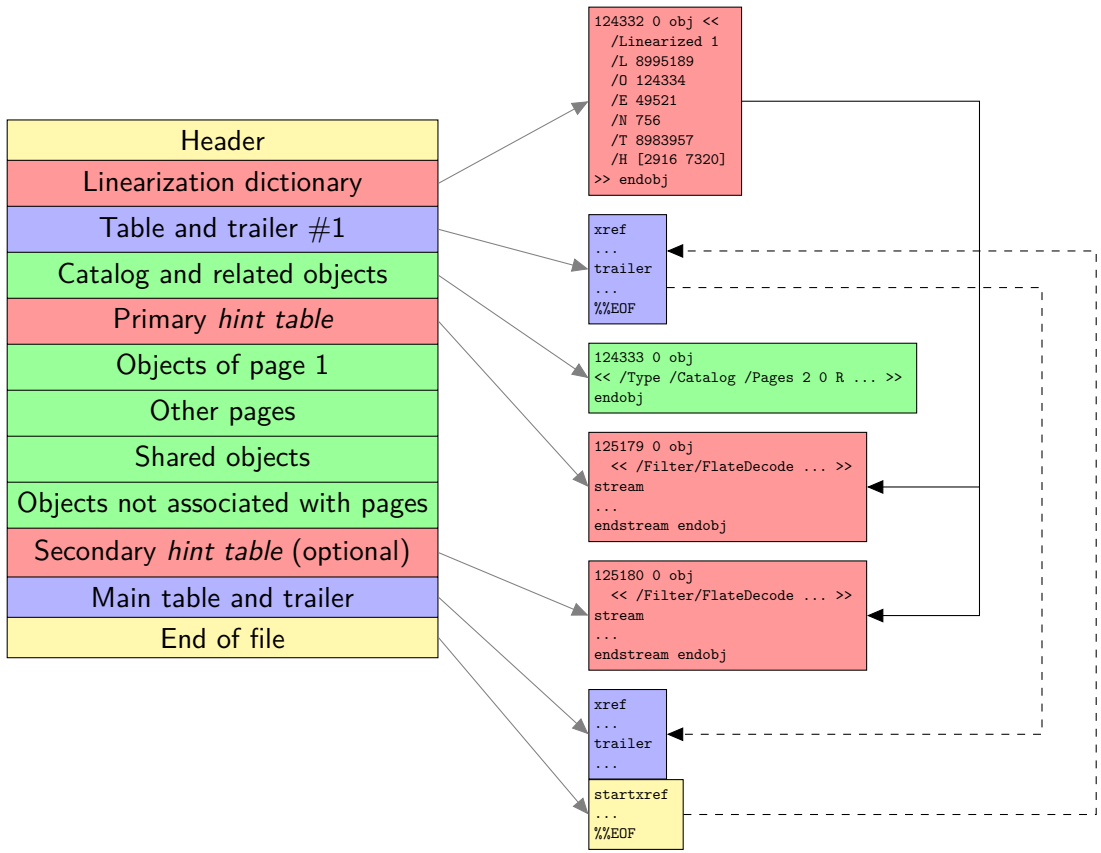
Comment afficher un fichier avant de l'avoir téléchargé complètement ?



Source : `https://gendignoux.com/blog/`

Et l'expérience utilisateur ?

Comment afficher un fichier avant de l'avoir téléchargé complètement ?



Source : `https://gendignoux.com/blog/`

- Quelle cohérence entre les deux manières de lire le fichier ?

Quid de la sécurité de PDF ?

It's not a bug, it's a feature

La richesse du format est une première cause de vulnérabilités

- JavaScript
- inclusion d'autres plugins *actifs*
  - vidéo
  - Flash
  - ...
- inclusion de ressources en tout genre
  - les vulnérabilités des *parsers* correspondants sont donc des vecteurs d'attaques

Des lecteurs passoires

Face à ce large périmètre d'attaque, les lecteurs classiques peinent à contenir les assauts

- plus de 500 vulnérabilités sur Adobe Reader entre 1999 et 2016
- un temps de réponse pour la correction parfois de l'ordre de l'année

Des lecteurs passoires

Face à ce large périmètre d'attaque, les lecteurs classiques peinent à contenir les assauts

- plus de 500 vulnérabilités sur Adobe Reader entre 1999 et 2016
- un temps de réponse pour la correction parfois de l'ordre de l'année

- plusieurs *bugs* remontés pendant le stage
- un des projets avait des dizaines de failles référencées depuis longtemps, sans espoir d'être corrigées rapidement

Fichiers polyglottes

Comment connaître le *type* d'un fichier ?

- en regardant son extension
- en faisant confiance à l'utilisateur
- en devinant à partir du contenu

Fichiers polyglottes

Comment connaître le *type* d'un fichier ?

- en regardant son extension
- en faisant confiance à l'utilisateur
- en devinant à partir du contenu

En pratique cette question n'a pas toujours une réponse unique

- un fichier PDF + ZIP peut être légitime pour les deux formats
- PoC||GTFO et travaux d'Ange Albertini

Fichiers polyglottes

Comment connaître le *type* d'un fichier ?

- en regardant son extension
- en faisant confiance à l'utilisateur
- en devinant à partir du contenu

En pratique cette question n'a pas toujours une réponse unique

- un fichier PDF + ZIP peut être légitime pour les deux formats
- PoC||GTFO et travaux d'Ange Albertini

Problèmes de sécurité ?

- confusion dangereuse dans un contexte web (*The Tangled Web* de Michal Zalewski)
- différence d'interprétations entre un anti-virus et un lecteur PDF

Autres idées

- La richesse du format permet de cacher facilement de l'information de manière discrète
- À l'inverse, peut-on garantir que le contenu d'une version précédente a bien été effacée ?
- Quelle interaction entre la variété d'interprétations possibles de certains fichiers et la signature électronique ?

Les PDF dans la vraie vie

Le marqueur de début de fichier

Un fichier PDF doit commencer par une ligne `%PDF-1.5` (où 1.5 est la version du format)

Le marqueur de début de fichier

Un fichier PDF doit commencer par une ligne `%PDF-1.5` (où 1.5 est la version du format)

- sauf que la version peut être supprimée sans risque

Le marqueur de début de fichier

Un fichier PDF doit commencer par une ligne `%PDF-1.5` (où 1.5 est la version du format)

- sauf que la version peut être supprimée sans risque
- sauf que certains lecteurs acceptent que le marqueur soit placé plus loin

Le marqueur de début de fichier

Un fichier PDF doit commencer par une ligne `%PDF-1.5` (où 1.5 est la version du format)

- sauf que la version peut être supprimée sans risque
- sauf que certains lecteurs acceptent que le marqueur soit placé plus loin
- sauf que d'autres acceptent qu'il soit absent

Le marqueur de début de fichier

Un fichier PDF doit commencer par une ligne `%PDF-1.5` (où 1.5 est la version du format)

- sauf que la version peut être supprimée sans risque
- sauf que certains lecteurs acceptent que le marqueur soit placé plus loin
- sauf que d'autres acceptent qu'il soit absent

```
Syntax Warning: May not be a PDF file (continuing anyway)
```

Be liberal in what you accept... and accept the consequences

De manière générale, la norme PDF invite les développeurs à être tolérant (reconstruction de fichiers corrompus, laxisme face à des erreurs de syntaxe)

> *When a conforming reader reads a PDF file with a damaged or missing cross-reference table, it **may attempt** to rebuild the table by scanning all the objects in the file.*
>
> — *ISO 32000-1-2008, annex C.2*

Be liberal in what you accept... and accept the consequences

De manière générale, la norme PDF invite les développeurs à être tolérant (reconstruction de fichiers corrompus, laxisme face à des erreurs de syntaxe)

> *When a conforming reader reads a PDF file with a damaged or missing cross-reference table, it **may attempt** to rebuild the table by scanning all the objects in the file.*
>
> — *ISO 32000-1-2008, annex C.2*

En général, cette approche est une hérésie en sécurité

Be liberal in what you accept... and accept the consequences

De manière générale, la norme PDF invite les développeurs à être tolérant (reconstruction de fichiers corrompus, laxisme face à des erreurs de syntaxe)

> *When a conforming reader reads a PDF file with a damaged or missing cross-reference table, it **may attempt** to rebuild the table by scanning all the objects in the file.*
>
> — *ISO 32000-1-2008, annex C.2*

En général, cette approche est une hérésie en sécurité

- vieil exemple : laxisme sur le format des `crontab`

Be liberal in what you accept... and accept the consequences

De manière générale, la norme PDF invite les développeurs à être tolérant (reconstruction de fichiers corrompus, laxisme face à des erreurs de syntaxe)

> *When a conforming reader reads a PDF file with a damaged or missing cross-reference table, it **may attempt** to rebuild the table by scanning all the objects in the file.*
>
> — *ISO 32000-1-2008, annex C.2*

En général, cette approche est une hérésie en sécurité

- vieil exemple : laxisme sur le format des `crontab`
- différence d'interprétation d'un lecteur à l'autre

Be liberal in what you accept... and accept the consequences

De manière générale, la norme PDF invite les développeurs à être tolérant (reconstruction de fichiers corrompus, laxisme face à des erreurs de syntaxe)

> *When a conforming reader reads a PDF file with a damaged or missing cross-reference table, it **may attempt** to rebuild the table by scanning all the objects in the file.*
>
> — *ISO 32000-1-2008, annex C.2*

En général, cette approche est une hérésie en sécurité

- vieil exemple : laxisme sur le format des `crontab`
- différence d'interprétation d'un lecteur à l'autre
- ou entre un lecteur et un anti-virus

Be liberal in what you accept... and accept the consequences

De manière générale, la norme PDF invite les développeurs à être tolérant (reconstruction de fichiers corrompus, laxisme face à des erreurs de syntaxe)

> *When a conforming reader reads a PDF file with a damaged or missing cross-reference table, it **may attempt** to rebuild the table by scanning all the objects in the file.*
>
> — *ISO 32000-1-2008, annex C.2*

En général, cette approche est une hérésie en sécurité

- vieil exemple : laxisme sur le format des `crontab`
- différence d'interprétation d'un lecteur à l'autre
- ou entre un lecteur et un anti-virus
- que vaut un PDF corrompu signé ?

Le principe de robustesse est une ânerie

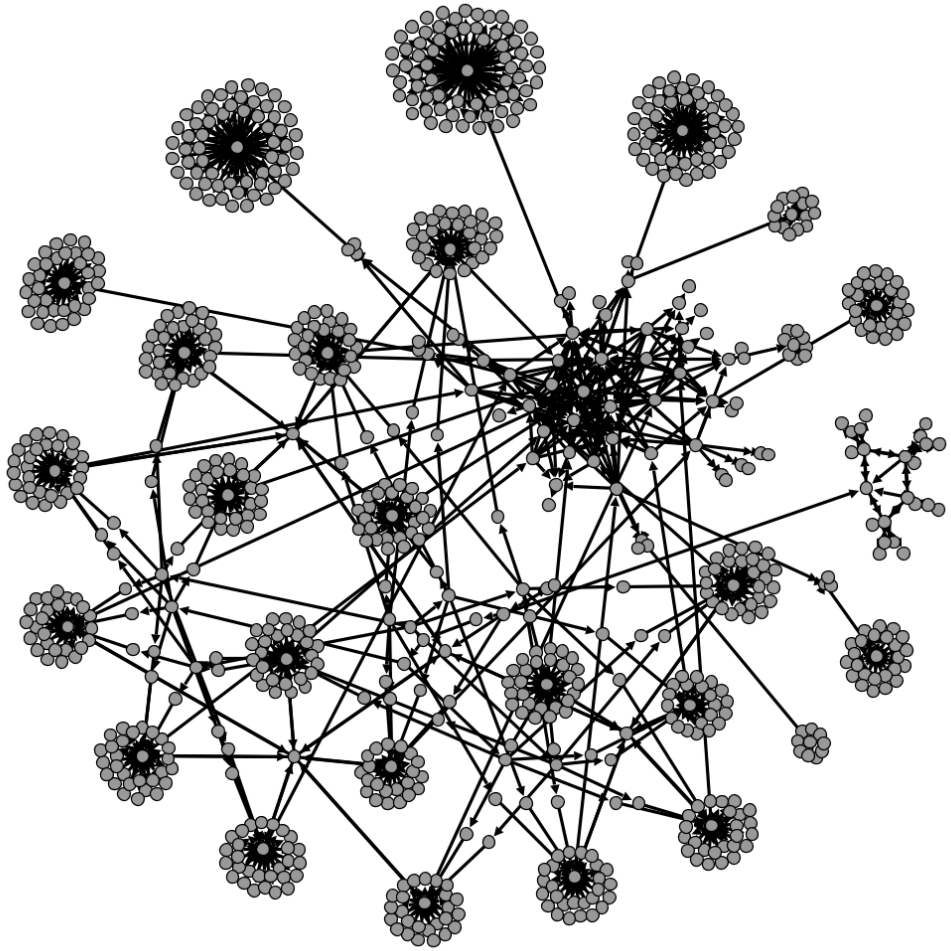
La *loi de Postel*, aussi connu sous le nom de *principe de robustesse*, est un dogme cher aux développeurs de protocoles réseau

Be strict in what you send, be liberal in what you accept

Ce principe favorise les déviations du standard et la profusion d'implémentations de mauvaise qualité

Le véritable bon conseil est simplement *Be strict*

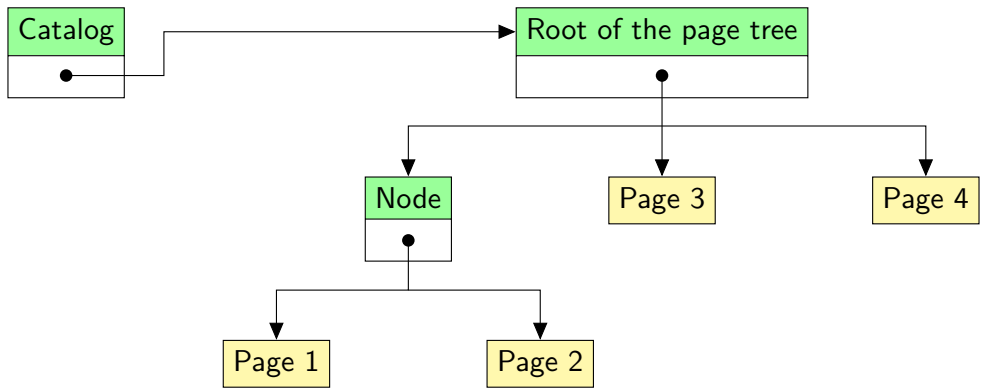
Structure logique d'un fichier PDF



Exemple d'un document de 17 pages (environ 1000 objets)

Organisation de ce graphe (1/2)

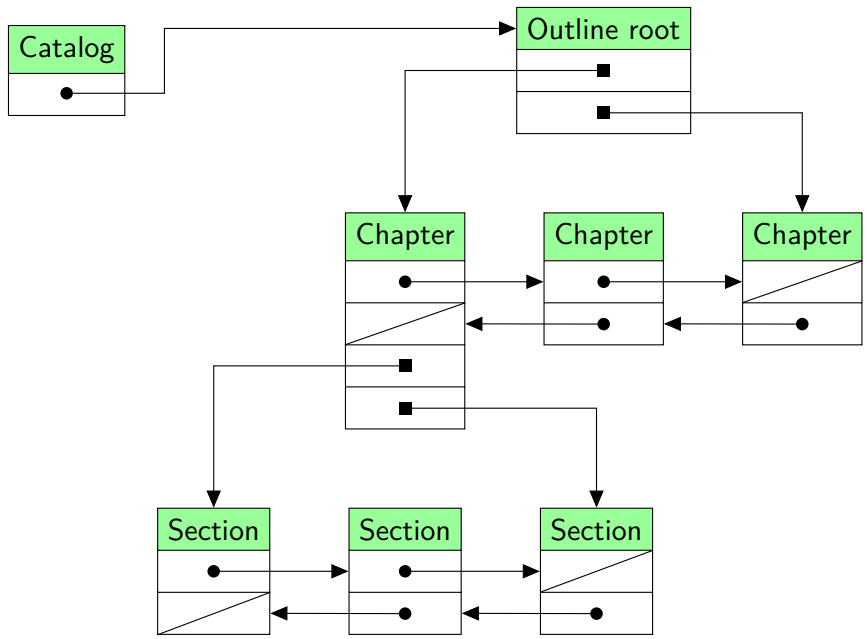
Le graphe des objets est organisé en sous-structures, dont des arbres, par exemple l'arbre des pages



Source : `https://gendignoux.com/blog/`

Organisation de ce graphe (2/2)

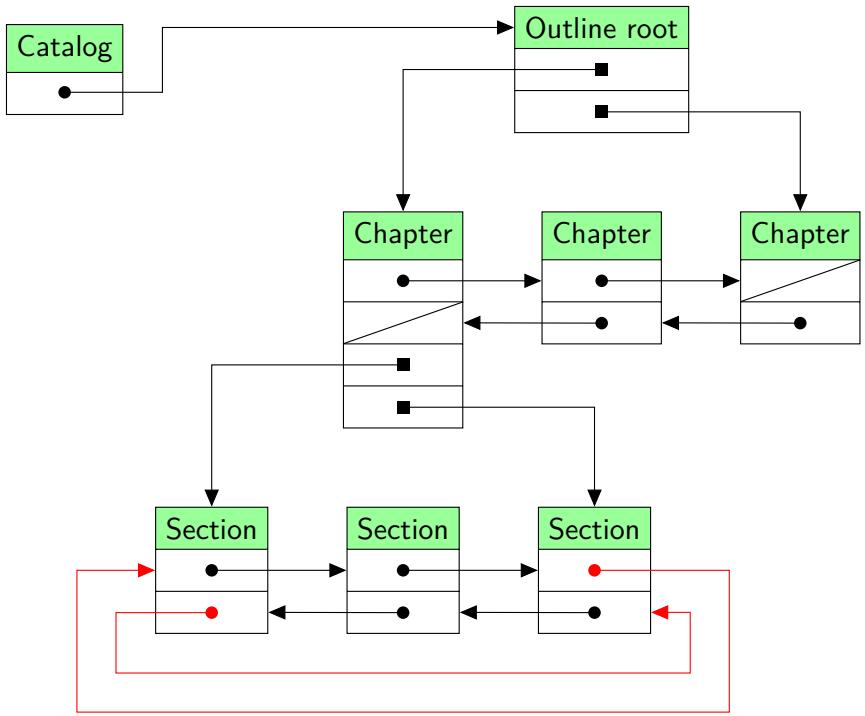
La table des matières est une liste doublement chaînée



Source : `https://gendignoux.com/blog/`

Format de la structure trop expressive ?

En réalité on peut décrire n'importe quel graphe dans le format



Source : `https://gendignoux.com/blog/`

Démonstrations

Cycle dans la table des matières

`https://github.com/ANSSI-FR/caradoc/blob/master/test_files/negative/outlines/cycle.pdf`

Fichier polymorphe

`https://github.com/ANSSI-FR/caradoc/blob/master/test_files/negative/polymorph/polymorph.pdf`

Un autre cycle problématique

Les tables de références croisées

- pour chaîner les tables, chaque table pointe vers la précédente

Un autre cycle problématique

Les tables de références croisées

- pour chaîner les tables, chaque table pointe vers la précédente
- que se passe-t-il si on crée un cycle ?

Un autre cycle problématique

Les tables de références croisées

- pour chaîner les tables, chaque table pointe vers la précédente
- que se passe-t-il si on crée un cycle ?

- des chercheurs ont tenté d'écrire un *parser* PDF en Coq
  - ils ont ainsi découvert ce problème de cycle (LangSec@SSP 2014)
  - *crash* de certains lecteurs
  - leur conclusion est là encore que la spécification est mauvaise

Solutions existantes

Une solution extrême

PDF est perdu pour la cause

- position de Invisible Things Lab (Qubes)
- l'affichage de confiance passe par la transformation en bitmaps

Une solution extrême

PDF est perdu pour la cause

- position de Invisible Things Lab (Qubes)
- l'affichage de confiance passe par la transformation en bitmaps

- le format cible est complètement passif
- la transformation peut être faite dans un conteneur jetable non de confiance

Une solution extrême

PDF est perdu pour la cause

- position de Invisible Things Lab (Qubes)
- l'affichage de confiance passe par la transformation en bitmaps

- le format cible est complètement passif
- la transformation peut être faite dans un conteneur jetable non de confiance

- le coût en stockage/transmission est potentiellement important
- les fonctionnalités de recherches de texte sont perdues

Une solution extrême

PDF est perdu pour la cause

- position de Invisible Things Lab (Qubes)
- l'affichage de confiance passe par la transformation en bitmaps

- le format cible est complètement passif
- la transformation peut être faite dans un conteneur jetable non de confiance

- le coût en stockage/transmission est potentiellement important
- les fonctionnalités de recherches de texte sont perdues
- c'est un peu de la triche

La liste noire

Filtrage des scripts et des fonctionnalités dangereuses

- approches IDS ou IPS possibles

La liste noire

Filtrage des scripts et des fonctionnalités dangereuses

- approches IDS ou IPS possibles
- a priori efficace contre les menaces connues

La liste noire

Filtrage des scripts et des fonctionnalités dangereuses

- approches IDS ou IPS possibles
- a priori efficace contre les menaces connues
- certains documents *légitimes* nécessitent ces fonctionnalités...
- quid des différences d'interprétations entre *parsers*
  - on a besoin d'outils robustes et fiables pour interpréter *la structure*

L'analyse *comportementale*

Techniques de *machine learning* pour classifier les fichiers à partir de l'analyse des caractéristiques de fichiers PDF

- exemple : PDFrate (ACSAC 2012)

L'analyse *comportementale*

Techniques de *machine learning* pour classifier les fichiers à partir de l'analyse des caractéristiques de fichiers PDF

- exemple : PDFrate (ACSAC 2012)

- jolis papiers académiques
- approche potentiellement efficace pour des menaces pas encore connues

L'analyse *comportementale*

Techniques de *machine learning* pour classifier les fichiers à partir de l'analyse des caractéristiques de fichiers PDF

- exemple : PDFrate (ACSAC 2012)

- jolis papiers académiques
- approche potentiellement efficace pour des menaces pas encore connues

- il faut toujours regarder les taux de faux positifs / faux négatifs
- les outils sont souvent très sensibles au jeu de données d'apprentissage ?
- en pratique des chercheurs ont réussi à tromper PDFrate (SSP 2014) car il repose sur un *parser* simpliste
- quid des différences d'interprétations entre *parsers*
  - on a besoin d'outils robustes et fiables pour interpréter *la structure*

Caradoc

Présentation de notre approche

Le *parsing*, qui transforme un fichier en une structure complexe, est crucial

Comment améliorer cette étape ?

- supprimer certaines fonctionnalités intrinsèquement trop complexes
- implémenter un *parser* strict
  - dire non à Postel
  - rejeter les constructions ambiguës
- vérifier les types des différents éléments
- analyser les graphes pour rejeter les cycles et autres incohérences

Présentation de notre approche

Le *parsing*, qui transforme un fichier en une structure complexe, est crucial

Comment améliorer cette étape ?

- supprimer certaines fonctionnalités intrinsèquement trop complexes
- implémenter un *parser* strict
  - dire non à Postel
  - rejeter les constructions ambiguës
- vérifier les types des différents éléments
- analyser les graphes pour rejeter les cycles et autres incohérences

- jolis papiers académiques
- les fichiers ainsi validés sont très propres
- ils ont peu de risque d'être mal interprétés

Présentation de notre approche

Le *parsing*, qui transforme un fichier en une structure complexe, est crucial

Comment améliorer cette étape ?

- supprimer certaines fonctionnalités intrinsèquement trop complexes
- implémenter un *parser* strict
  - dire non à Postel
  - rejeter les constructions ambiguës
- vérifier les types des différents éléments
- analyser les graphes pour rejeter les cycles et autres incohérences

- jolis papiers académiques
- les fichiers ainsi validés sont très propres
- ils ont peu de risque d'être mal interprétés

- peu de fichiers passent ce filtre
- les garanties apportées ne dépassent pas la structure

Normalisation de fichiers PDF

En autorisant une phase de normalisation, on peut valider plus de fichiers

- réécriture du fichier pour se passer des fonctionnalités rejetées
- nettoyage des objets lorsqu'il existe une interprétation non ambiguë
- correction de *bugs* connus (typos dans certains champs)

Normalisation de fichiers PDF

En autorisant une phase de normalisation, on peut valider plus de fichiers

- réécriture du fichier pour se passer des fonctionnalités rejetées
- nettoyage des objets lorsqu'il existe une interprétation non ambiguë
- correction de *bugs* connus (typos dans certains champs)

- utilisation plus réaliste

Normalisation de fichiers PDF

En autorisant une phase de normalisation, on peut valider plus de fichiers

- réécriture du fichier pour se passer des fonctionnalités rejetées
- nettoyage des objets lorsqu'il existe une interprétation non ambiguë
- correction de *bugs* connus (typos dans certains champs)

- utilisation plus réaliste

- c'est un peu de la triche
- certaines réécritures sont discutables

Résultats de l'étude

Bugs remontés pendant le stage

- *crash* de certains lecteurs sur des cycles
- consommation CPU à 100 % sur certains cycles
- techniques d'évasion en jouant sur des différences d'interprétation

Propositions pour une meilleure spécification

Vers un format *PDF/Restricted* ?

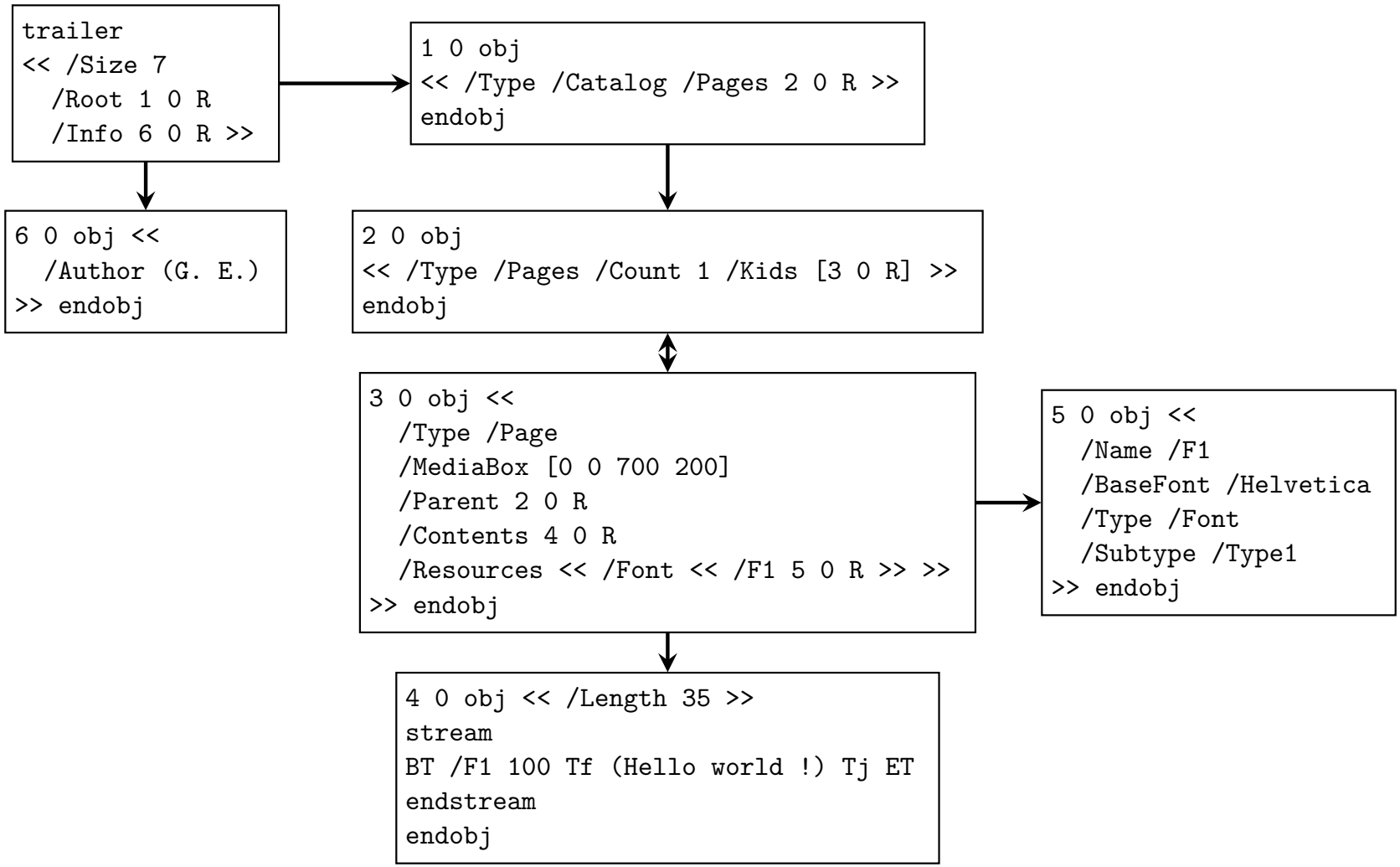
- grammaire BNF décrivant le format
- levée des ambiguïtés sur les types et la structure
- contrainte sur les graphes pour interdire les cycles

Implémentation

Caradoc

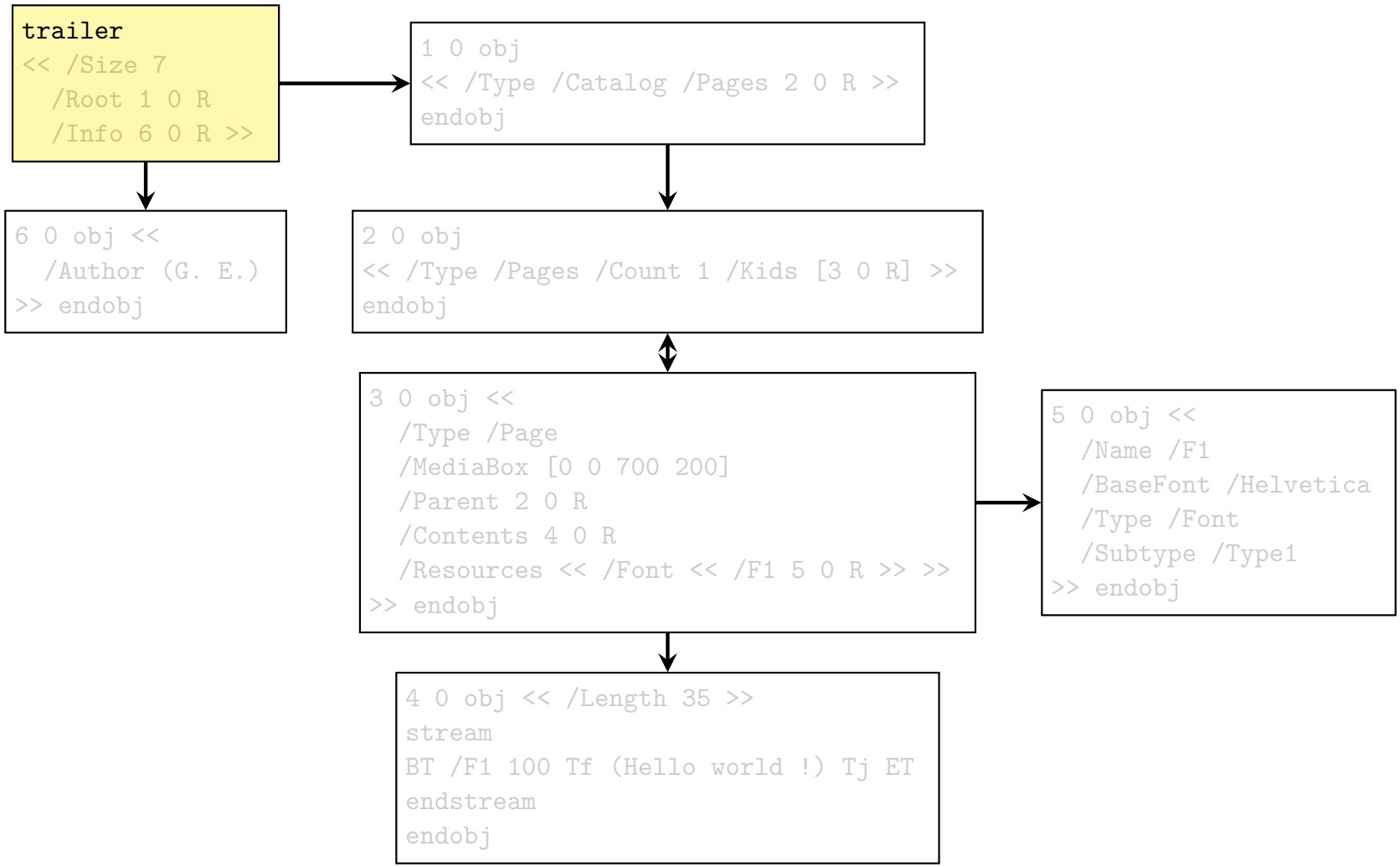
- `https://github.com/ANSSI-FR/caradoc`
- mise en œuvre de la proposition
- extraction d'informations sur un fichier PDF
- normalisation / nettoyage de fichiers
- production de jolis schémas

Algorithme de *Type checking*



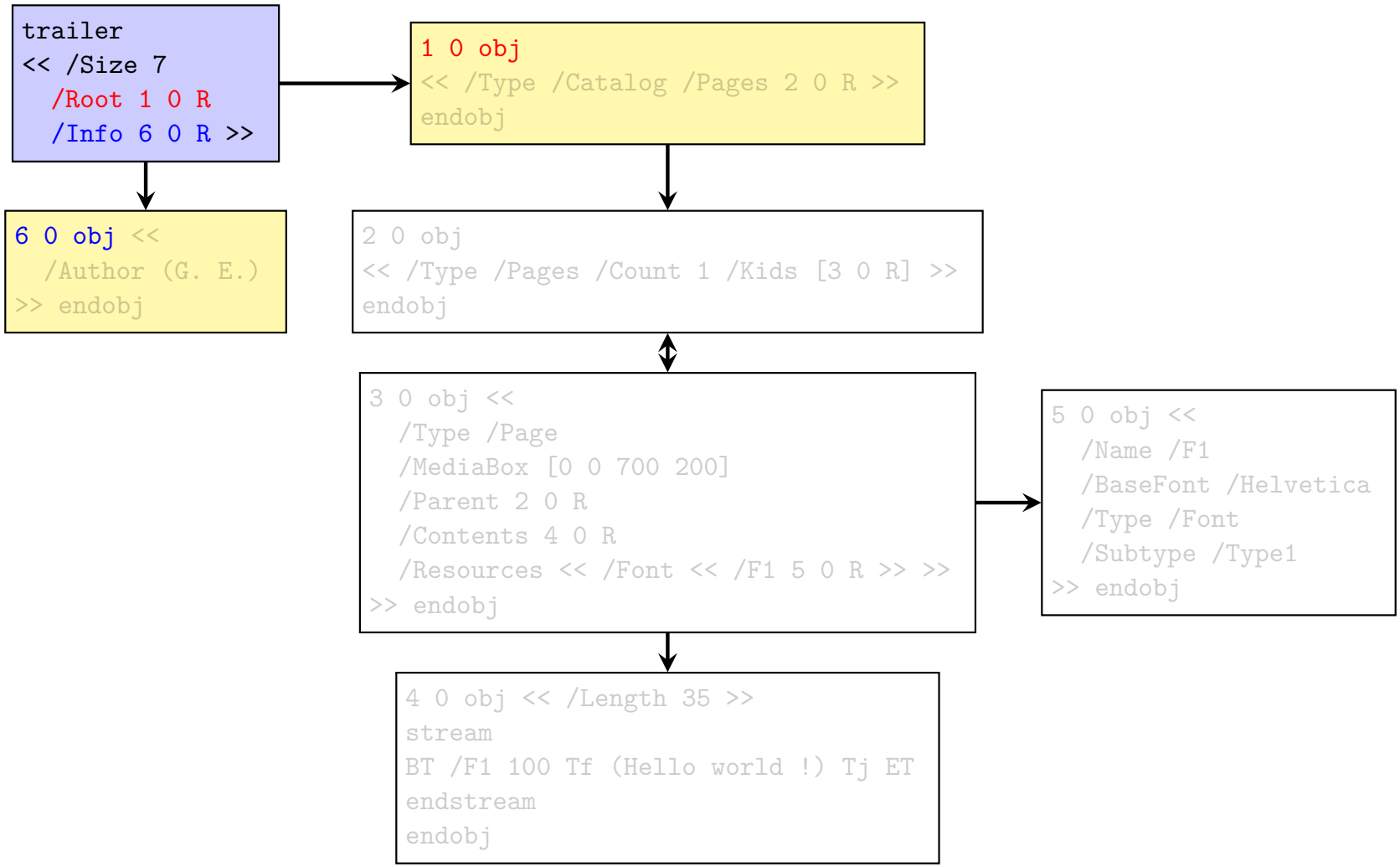
Exemple sur un fichier *Hello World*

Algorithme de *Type checking*



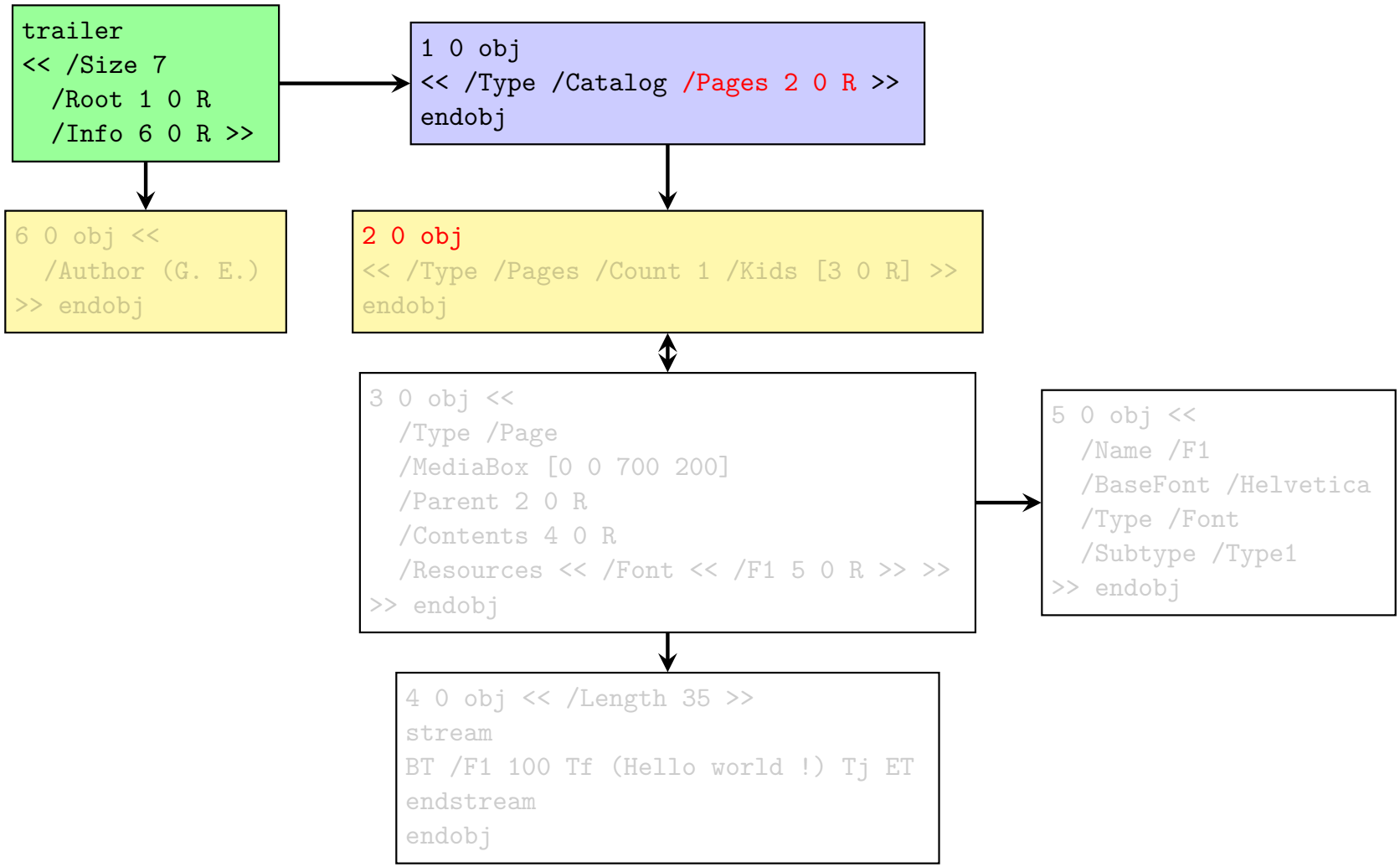
Propagation des contraintes

Algorithme de *Type checking*



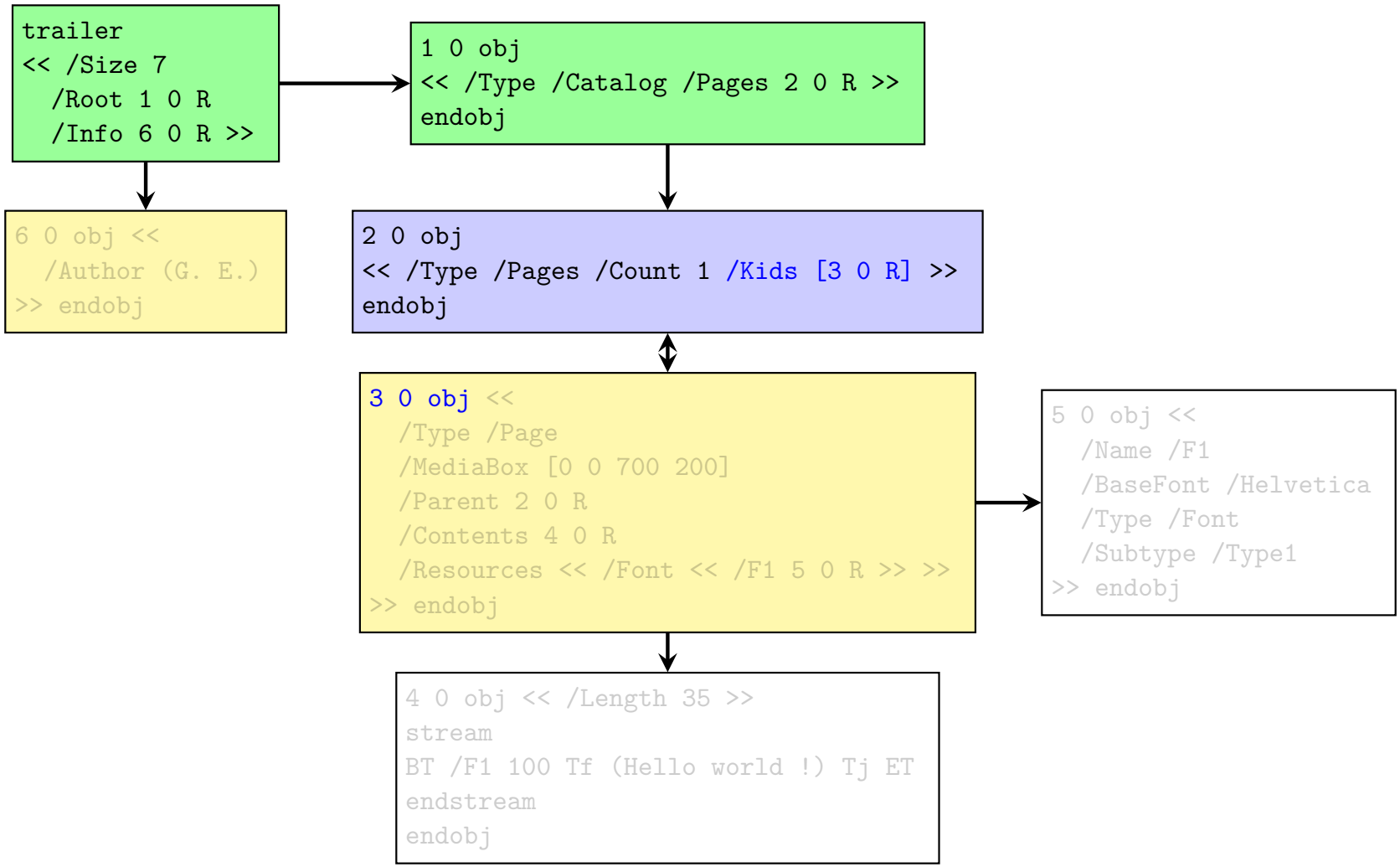
Propagation des contraintes

Algorithme de *Type checking*



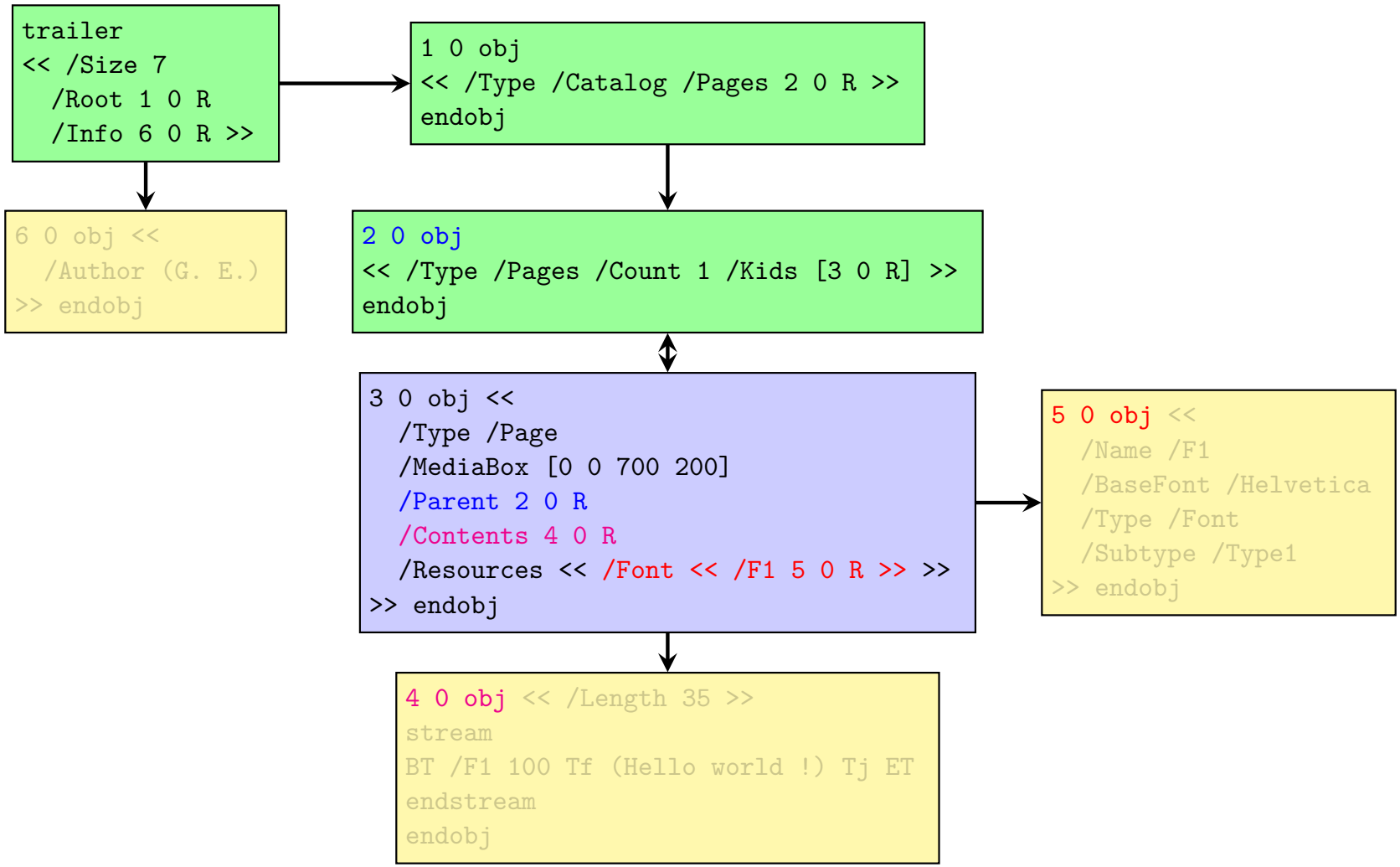
Propagation des contraintes

Algorithme de *Type checking*



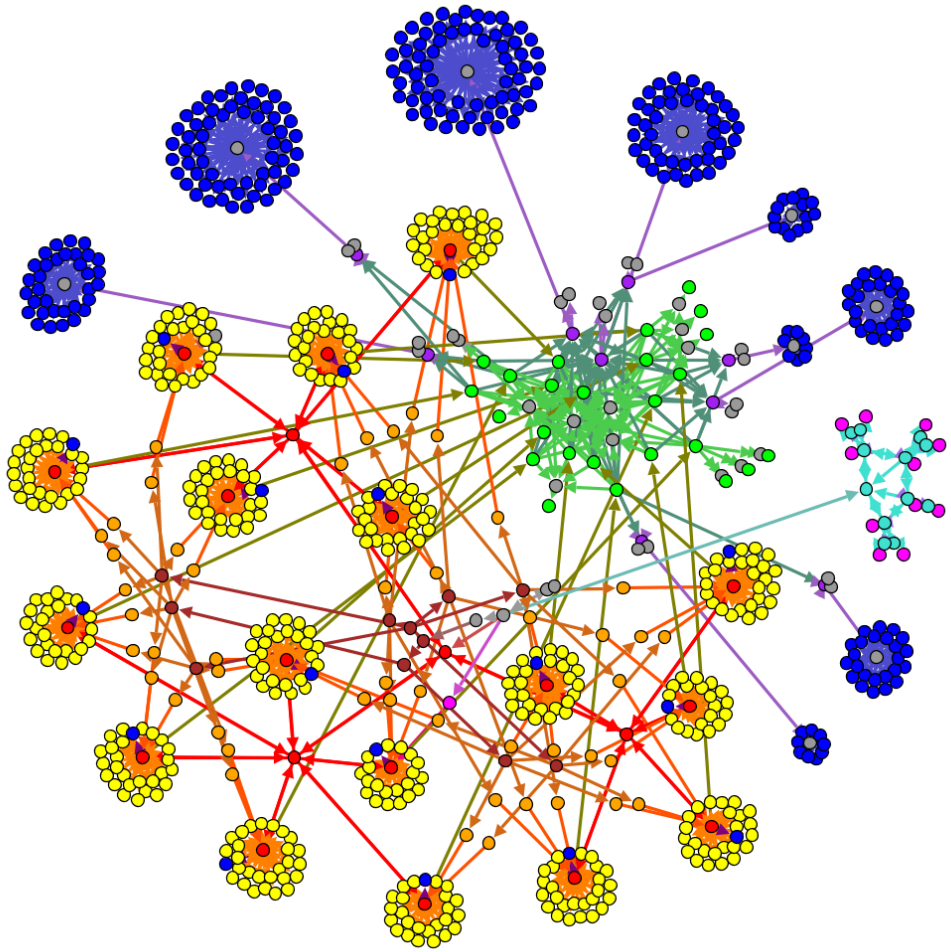
Propagation des contraintes

Algorithme de *Type checking*



Propagation des contraintes

Structure logique d'un fichier PDF (v2)



Le document précédent, avec ses types

Expérimentations sur des fichiers trouvés dans la nature

Problème de validations rencontrés

- des types inconnus
- des objets non référencés
- l'utilisation du chiffrement
- des incohérences dans les listes chaînées
- des typos dans les champs
  - `/Blackls1` à la place de `/BlackIs1`
  - `/XObjcect` à la place de `/XObject`

Au-delà de la structure

État des travaux récents sur caradoc

- interprétation du contenu graphique
- prise en compte du chiffrement
  - en particulier la clé vide par défaut...

Perspectives

- Plus d'objets
- Plus de filtres de compressions
- Support des polices et des images

- Inclusion d'analyses de plus haut niveau (JS)
- Intégration dans des outils de classification

- Industrialisation de l'outil
- Nouvelles expérimentations sur des jeux de données

- Propositions d'évolution du standard

Conclusion

Conclusion

- PDF est un format riche et complexe
- La spécification et l'écosystème laissent à désirer
- Les travaux menés ont permis
  - d'identifier des problèmes dans la spécification et les implémentations
  - de proposer des contraintes sur le format
  - d'implémenter un outil vérifiant ces contraintes
- Il reste des choses intéressantes à faire sur le sujet

Questions ?

Pour aller plus loin

- le projet Caradoc : `https://github.com/ANSSI-FR/caradoc`
- nos publications : `http://spw16.langsec.org/papers.html` et `https://www.sstic.org/2017/presentation/caradoc/`
- quelques articles de blog de Guillaume : `https://gendignoux.com/blog/tags.html#pdf`

Remerciements : Guillaume Endignoux et Jean-Yves Migeon