تحليل المستندات الضارة -- الجزء 02 -- مستندات بي دي إف

بعد إلقاء نظرة عامة على إدارة الأجهزة الظاهرية وتثبيت ريمنوكس ليكون بيئة لتحليل الآثار المشبوهة، سنستكشف ملفات بي دي إف (PDF) من حيث صيغتها والطرق التي يمكن استخدامها فيها لإلحاق الضرر بالمستخدمين.

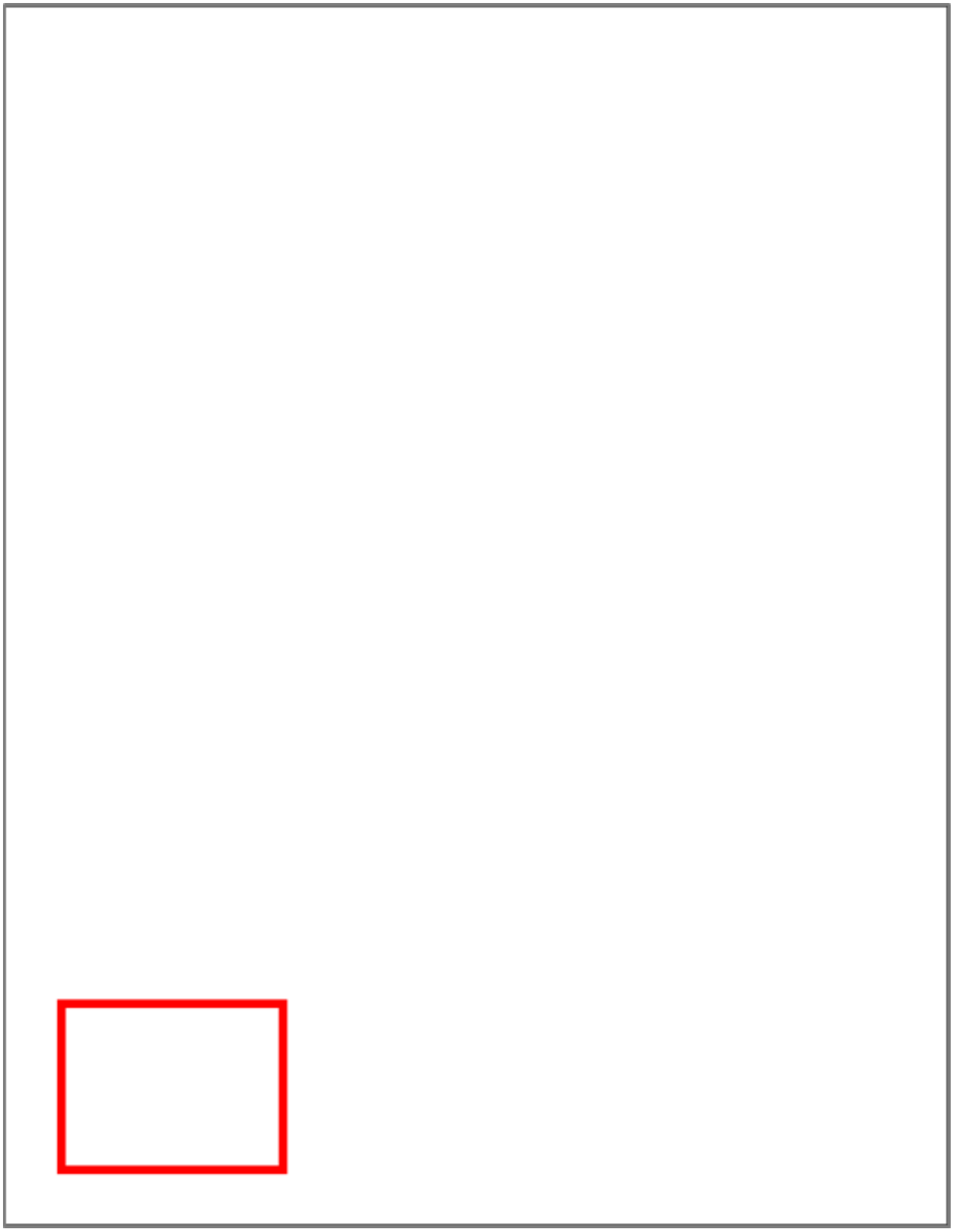
ما هو ملف بي دي إف؟

عندما نفتح ملف بي دي إف، نستخدم برنامجًا محددًا يعرض محتواه بطريقة قابلة للقراءة، ومع ذلك، كما هو الحال مع العديد من أنواع الملفات الأخرى، يتم إنشاء ملفات بي دي إف باستخدام مزيج من النص العادي والبيانات الثنائية (للصور والعناصر الأخرى التي قد تتطلب ذلك)، على سبيل المثال، يعرض النص التالي ملف بي دي إف الذي يظهر بعد:

Unset

```
%PDF-1.4
1 0 obj
<<
      /Length 51
>>
stream
1 0 0 RG
5 w
36 144 m
180 144 l
180 36 l
36 36 l
s
endstream
endobj
2 0 obj
<<

      /Type /Catalog
      /Pages 3 0 R
>>
endobj
3 0 obj
<<
      /Type /Pages
      /Kids [4 0 R ]
      /Count 1
```

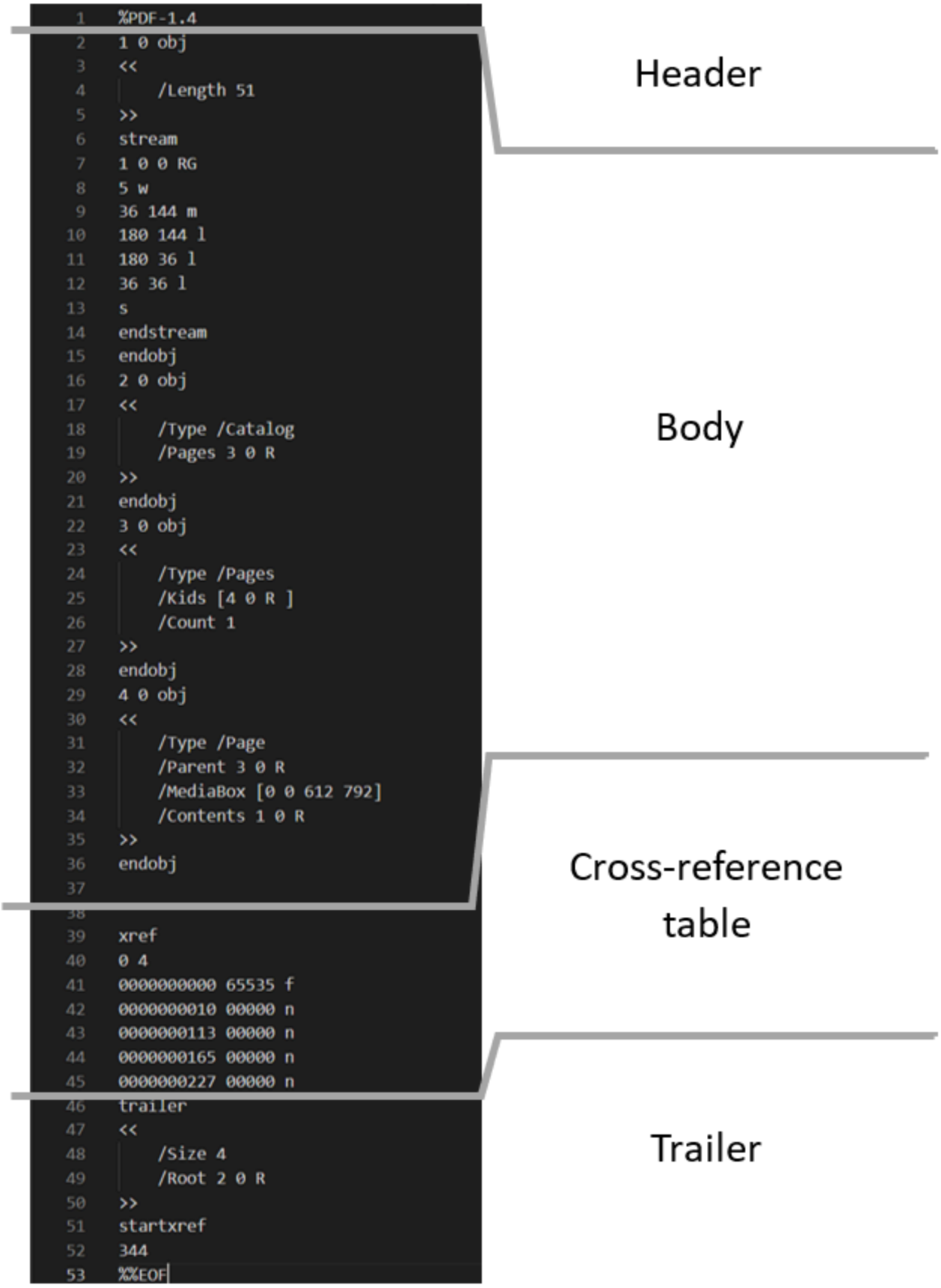
```
>>
endobj
4 0 obj
<<

      /Type /Page
      /Parent 3 0 R
      /MediaBox [0 0 612 792]
      /Contents 1 0 R
>>
endobj


xref
0 4
0000000000 65535 f
0000000010 00000 n
0000000113 00000 n
0000000165 00000 n
0000000227 00000 n
trailer
<<
      /Size 4
      /Root 2 0 R
>>
startxref
344
%%EOF
```



من "كيفية إنشاء ملف بي دي إف بسيط" من برنامج كالاس (Callas)

بنية الملف

من هذا المثال يمكننا رؤية البنية القياسية لأي ملف بي دي إف:


%PDF-1.4
1 0 obj
<<
/Length 51
>>
stream
1 0 0 RG
5 w
36 144 m
180 144 l
180 36 l
36 36 l
s
endstream
endobj
2 0 obj
<<
/Type /Catalog
/Pages 3 0 R
>>
endobj
3 0 obj
<<
/Type /Pages
/Kids [4 0 R]
/Count 1
>>
endobj
4 0 obj
<<
/Type /Page
/Parent 3 0 R
/MediaBox [0 0 612 792]
/Contents 1 0 R
>>
endobj
xref
0 4
0000000000 65535 f
0000000010 00000 n
0000000113 00000 n
0000000165 00000 n
0000000227 00000 n
trailer
<<
/Size 4
/Root 2 0 R
>>
startxref
344
%%EOF
Header
Body
Cross-reference table
Trailer

الرأس: يحتوي على إصدار البروتوكول الذي تم إنشاء الملف به لإرشاد برنامج القارئ حول كيفية قراءة بقية البنية وتقديم جميع عناصره.

العرض: هنا توجد جميع الكائنات التي تشكل ملف بي دي إف والصفحات والصور والنصوص والخطوط وما إلى ذلك حتى التعليمات البرمجية والإجراءات الآلية إذا كان الملف يحتوي عليها.

جدول الإسناد الترافقي: سنجد هنا قائمة بجميع كائنات المستند لتسهيل الوصول إليها وأماكنها داخل الملف، وهي أشبه "بجدول المحتويات" ولكن مخصص لبرنامج قارئ بي دي إف. إذا احتاج القارئ في أي وقت من الأوقات إلى تقديم كائن معين (على سبيل المثال، إذا انتقلنا إلى صفحة عشوائية في مستند كبير)، فسوف يرى برنامج القارئ الصفحة التي يجب أن يقدمها ويبحث عنها على هذا الجدول لتحديد موقع العناصر المعنية في العرض لتحميلها على الشاشة.

المعلومات الملحقة: سنجد هنا المكان في مستند جدول الإسناد الترافقي ومعلومات مفيدة أخرى مثل عدد الكائنات في جدول الإسناد الترافقي (للتحقق من أن الملف غير تالف على سبيل المثال) والكائن الجذر للمستند ومعلومات التشفير إن وجدت. تقوم برامج قراءة ملفات بي دي إف بحسب تصميمها بقراءة المستندات من النهاية حيث يمكنها العثور بسرعة على مكان الكائن الجذر وجدول الإسناد الترافقي لبدء عرض المحتوى.

فهم كائنات بي دي إف

بالنظر إلى طريقة هياكل ملفات بي دي إف، نرى أن القسم الأكثر إثارة للتعمّق به هو العرض لأنه يمكننا العثور على كل ما يمكننا رؤيته والتفاعل معه (مثل النص والصفحات والصور والنماذج والتعليمات البرمجية وما إلى ذلك). تسمى هذه العناصر جميعها كائنات، وهناك مجموعة واسعة منها وفقًا للمواصفات، مثل ما ورد في المثال السابق، الكائن.

```
Unset
4 0 obj
<<
      /Type /Page
      /Parent 3 0 R
      /MediaBox [0 0 612 792]
      /Contents 1 0 R
>>
endobj
```

وهو عبارة عن صفحة تحمل المعرف 4 ولها عنصر أصلي يحمل المعرف 3، وتحتوي كعنصر تابع على العنصر الذي يحمل المعرف 1 (المستطيل الأحمر). عرض الطريقة الدقيقة لكتابة كل كائن خارج نطاق هذه المادة، لكن هناك بعض أنواع العناصر التي يمكن استخدامها لأغراض ضارة ونريد تغطيتها أكثر قليلًا.

أوبن آكشن (OpenAction): يُشير هذا الكائن إلى مجموعة من الإجراءات التي سيتم تنفيذها عند فتح ملف بي دي إف، ويمكن استخدامه لإطلاق موقع ويب لدعم المحتوى أو لأغراض التتبع وتنفيذ شفرة جافا سكريبت، وما إلى ذلك. كما يمكن استخدام هذا لخداع المستخدمين لتنفيذ إجراءات إضافية يمكن أن تكون ضارة، أو حسب بيئة الكمبيوتر، يمكن لهذا الكائن تنفيذ البرمجيات الضارة مباشرة.

إيه إيه (AA): يتضمن هذا الكائن أيضًا سلسلة من الإجراءات التي يتم تشغيلها في ظل ظروف مختلفة، مثل العرض المرئي لصفحة، وتحريك مؤشر الفأرة فوق كائنات محددة، وملء حقول النموذج، وما إلى ذلك، وتعد المخاطر المرتبطة مشابهة لكائن /أوبن آكشن ولكن مع العديد من سيناريوهات التشغيل.

جيه إس (JS) أو جافا سكريبت (Javascript): يشمل تعليمات جافا سكريبت ليتم تنفيذها بعد تشغيل الإجراء وقد يتضمن هذا التعليمات البرمجية وظائف حصرية لملفات بي دي إف.

لانش (Launch): يحاول تشغيل تطبيق خارجي على الجهاز بعد تشغيل إجراء ما قد يستخدم، على سبيل المثال، لفتح مستندات أخرى أو تنفيذ أوامر محددة والتي قد تكون ضارة.

إمبدد فايل (EmbeddedFile): يسمح بتضمين الملفات العشوائية من المستندات إلى الملفات التنفيذية داخل ملف بي دي إف، وهناك سوابق لملفات بي دي إف الحميدة التي تحتوي على ملفات ضارة أخرى مضمّنة، مثل الملفات التنفيذية البرمجيات الضارة أو مستندات مايكروسوفت أوفيس (Microsoft Office) مع تعليمات الماكرو الضارة.

أوبجستم (ObjStm): يشمل معلومات عشوائية ستتم معالجتها وفقًا للطريقة التي تسمى بها، والاستخدام الرئيسي لهذا الكائن هو تجميع العديد من الكائنات وضغطها مما يؤدي إلى ملف أصغر. لكن يمكن استخدام هذا أيضًا لضغط التعليمات البرمجية الضارة كوسيلة للتعتيم عليها وتجنب اكتشاف برامج مكافحة الفيروسات. بالنظر إلى العديد من حالات الاستخدام المختلفة لهذا النوع من الكائنات فإن افتراض وجوده على أنه ضار سيؤدي بنا إلى العديد من الإيجابيات الخاطئة.

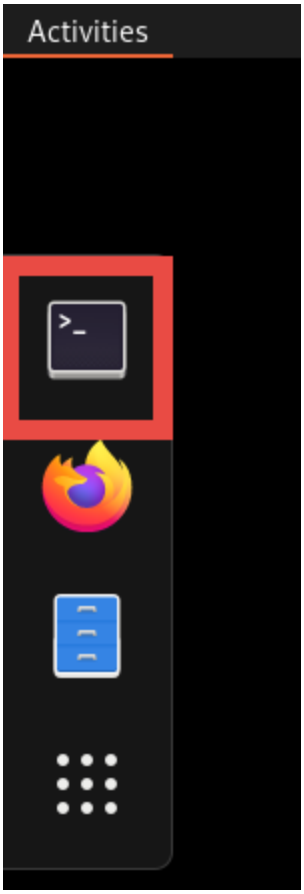
في الهجمات الأكثر تفصيلًا يمكن استخدام مزيج من هذه الكائنات، على سبيل المثال قد يؤدي ملف بي دي إف إلى إجراء يفتح ملفًا مضمنًا في نفس الملف المشفر في/أوبجستم.

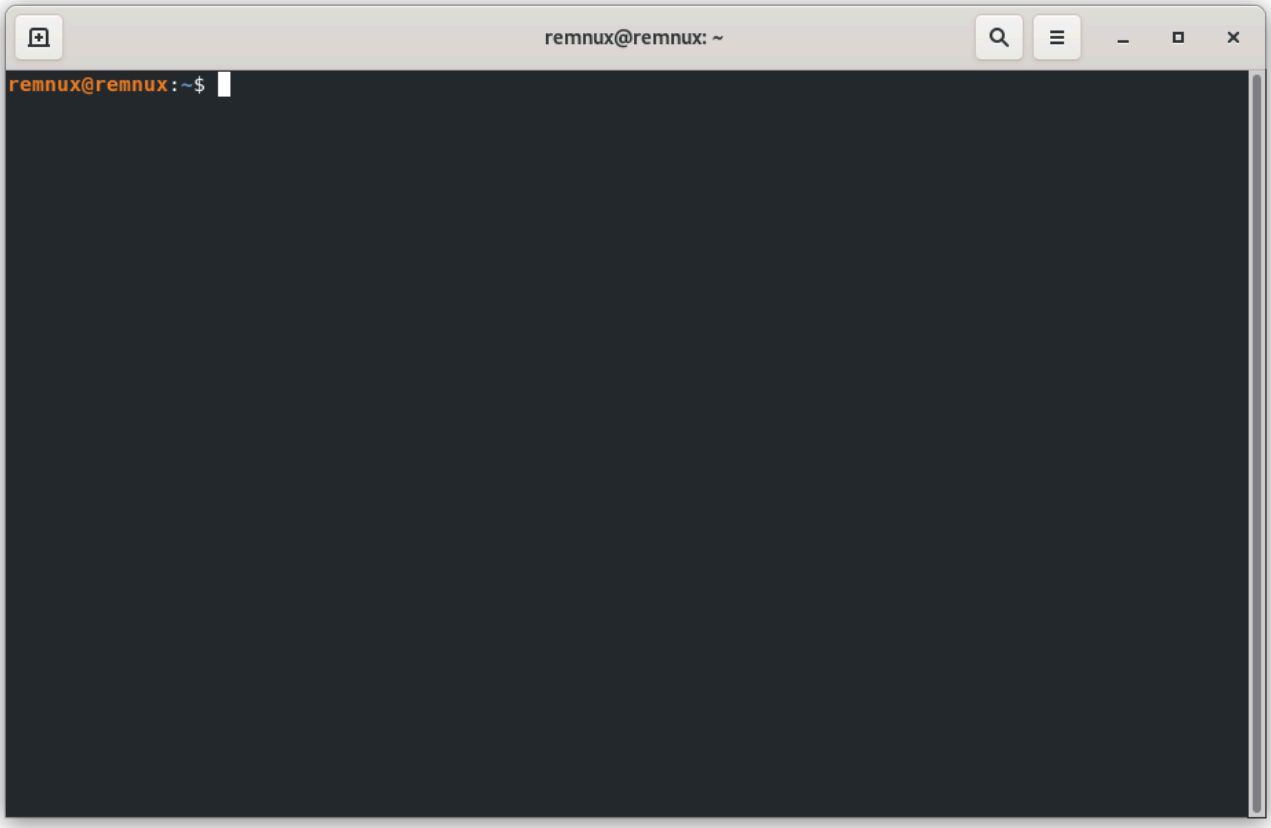
بالنظر إلى كل هذا، نريد أن نعرف ما إذا كان الملف يحتوي على أي من أنواع الكائنات هذه كخطوة أولى لمعرفة ما إذا كان ملف بي دي إف ضارًا أو على الأقل للتأكد من أنه ليس ضارًا.

تعريف بأداة بي دي إف آي دي (Pdfid)

نظرًا لأننا نعرف من أين نبدأ البحث عن المؤشرات التي تدق ناقوس الخطر على ملفات بي دي إف التي قد نعتبرها مشبوهة، يمكننا البدء في استخدام أداة بي دي إف آي دي كخطوة أولى لمعرفة أنواع الكائنات الموجودة في ملفنا. تُعدّ أداة بي دي إف آي دي جزءًا من مجموعة من الأدوات التي طورها ديدييه ستيفنز (Didier Stevens) لتبسيط بعض عمليات التحليل على ملفات بي دي إف. تُنفذ هذه الأدوات باستخدام واجهة سطر الأوامر ولذلك تُعرف باسم تطبيقات واجهة سطر الأوامر. سنشرح كيفية استخدامها باستخدام الأجهزة الظاهرية ريمنوكس الذي أعددناه في الجزء السابق من هذه الدورة.

لاستخدام بي دي إف آي دي سنحتاج إلى فتح تطبيق نافذة موجه الأوامر في الجهاز الظاهري، وعندما نبدأ بتشغيل الجهاز الافتراضي يجب أن تكون هذه النافذة مفتوحة بالفعل، ولكن يمكننا دائمًا النقر على قائمة الأنشطة في الزاوية العلوية اليسرى، ومن ثم أيقونة نافذة موجه الأوامر في اللوحة اليسرى، كم هو موضح في الصورة.


Activities

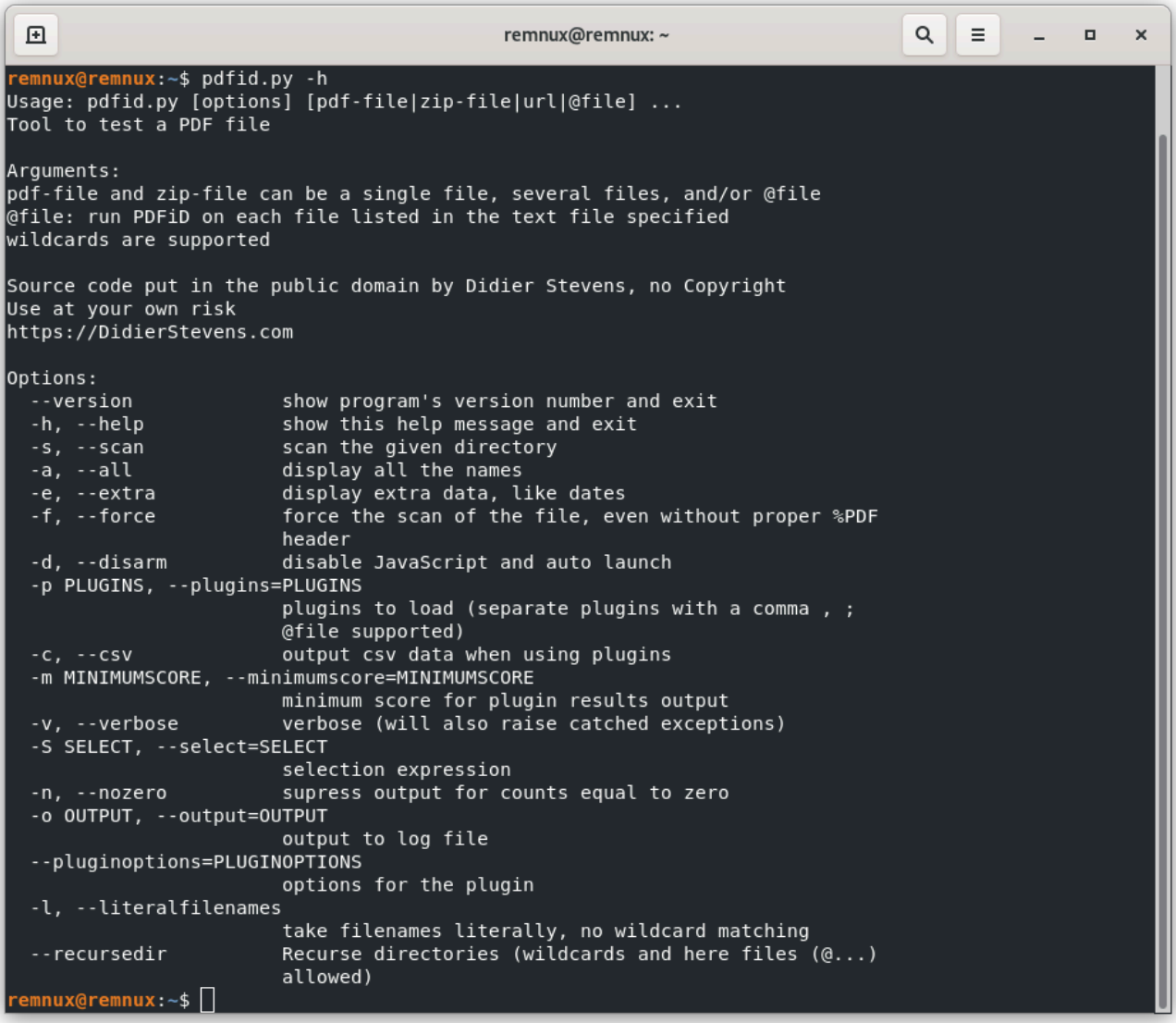


بمجرد الدخول إلى نافذة موجه الأوامر، يمكننا البدء في استكشاف استخدام أمر بي دي إف آي دي ولأجل الحصول على مساعدته نحتاج فقط إلى كتابة:

```
pdfid.py -h
```

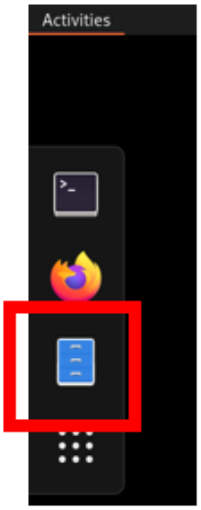
`-h` is for "**h**elp"

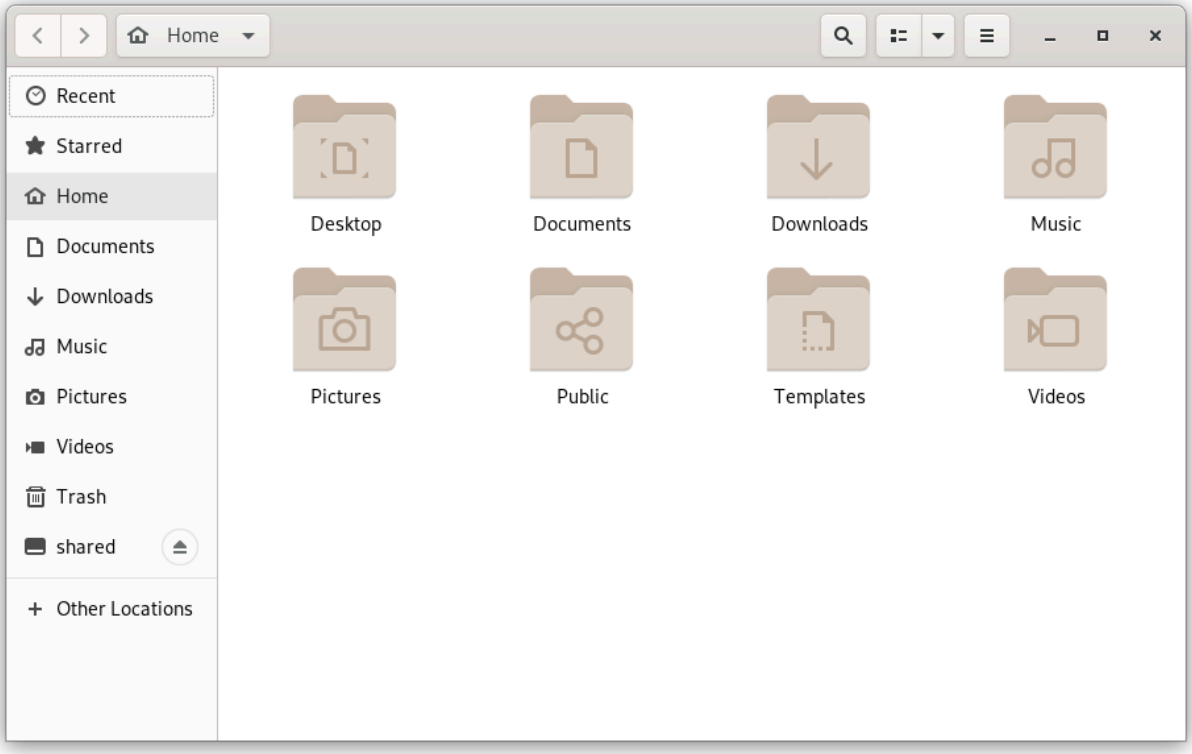
نصيحة: يمكننا دائمًا استخدام مفتاح Tab لإكمال بعض الأوامر تلقائيًا.



يمكننا أن نرى هنا عددًا من الخيارات التي يمكننا استخدامها عند استعمال بي دي إف آي دي وقد يمكن أن يكون هذا تحديًا للجدد على استخدام نافذة موجه الأوامر ولكننا عادة ما نلتزم بعدد من هذه الخيارات ومع القليل من الممارسة تصبح العملية أيضًا أسرع وأسهل.

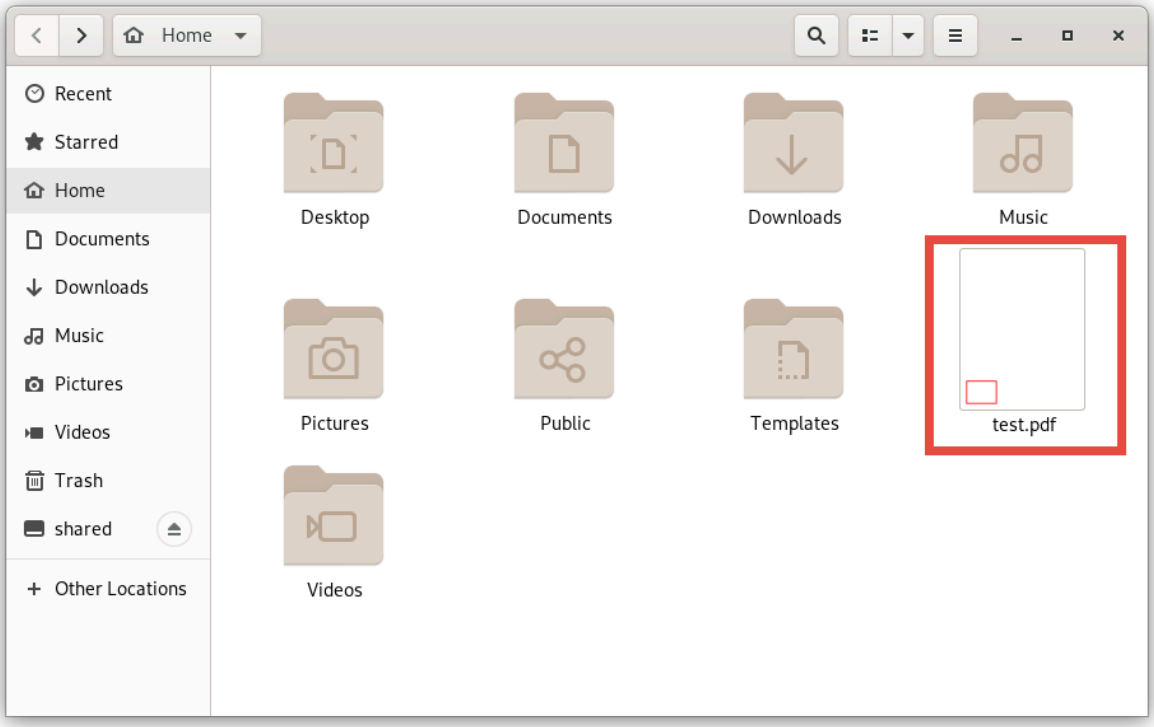
لتحليل ملفنا الأول نحتاج إلى مراعاة أن الأمر الذي نشغله في سطر الأوامر يعمل "من مجلد/دليل" ولذلك نحتاج إلى معرفة مكان تشغيل الأمر ومكان الملف الذي نريد تحليله. لأجل توضيح السياق وفي كل مرة نفتح فيها تطبيق نافذة موجه الأوامر في ريمنوكس، نفتح نافذة موجه الأوامر في الدليل الرئيسي وهو الموقع ذاته الذي نراه عند فتح تطبيق فايلز (Files).





لتسهيل الأمور في الوقت الحالي، يمكننا وضع ملفات بي دي إف الخاصة بنا في هذا المجلد بحيث يتم تنفيذ أمر نافذة موجه الأوامر من نفس دليل ملف بي دي إف الخاص بنا.

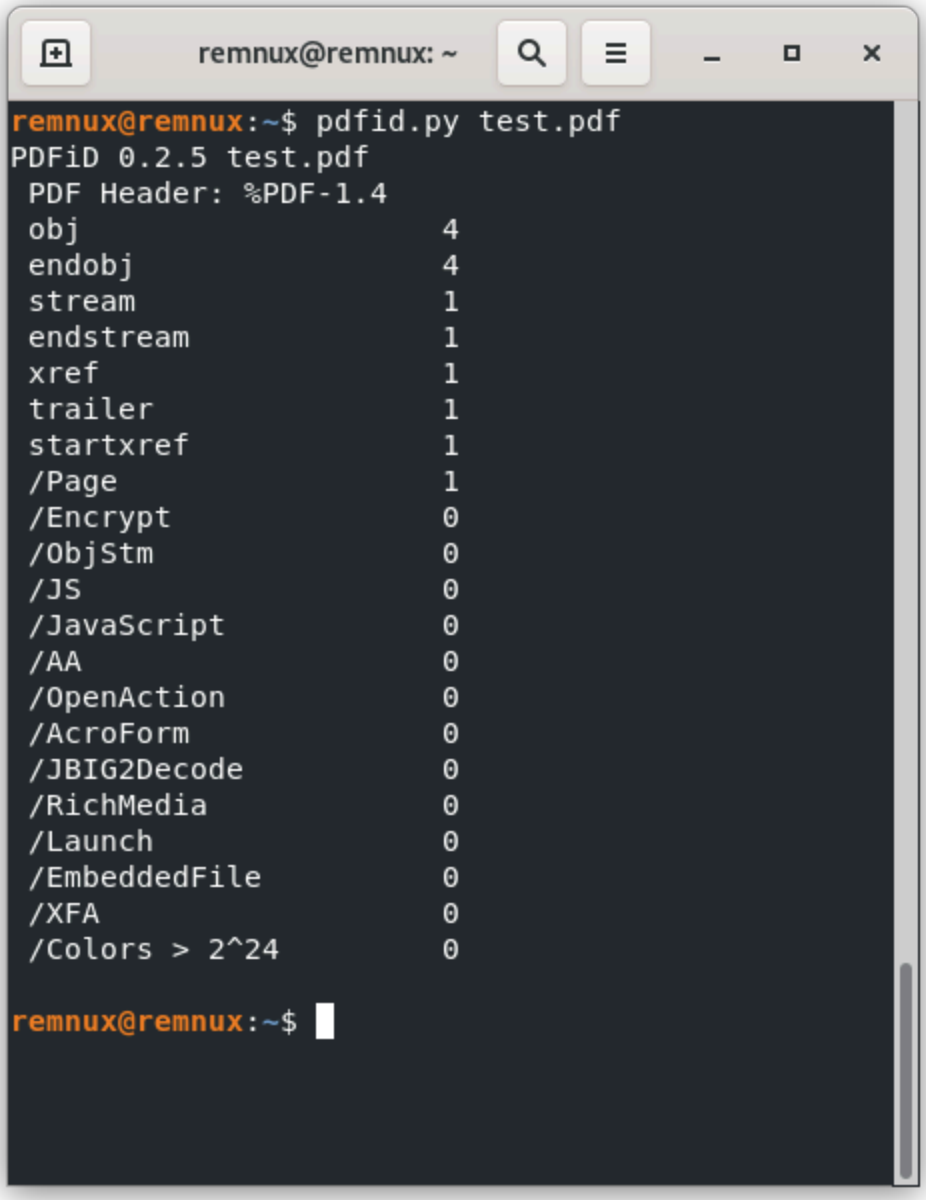
يمكننا نقل ملف المثال من الأعلى وحفظه كملف بي دي إف بمساعدة محرر نصوص على جهاز الكمبيوتر المضيف الخاص بنا وسحب الملف وإفلاته في الدليل الرئيسي لريمنوكس.



وبهذا يمكننا تشغيل الأمر التالي في نافذة موجه الأوامر:

```
pdfid.py test.pdf
```

لتلقي هذا الرد:

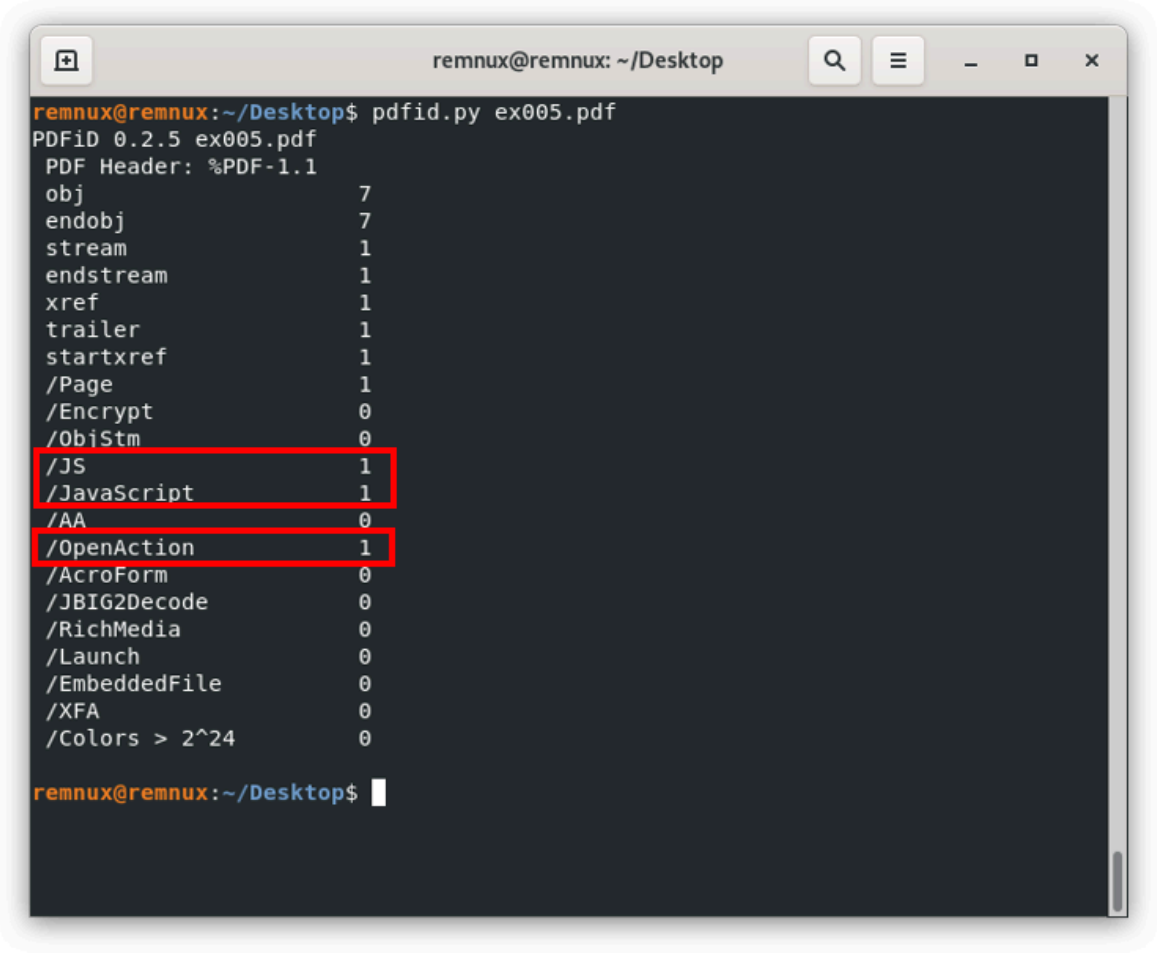


كما يمكننا أن نرى فإن جميع الكائنات التي شاهدها بي دي إف آي دي تتطابق مع تلك التي نعرفها من مصدر ملف بي دي إف ولا يبدو أن أيًا منها مدرج في قائمة الكائنات المشبوهة التي وصفناها أعلاه.

كما ذكرنا سابقًا هناك تقنيات يستخدمها الفاعلون الخبيثون لتجنب الكشف السهل عن أنواع كائنات مختلفة ويحاول بي دي إف آي دي إظهار حتى الأشياء المبهمة، ولكن في بعض الحالات الأقل شيوعًا قد تكون هناك أشياء خفية تتطلب الغوص أعمق قليلًا لاكتشافها.

إدخال pdf-parser، المثال 1

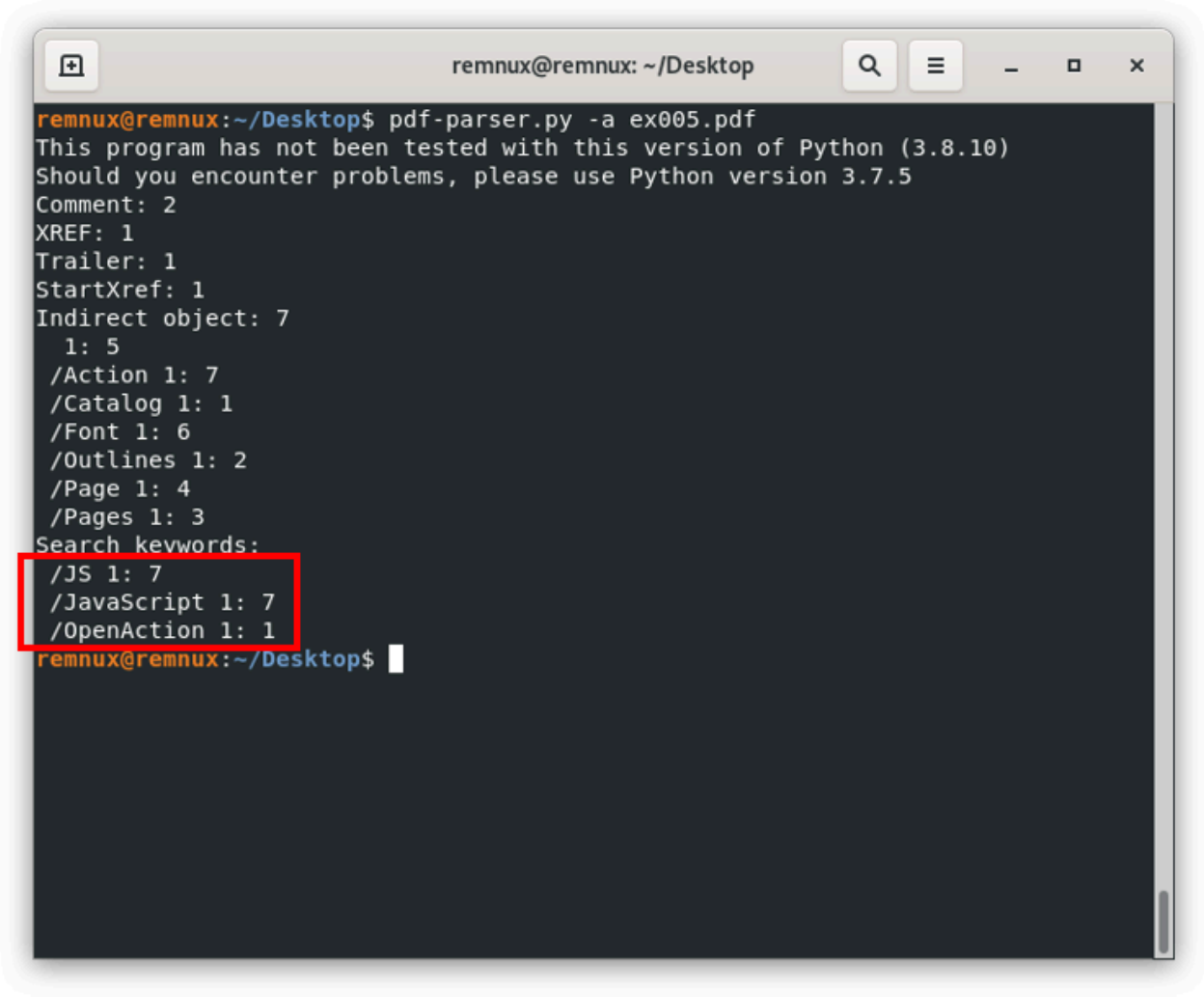
ماذا يحدث عندما نجد ملف بي دي إف يحتوي على كائن مشبوه؟ لنتخيل أن لدينا ملف ex005.pdf يعطينا مخرجات مثل تلك على بي دي إف آي دي:



استنادًا إلى ذلك وإلى الإرشادات أعلاه لهذا المورد نعلم أن هناك 3 كائنات من الأنواع /جيه إس و/جافا سكريبت و/أوبن آكشن قد تستدعي مراجعتها وبالأخص لأنها تشير إلى أن الملف يحاول تنفيذ بعض الإجراءات عند فتح الملف. يمكننا هنا معالجة ذلك باستخدام pdf-parser للتعرف على كل نوع كائن ولرؤية محتوى الكائنات المذكورة. بالنسبة لملف المثال، سنقوم بتنفيذ الأمر التالي:

```
pdf-parser.py -a ex005.pdf
```

نستخدم الوسيطة -a لرؤية بيانات الملف ولأجل الحصول على مرجعية أفضل يمكننا دائمًا استخدام pdf-parser.py –help للمساعدة في رؤية قائمة بالخيارات على الشاشة.

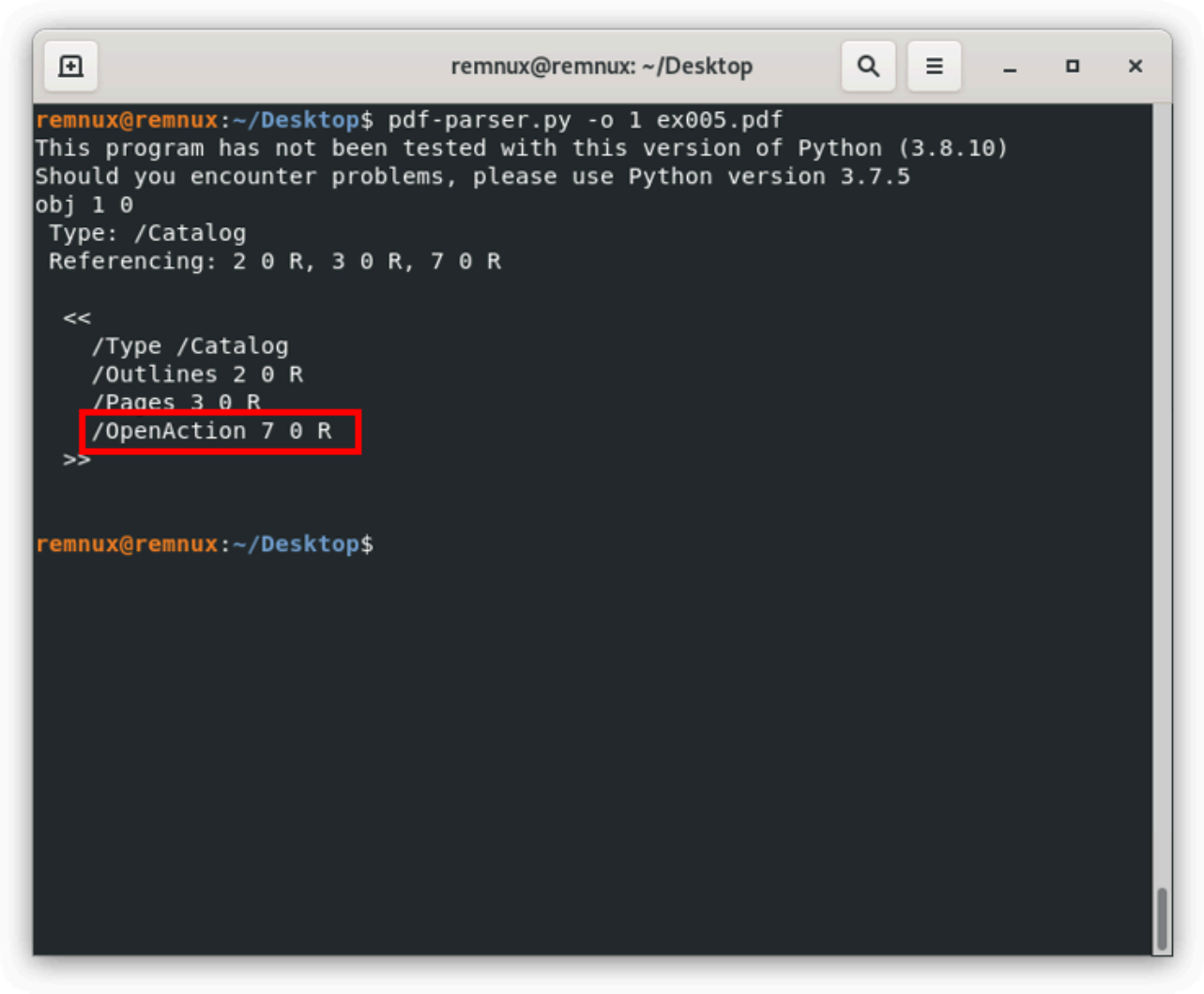


يمكننا أن نرى هنا أن لدينا بالفعل هذه الكائنات الإشكالية الثلاثة ولكن أيضًا بعض المعلومات الإضافية ويمكننا أن نرى معرف الكائنات لكل نوع كائن. نحن نعلم الآن أن كائن (أو كائنات) /جيه إس و/جافا سكريبت تعيش في الكائن الذي يحمل المعرف 7 وأن كائن/أوب آكشن يعيش في الكائن الذي يحمل المعرف 1. يمكننا بعد ذلك رؤية محتوى كائن/أوبن آكشن لمعرفة ما يحاول المستند القيام به عند فتحه ولهذا نستخدم هذا الأمر.

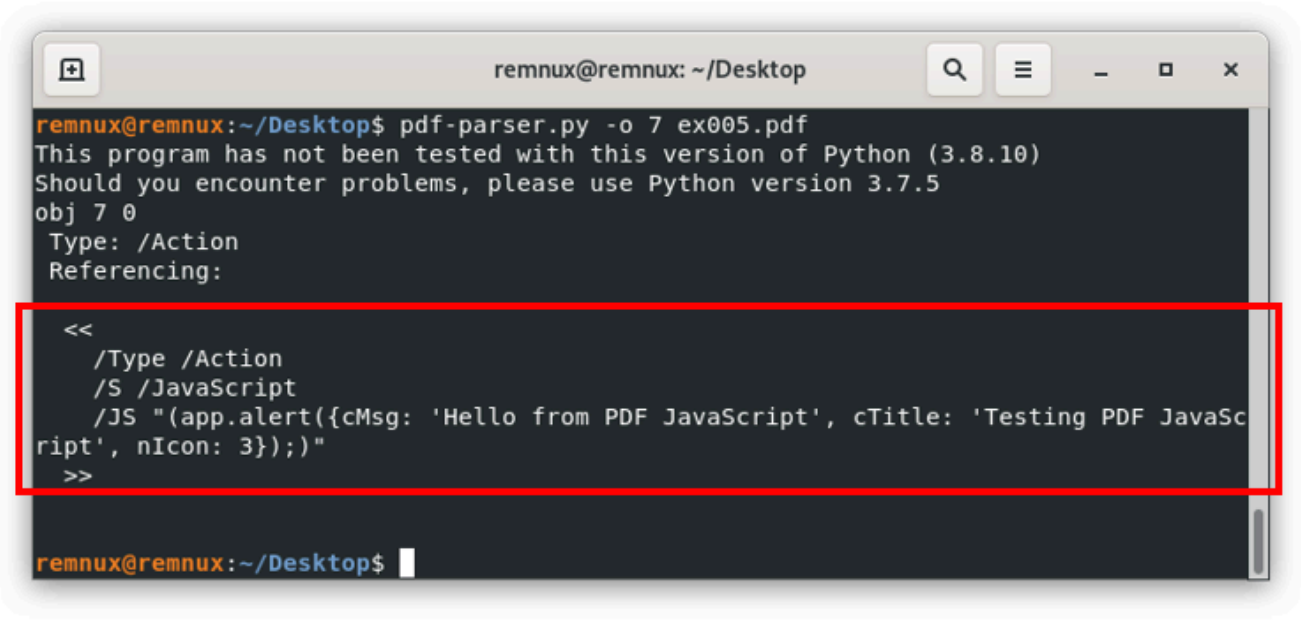
Unset

```
Pdf-parser.py -o 1 ex005.pdf
```

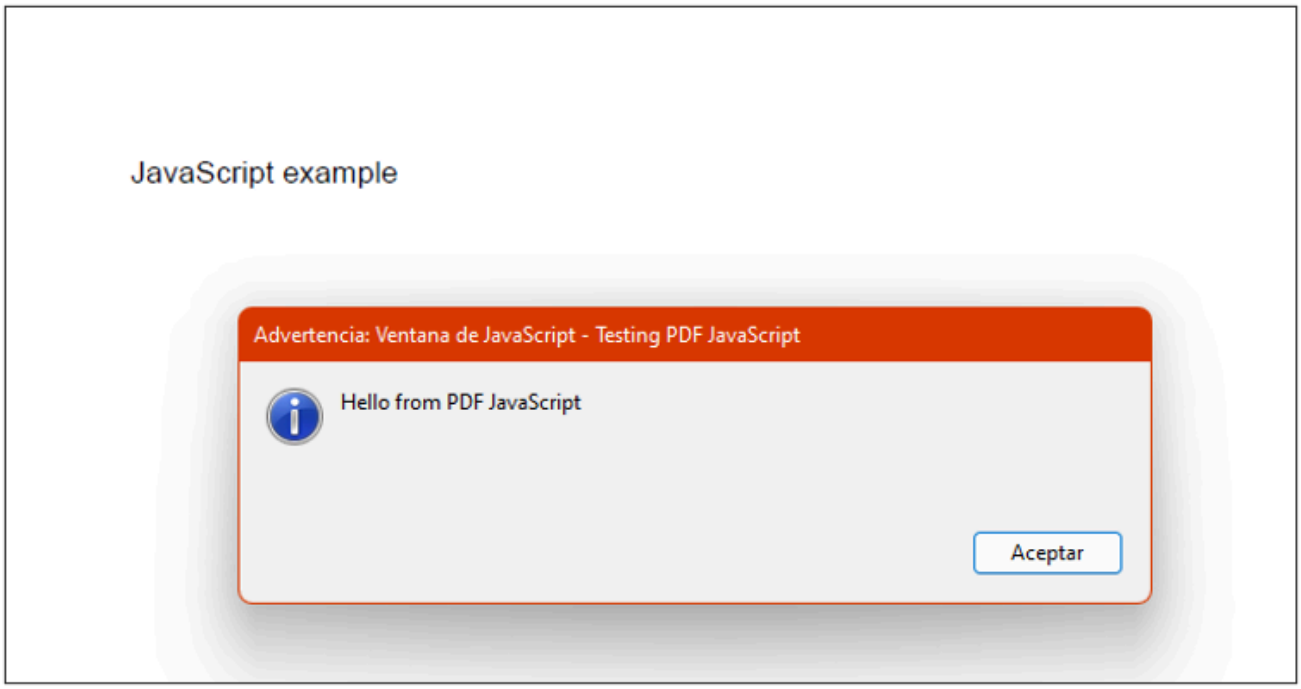
هنا نستخدم الوسيطة -o لإعطاء الأداة معرف الكائن الذي نريد أن نراه على الشاشة:



يمكننا هنا أن نرى السطر "/OpenAction 7 0 R" وهذا يعني أن المحتوى الفعلي للكائن/أوبن آكشن موجود في الكائن بالمعرف 7 وعندما نفتح الملف سنقوم باستدعاء أو الرجوع إلى الكائن المذكور. عند تكرار العملية لرؤية محتوى الكائن الذي يحمل المعرف 7 نحصل على:

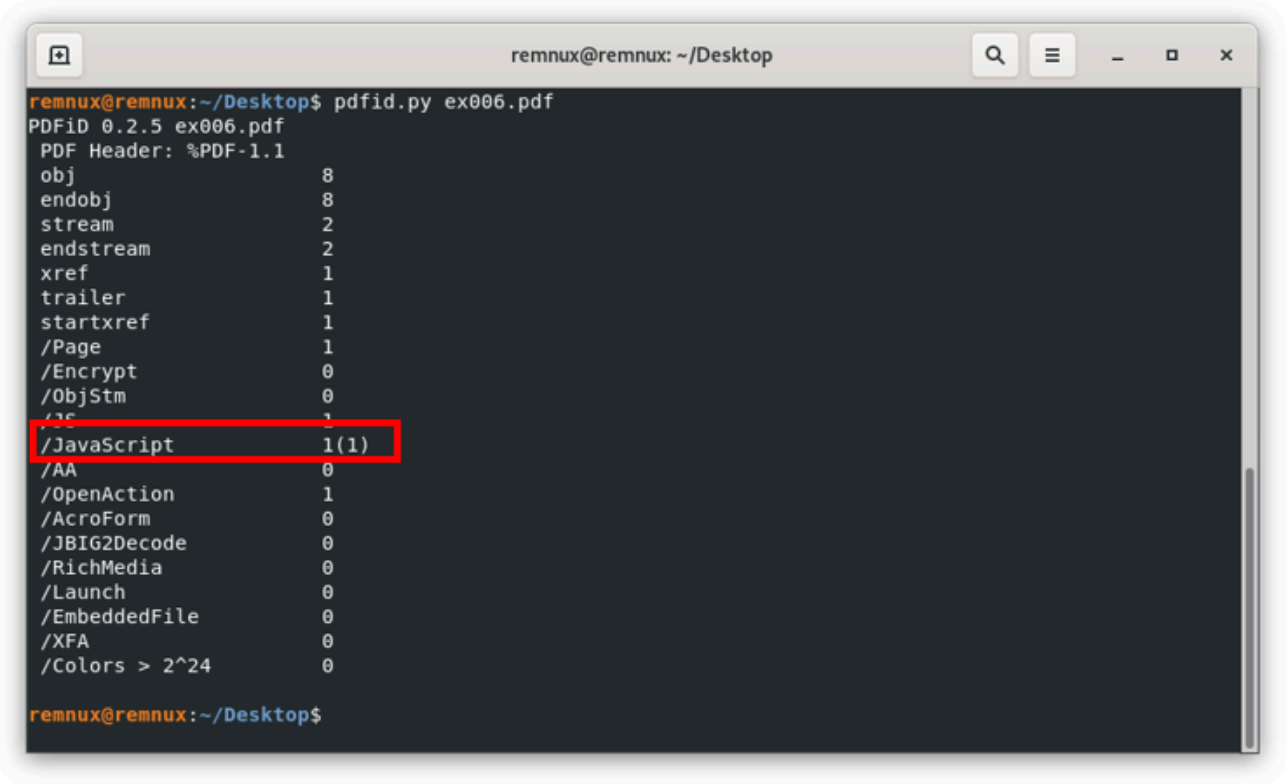


حيث يمكننا أن نرى أن المستند يحاول إظهار تنبيه أو نافذة منبثقة تحتوي على الرسالة الموضحة في نافذة موجه الأوامر، وإذا فتحنا الملف فسيبدو كما يلي:

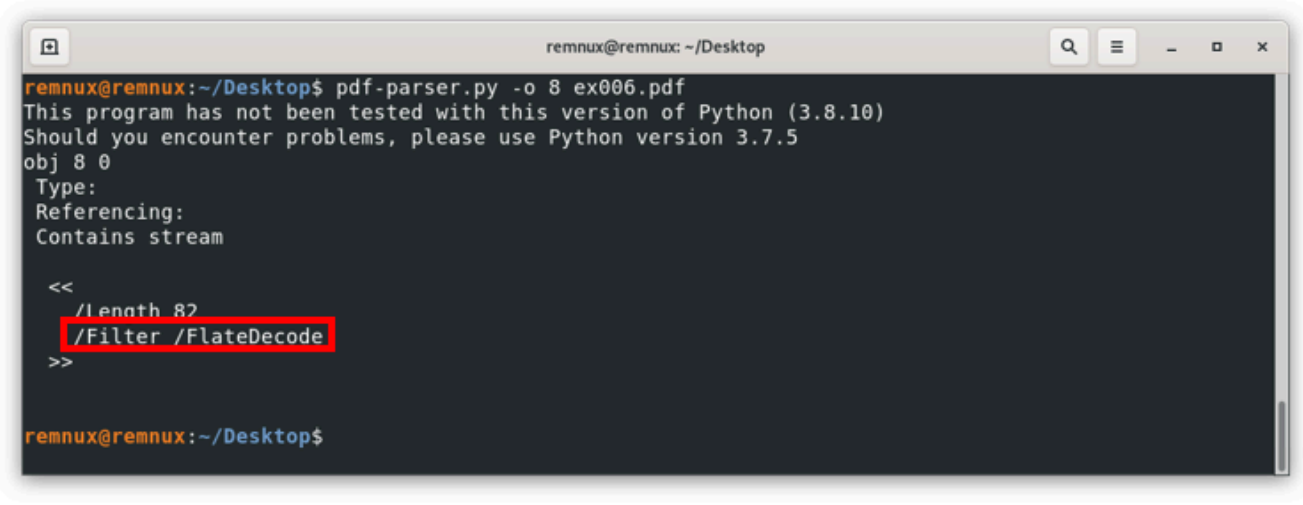


المثال 2

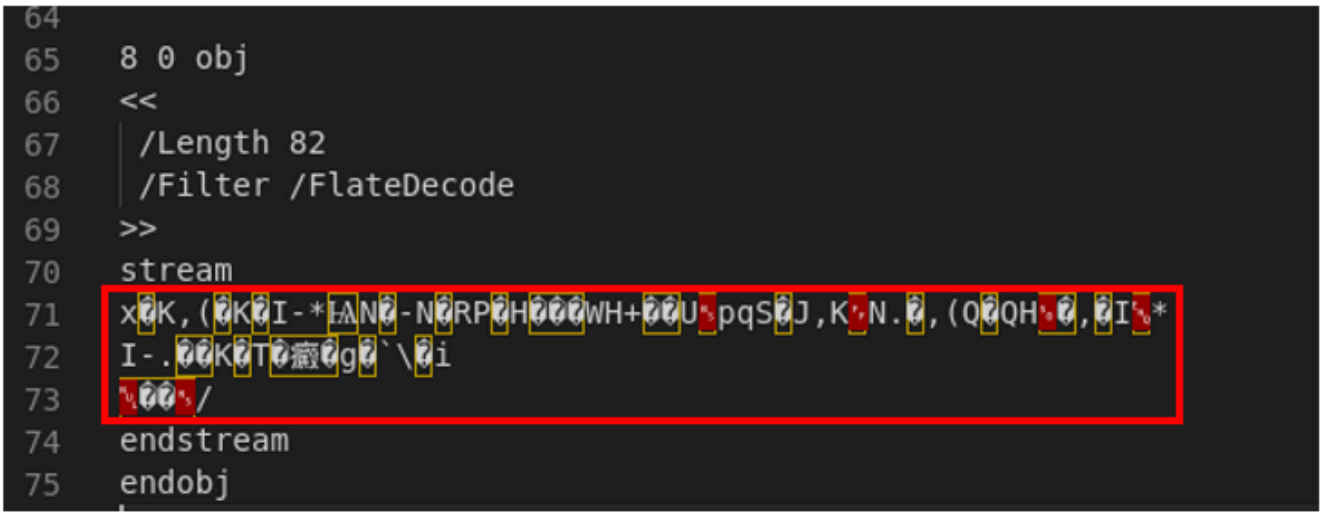
كما ذكرنا من قبل، قد تكون هناك ملفات لا يمكن فيها عرض المحتوى المشبوه بنص عادي وقد يكون هناك عدد من الأسباب للقيام بذلك في الحالات المشروعة، مثل ضغط أجزاء طويلة من المعلومات لتقليل حجم الملف من بين أمور أخرى، ولكن تستخدم الملفات الضارة هذه التقنيات لأغراض الطمس للمساعدة في تجنب الكشف من قبل برامج مكافحة الفيروسات والحلول الأمنية الأخرى. على سبيل المثال، إذا كررنا سير العمل السابق على الملف ex006.pdf، فسنرى أن مخرجات أمر بي دي إف آي دي هي كما يلي:



يمكننا هنا أن نرى في سطر/جافا سكريبت "1(1)" أن هذا يعني أن بي دي إف آي دي اكتشف كائنًا من هذا النوع ولكنه مطموس، وبتكرار سير العمل الذي نعرفه بالفعل نراجع الكائن ذي المعرف 8 (حيث توجد التعليمات البرمجية لجافا سكريبت) لرؤية ما يلي:



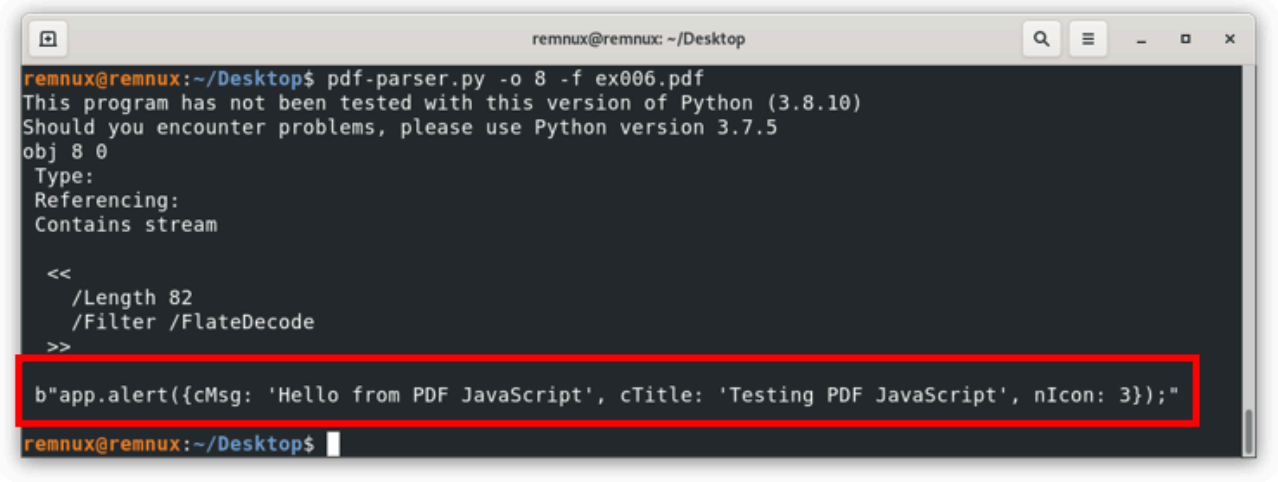
لا يمكننا رؤية التعليمات البرمجية الفعلية كما في المثال الأخير وبدلًا من ذلك نرى من بين أشياء أخرى سطر "/Filter /FlateDecode". يقوم خيار /Filter بتنفيذ عملية على المحتوى النهائي في تدفق لفك تشفيره ثم يقوم /FlateDecode بالإشارة إلى الترميز المرتبط الذي يجب مراعاته عند فك تشفير المحتوى ولإعطاء فكرة أفضل عنه إذا فتحنا الملف باستخدام محرر نص والبحث يدويًا عن هذا العنصر فيجب أن نرى شيئًا من هذا القبيل:



عندما يكون المحتوى داخل المربع الأحمر هو المحتوى المرمز الفعلي، في هذه الحالة يمكن لأداة pdf-parser محاولة فك تشفير المحتوى الفعلي ولهذا نستخدم وسيطة -f وبالتالي نستخدم هذا الأمر:

Unset

```
Pdf-parser.py -o 8 -f ex006.pdf
```

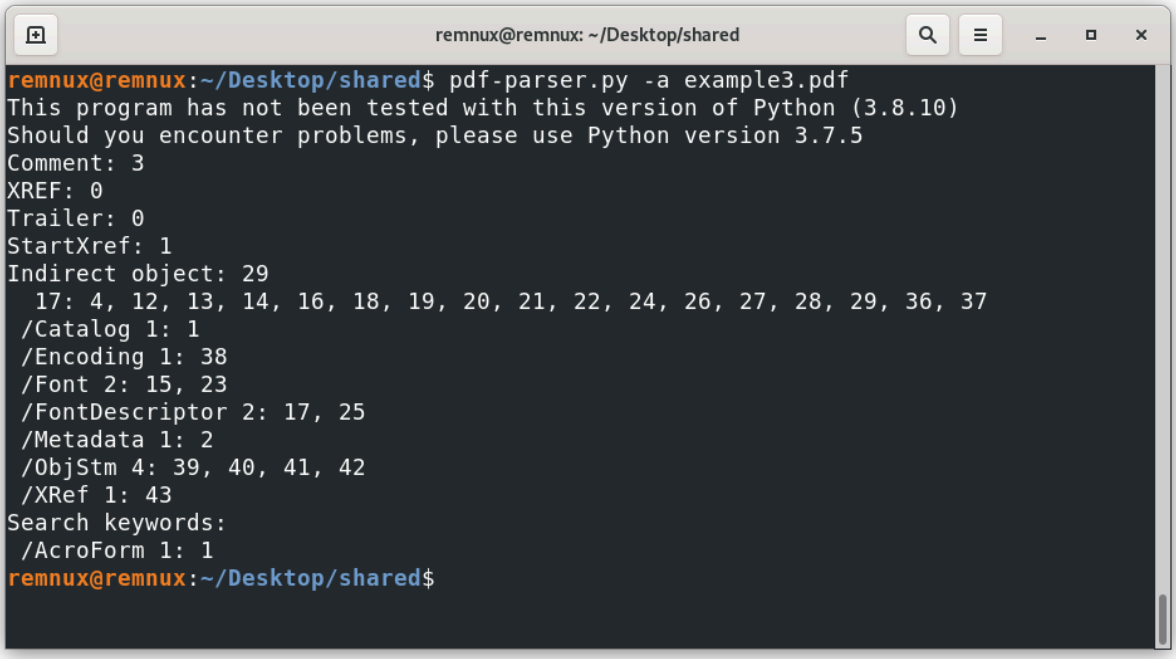


حيث يمكننا رؤية المحتوى الفعلي للكائن الذي سيعرضه قارئ بي دي إف.

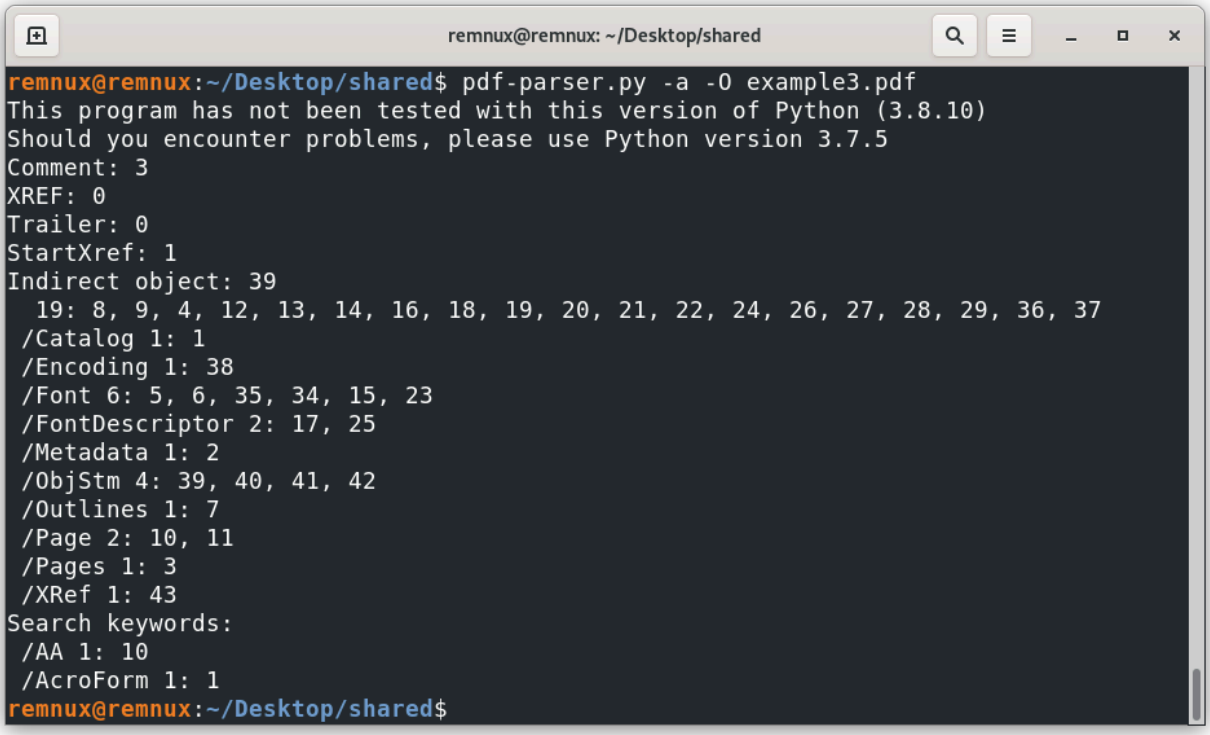
الآن بعد أن عرفنا أساسيات كيفية مراجعة ملفات بي دي إف للبحث عن الأشياء المشبوهة، هناك بعض التحديات التي تواجهك.

المثال 3

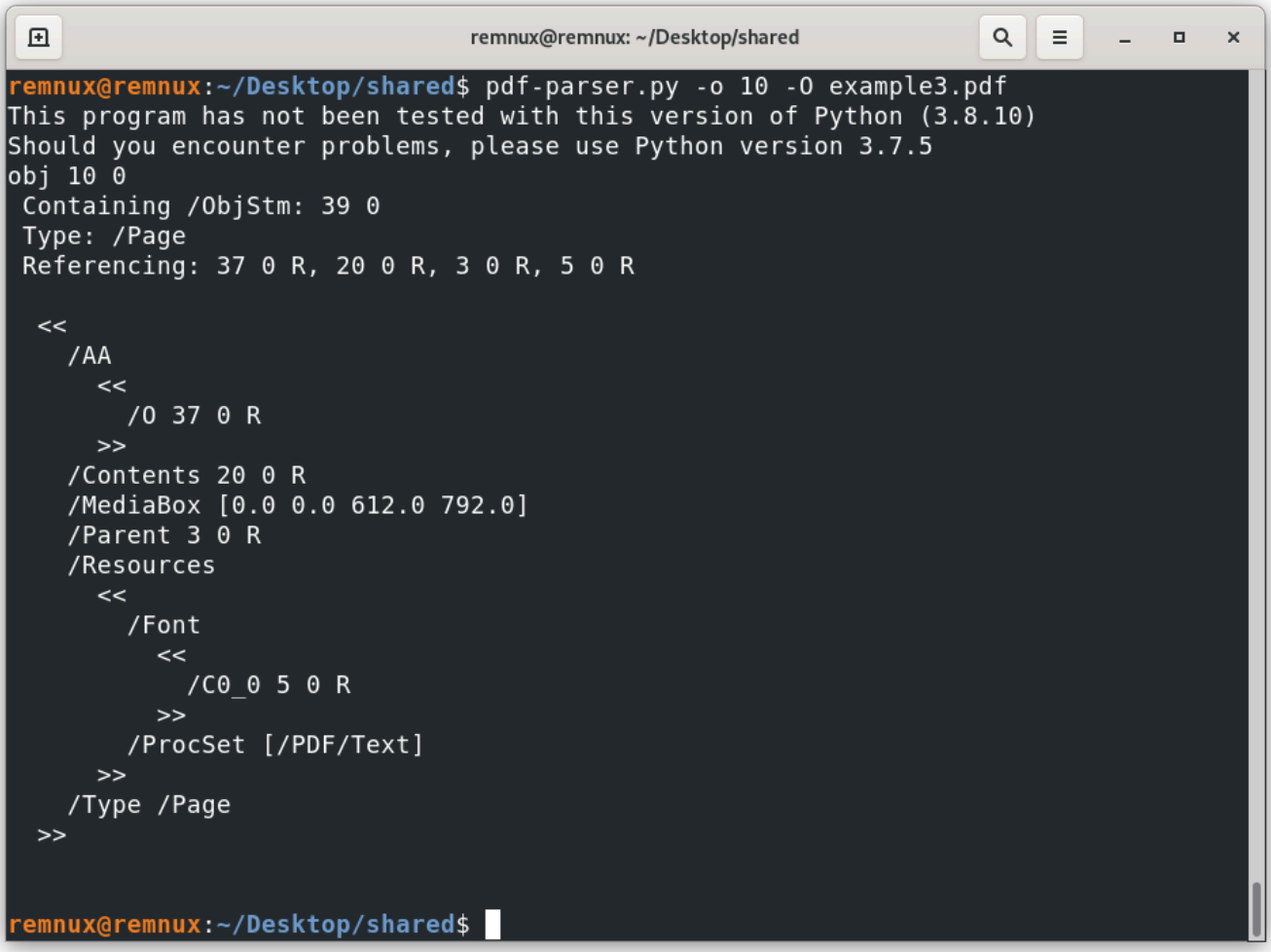
هناك أمر آخر يقوم به برنامج منشئ ملفات بي دي إف عادة لإنشاء ملفات جديدة وهو إنشاء كائنات مرمزة ضمن التدفق لجعل الملفات الناتجة أصغر وهو أمر مرغوب فيه بشكل عام ولكنه ينشئ أيضًا طريقة لتعتيم أكبر على التعليمات البرمجية الضارة، ومن خلال تحليل الملف example3.pdf نرى بعض /أوبجستم (أو تدفقات الكائنات) التي قد تحتوي كائنات أخرى قد تسترعي الانتباه، وهي في الواقع تحتويها.



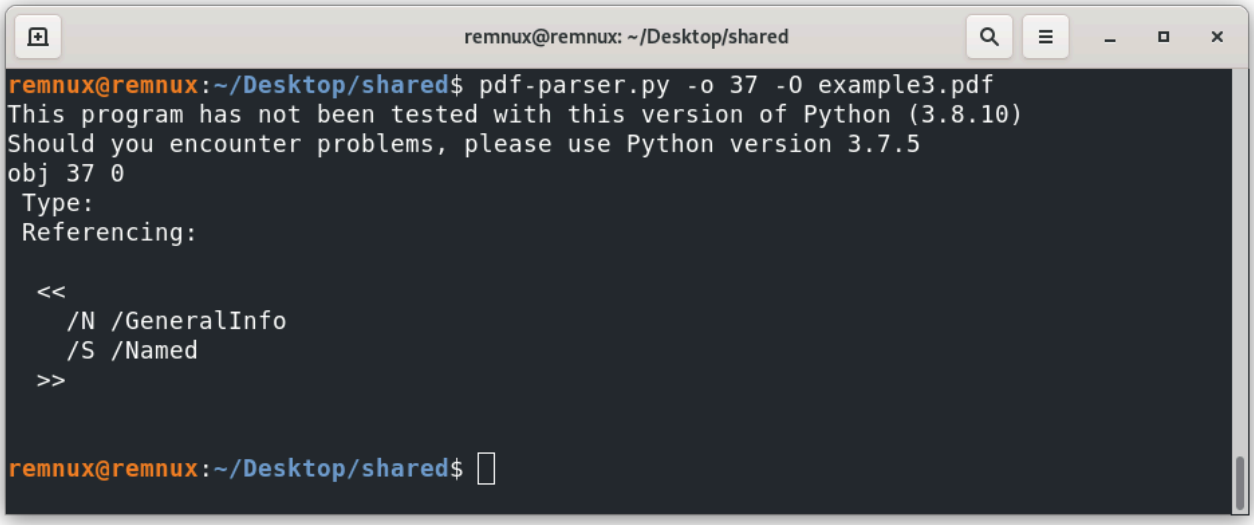
بالنسبة لهذا النوع من السيناريوهات، يُنصح باستخدام الخيار O- (أي حرف o كبير) لأدلة pdf-parser سيحاول هذا الخيار تحليل أي تدفق يحتوي على كائن ومعاملته باعتباره كائنًا عاديًا للملف، على سبيل المثال باستخدام هذا الخيار بهذه الطريقة.



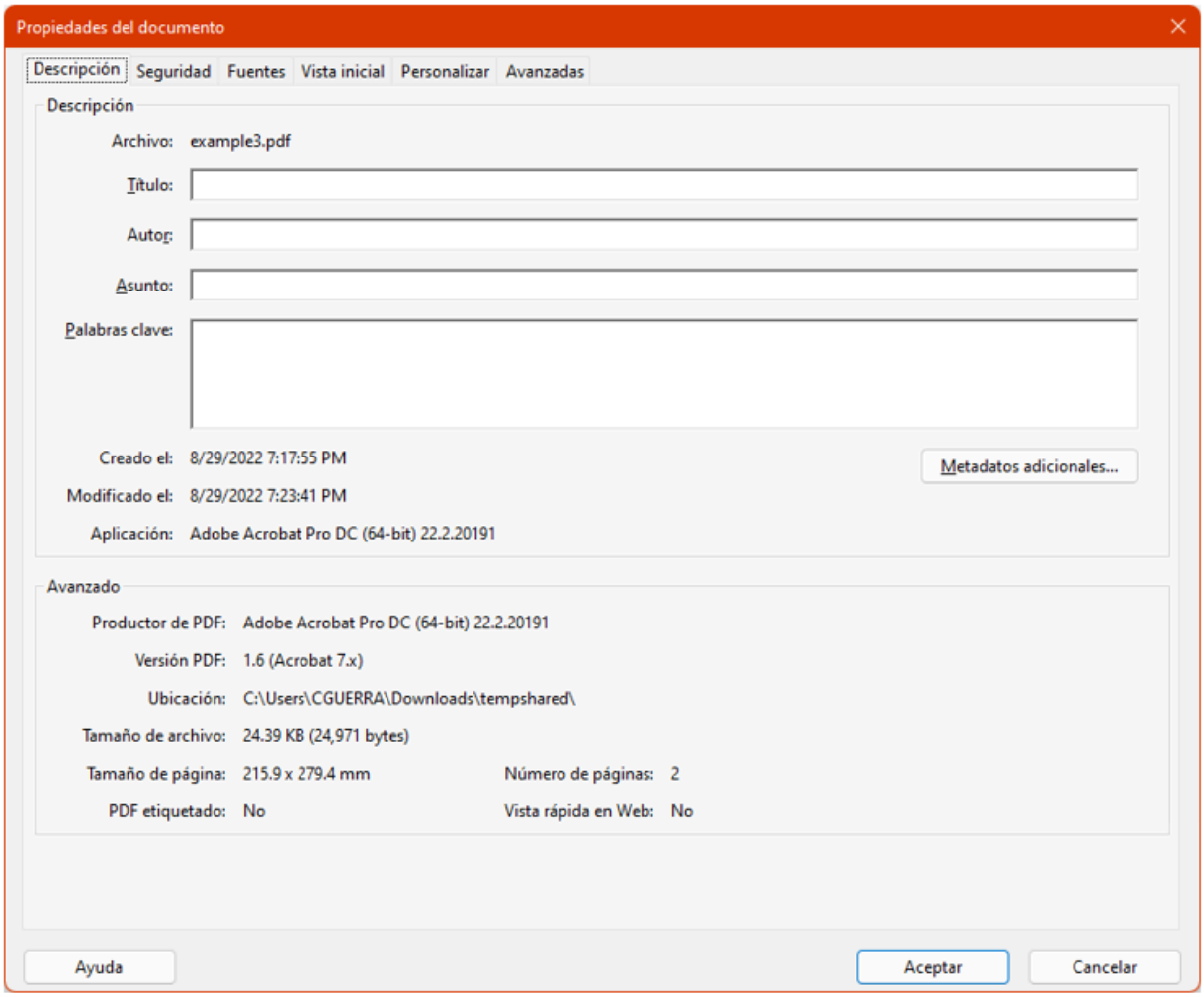
كشف أن الملف يحتوي على كائنات "جديدة" وأن أحدها هو /AA وهو أمر مثير للاهتمام للبحث عن السلوك الضار من خلال النظر إلى الكائن المعني الذي لدينا



يخبرنا أن الإجراء مرتبط بكائن صفحة وفي هذه الحالة يشير /O إلى تشغيل الإجراء عند فتح الصفحة ويخزن الإجراء الفعلي في الكائن 37، وبعد ذلك



بعد البحث يمكننا أن نستنتج أن هذا الكائن يحاول فتح مربع حوار خصائص قارئ بي دي إف مثل هذا (ليس هذا شيئًا خطيرًا، مثال في جهاز كمبيوتر تم تكوينه باللغة الإسبانية)



الختام: حاول استخدام المعلمة O- في حالة وجود كائنات أخرى تختبئ ضمن التدفقات

التحديات

السؤال: تحليل الملف challenge1.pdf (md5: 3b20821cb817e40e088d9583e8699938)، ما نوع العنصر المثير للاهتمام الذي تراه مختبئاً خلف تدفق؟

1. أوبن آكشن (OpenAction)

غير صحيح -- ...

2. إيه إيه (AA)

صحيح -- ...

3. جافا سكريبت (جيه إس)

غير صحيح -- ...

4. جافا سكريبت (JavaScript)

غير صحيح -- ...

السؤال: تحليل الملف challenge2.pdf (md5: 30373b268d516845751c10dc2b579c97)، يمكننا رؤية إجراء يحاول فتح عنوان موقع ويب، ما الرمز الذي يتضمنه عنوان موقع الويب باعتباره رمز تتبع؟ (ملاحظة: 6 أرقام)

أريد مشاهدة الإجابة

88965

الآن بعد أن عرفنا المزيد عن ملفات بي دي إف، دعونا نغطي في الجزء التالي نوعًا آخر من الملفات التي يكثر استخدامها كأسلحة وهي ملفات أوفيس (Office).