‹ Hat Arsenal peepdf Challenge 2015 writeup

·epdf Challenge 2015 writeup

2 comments

a link on twitter by *Jose Miguel Esparza*, the author of **peepdf** tool, about a challenge he created for *Black Hat* eading the blog post I decided to play with the challenge and now here's my writeup solution. I hope that you

: of your time to try to solve the challenge without reading the solution. It's a very fun this challenge… Then ›lution. 😉

:ontext, you can read the blog post by Jose in his blog here.

·nge here or directly from the blog post linked above.

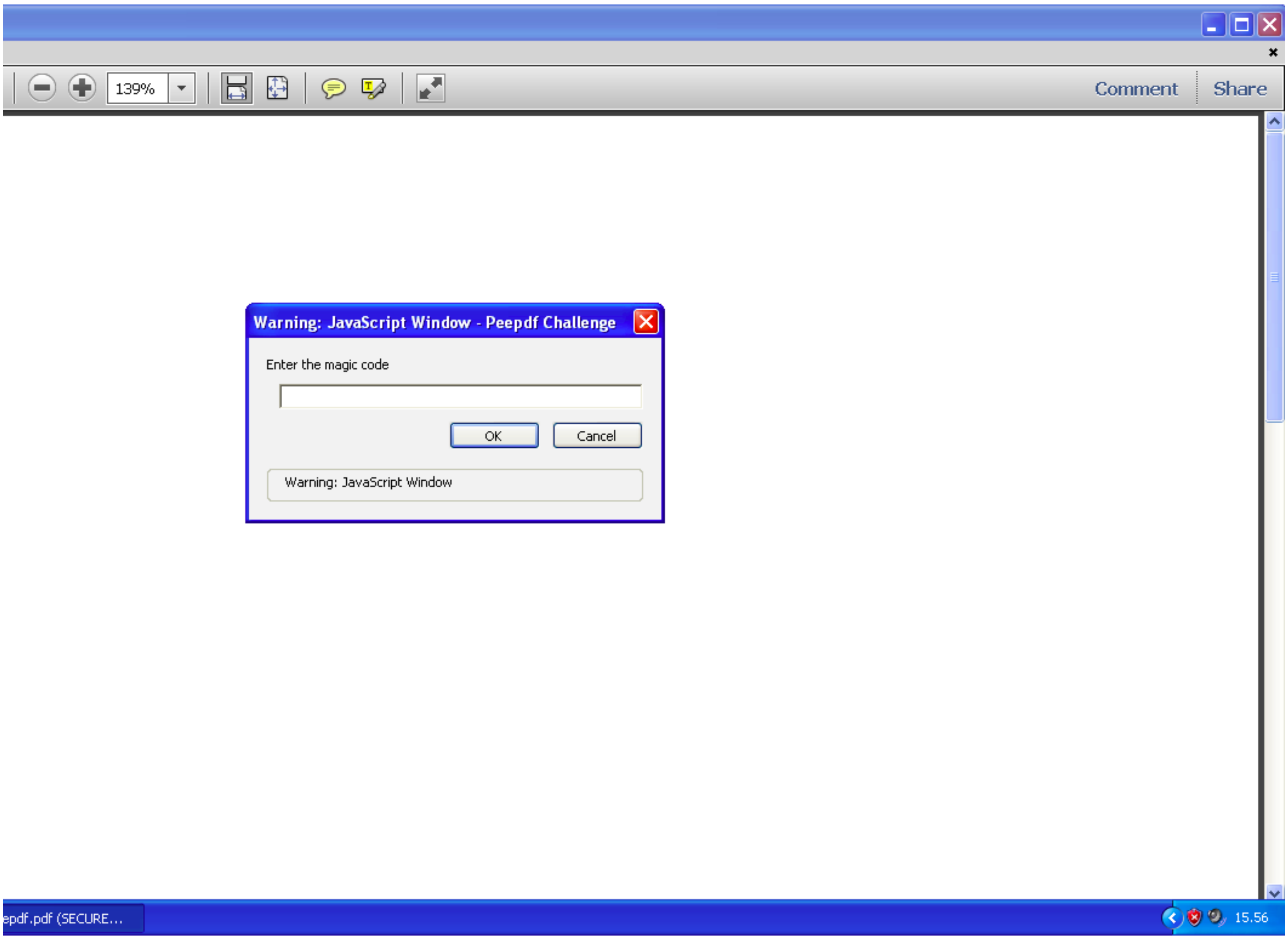
:an open with Adobe Reader on your PC without using a VM. The version of Adobe Reader we have to use is ıe, otherwise you can't understand and play the challenge. 😛

ı of Adobe Reader from this site: Adobe Reader X

ȷo to solve the challenge!

ʻ a page with the following image:

that there is an attachment inside this pdf named ***peepdf.pdf.*** So let's save and open it. This new pdf asks you
ıvascript form:



Enter the magic code

Warning: JavaScript Window

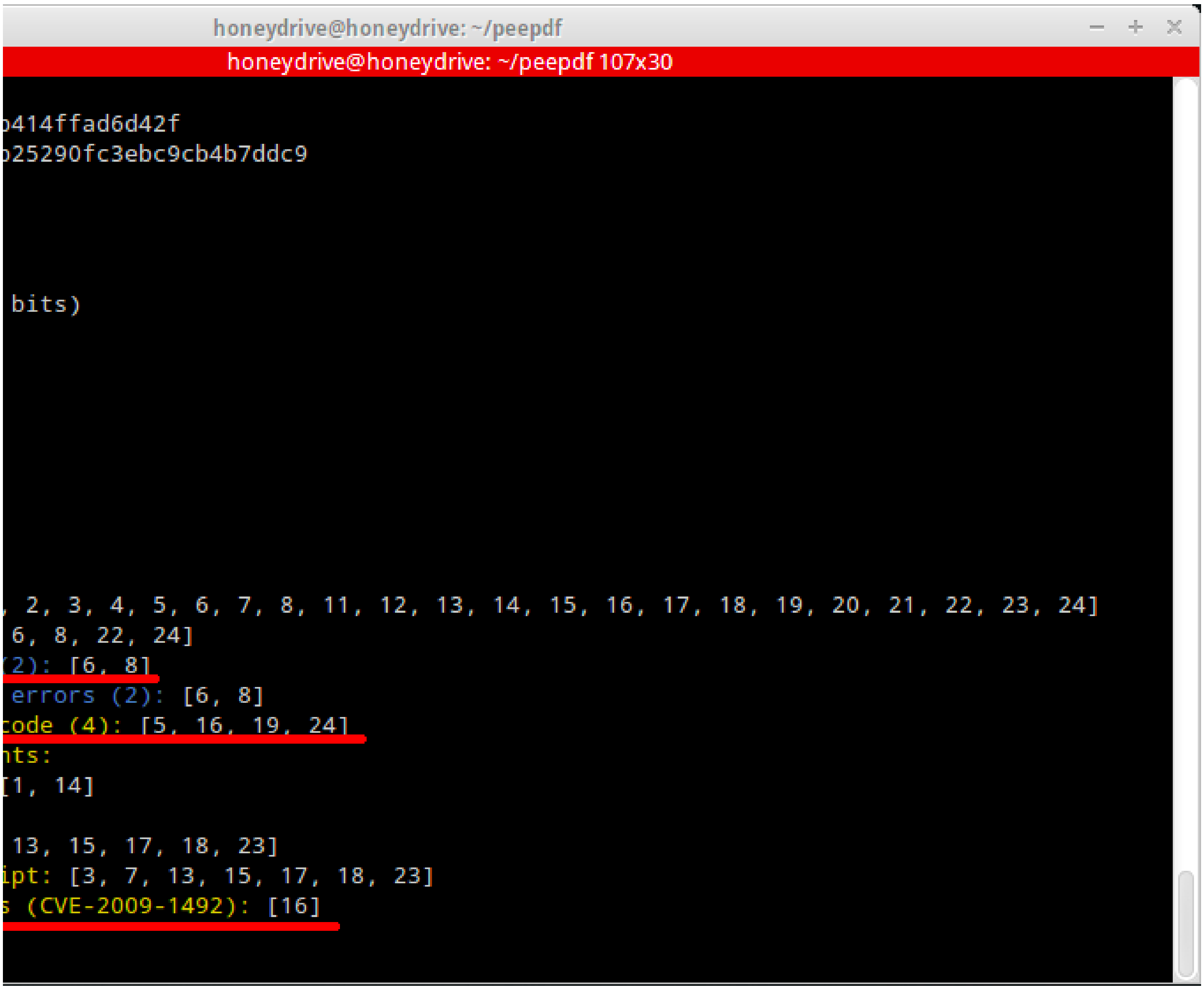
ackme, instead of an executable we have to break a PDF, cool, isn't it?

ɹ can also use *peepdf* tool which we'll see later. For example when you have identified the object that contains ɔle simply use the command:

```
ɛepdf.pdf
```

ɛ and analyze the pdf extracted to understand what is the password, that is the **flag!**

ɾepo we do a *git clone* to download peepdf and open the pdf in it. I recommend using the last version for a ɑ moment. 😉 We'll see the following info:


honeydrive@honeydrive: ~/peepdf
honeydrive@honeydrive: ~/peepdf 107x30

```
b414ffad6d42f
b25290fc3ebc9cb4b7ddc9




 bits)










, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
 6, 8, 22, 24]
(2): [6, 8]
 errors (2): [6, 8]
code (4): [5, 16, 19, 24]
nts:
[1, 14]

 13, 15, 17, 18, 23]
ipt: [3, 7, 13, 15, 17, 18, 23]
s (CVE-2009-1492): [16]
```

which contain Javascript code:

```


iewerVersion.toString().split(".")[0];

le:"Peepdf Challenge",cMsg:"You should try with an older version of Adobe Reader

rue);


(this.info.author)));







an();

etAnnots({nPage:0});
nPages].subject;
'x1/);
```

```
: buf.length; n++) {
)mCharCode("0" + "x" + buf[n]);









ı){var kk = "";for(var i=0;i<data.length;i++){kk +=
(data.charCodeAt(i) ^ key.charCodeAt(i%key.length));}return kk}






›



Paul Johnston 1999 - 2000.
; Holt 2000 - 2001.
ıome.org.uk/site/legal.html for details.

23456789abcdef";
)



 3; j++)
·.charAt((num >> (j * 8 + 4)) & 0x0F) +
·.charAt((num >> (j * 8)) & 0x0F);




(str)

ıgth + 8) >> 6) + 1;
/(nblk * 16);
ıblk * 16; i++) blks[i] = 0;
;tr.length; i++)
|= str.charCodeAt(i) << ((i % 4) * 8);
 0x80 << ((i % 4) * 8);
· 2] = str.length * 8;




)

)xFFFF) + (y & 0xFFFF);
 16) + (y >> 16) + (lsw >> 16);
L6) | (lsw & 0xFFFF);


 cnt)

:nt) | (num >>> (32 - cnt));


, b, x, s, t)

ıdd(add(a, q), add(x, t)), s), b);

 c, d, x, s, t)

 c) | ((~b) & d), a, b, x, s, t);
 c, d, x, s, t)

 d) | (c & (~d)), a, b, x, s, t);

 c, d, x, s, t)

 ^ d, a, b, x, s, t);

 c, d, x, s, t)

b | (~d)), a, b, x, s, t);


);



.length; i += 16)





, d, x[i+ 0], 7 , -680876936);
, c, x[i+ 1], 12, -389564586);
, b, x[i+ 2], 17,  606105819);
, a, x[i+ 3], 22, -1044525330);
, d, x[i+ 4], 7 , -176418897);
, c, x[i+ 5], 12,  1200080426);
, b, x[i+ 6], 17, -1473231341);
, a, x[i+ 7], 22, -45705983);
, d, x[i+ 8], 7 ,  1770035416);
, c, x[i+ 9], 12, -1958414417);
, b, x[i+10], 17, -42063);
, a, x[i+11], 22, -1990404162);
, d, x[i+12], 7 ,  1804603682);
, c, x[i+13], 12, -40341101);
, b, x[i+14], 17, -1502002290);
, a, x[i+15], 22,  1236535329);

, d, x[i+ 1], 5 , -165796510);
, c, x[i+ 6], 9 , -1069501632);
, b, x[i+11], 14,  643717713);
, a, x[i+ 0], 20, -373897302);
, d, x[i+ 5], 5 , -701558691);
, c, x[i+10], 9 ,  38016083);
, b, x[i+15], 14, -660478335);
, a, x[i+ 4], 20, -405537848);
, d, x[i+ 9], 5 ,  568446438);
, c, x[i+14], 9 , -1019803690);
, b, x[i+ 3], 14, -187363961);
, a, x[i+ 8], 20,  1163531501);
, d, x[i+13], 5 , -1444681467);
, c, x[i+ 2], 9 , -51403784);
, b, x[i+ 7], 14,  1735328473);
, a, x[i+12], 20, -1926607734);

, d, x[i+ 5], 4 , -378558);
, c, x[i+ 8], 11, -2022574463);
```

```
ı, b, x[i+11], 16,  1839030562);
ł, a, x[i+14], 23, -35309556);
:, d, x[i+ 1], 4 , -1530992060);
), c, x[i+ 4], 11,  1272893353);
ı, b, x[i+ 7], 16, -155497632);
ł, a, x[i+10], 23, -1094730640);
:, d, x[i+13], 4 ,  681279174);
), c, x[i+ 0], 11, -358537222);
ı, b, x[i+ 3], 16, -722521979);
ł, a, x[i+ 6], 23,  76029189);
:, d, x[i+ 9], 4 , -640364487);
), c, x[i+12], 11, -421815835);
ı, b, x[i+15], 16,  530742520);
ł, a, x[i+ 2], 23, -995338651);

:, d, x[i+ 0], 6 , -198630844);
), c, x[i+ 7], 10,  1126891415);
ı, b, x[i+14], 15, -1416354905);
ł, a, x[i+ 5], 21, -57434055);
:, d, x[i+12], 6 ,  1700485571);
), c, x[i+ 3], 10, -1894986606);
ı, b, x[i+10], 15, -1051523);
ł, a, x[i+ 1], 21, -2054922799);
:, d, x[i+ 8], 6 ,  1873313359);
), c, x[i+15], 10, -30611744);
ı, b, x[i+ 6], 15, -1560198380);
ł, a, x[i+13], 21,  1309151649);
:, d, x[i+ 4], 6 , -145523070);
), c, x[i+11], 10, -1120210379);
ı, b, x[i+ 2], 15,  718787259);
ł, a, x[i+ 9], 21, -343485551);

ła);
łb);
łc);
łd);

⊦ rhex(b) + rhex(c) + rhex(d);
```

check if the Adobe Reader version is higher than 10. If this is true the string "*You should try with an older* ›pears and the pdf is closed, else run *peepdf* function (which we'll understand what it does later).

nots() function (CVE-2009-1492). This function returns an array of ***Annotation*** objects and through ***subject*** ınnotations info. But where is the annotation array returned by ***getAnnots()***? It's simple, in the pdf file format ɜ is named **/Annots** so let's go to investigate the /Annots tag to discover what object contains this info. To ɪse the '***search***' command like this:

```
;




ıt << /F1 << /Type /Font

› >> >>
JavaScript
```

```
21 0 R ]
```

ȷ for, which refers to 2 specific objects: *20* and *21*. What is the right object? It's simple because if you check the

ıge:0});
:t;

ȷe is: *take the second object (because numPages = 1) of annots array and read the Subject field*. So the

```
300 210 ]
```

ɔ object *22*:

```
x120x13dx120x17bx10ax109x12fx12fx168x174x174x170x13ax12fx12fx177x177x177x12ex177x165x162
```

f *buf* var it's straightforward what the js does. It takes the content of the object 22, removes "*x1*" char and with
verts the hex value in the corresponding ASCII character. To accomplish to this task you have different options:
verter tool by Kahu Security, or, like I did, directly in peepdf.

```
to_replace
ble to_replace x1 ''
 replaced correctly
```

ımand we obtain the final value of the *z* var:

```
ble to_replace ahx

olkit.info/
ABCDEFGHInopqrstuvwxyz01234JKLMNOPQefghijklm56789+/=",
:) {

4;
4;

lace(/[^A-Za-z0-9\+\/\=]/g, "");
length) {
Of(input.charAt(i++));
Of(input.charAt(i++));
Of(input.charAt(i++));
Of(input.charAt(i++));
(e2 >> 4);
: 4) | (e3 >> 2);
 6) | e4;
fromCharCode(c1);
 = kk + String.fromCharCode(c2);}
 = kk + String.fromCharCode(c3);}
```

:tion definition used by js code contained in object 5.

/hich executes the code that checks the password typed in.

:ct 5 to understand the function inside the else statement.

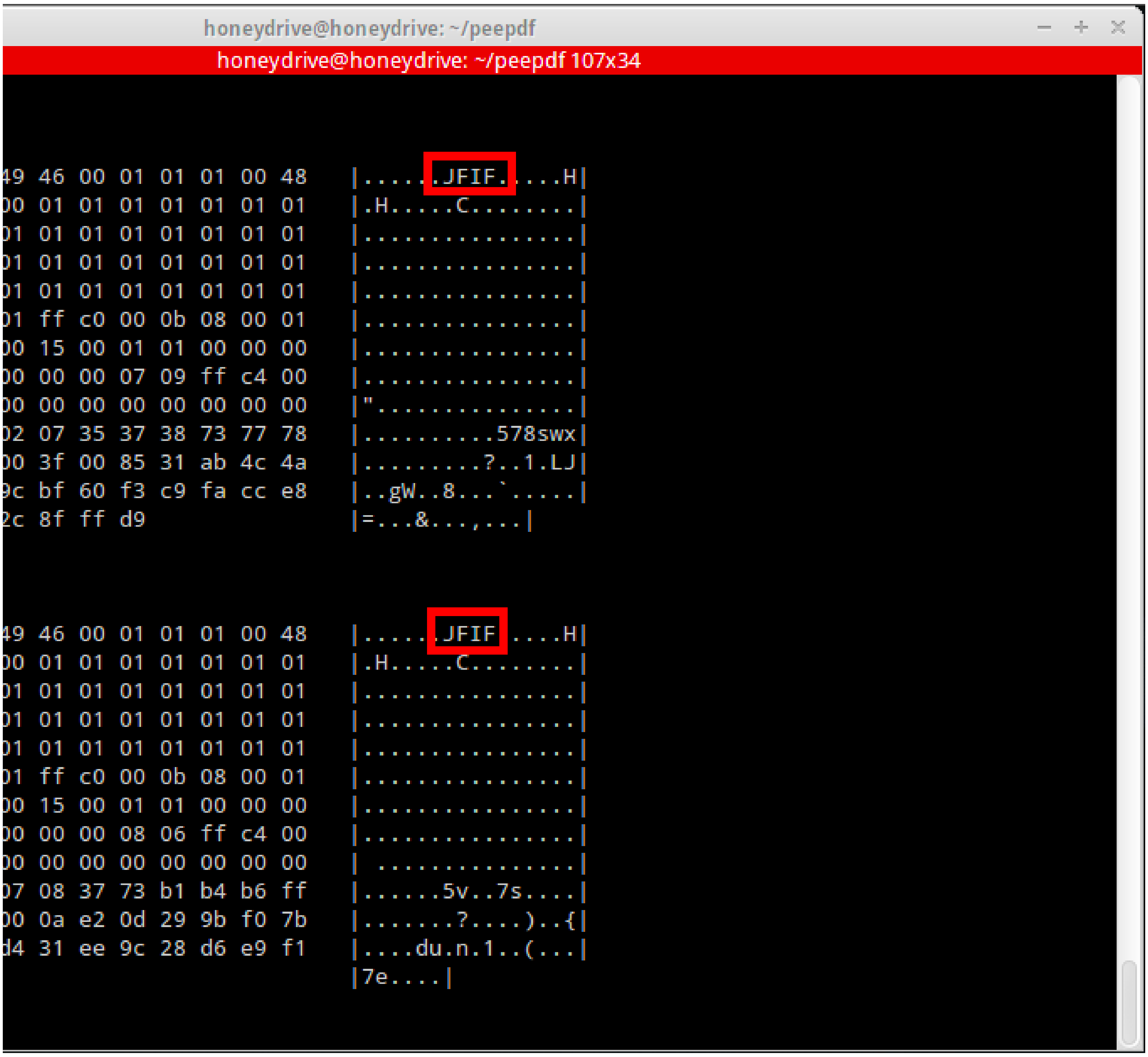
a (the ***key***) and *x.d(this.info.author)* (the ***data***). First we must search the data block to be decrypted by *r* ɔlock is very simple because we must find the object which contain the pdf *info*, that is **/Author** tag. To do that ȷ of the post and we can see a row like this: ***Info: 12***. So jump to object *12*:

```
19820925000000

.brary X
5153000
.3.3.14 >>
```

```
KZgxaCyxiAUeIbxx5aTxaBymjbndwWRHtAU9rBy8/qhEtAxZctFmIbyamrl2vSDZtCFyRsTt/Zz2qAiNMBFenZyZ
```

e looking for. Now the only thing left to do is find the key (first parameter of *r* function) to decrypt this data block.

'ided by the info command we can see that there are 2 decoding errors for objects 6 and 8. So let's inspect the

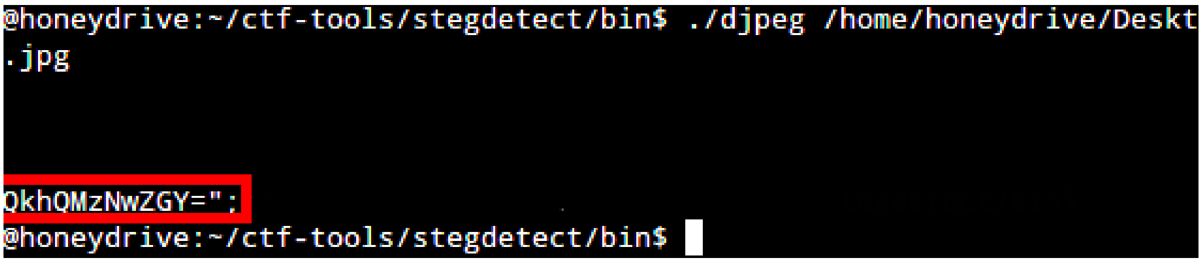

honeydrive@honeydrive: ~/peepdf
honeydrive@honeydrive: ~/peepdf 107x34

```
49 46 00 01 01 01 00 48   |......JFIF.....H|
00 01 01 01 01 01 01 01   |.H.....C........|
01 01 01 01 01 01 01 01   |................|
01 01 01 01 01 01 01 01   |................|
01 01 01 01 01 01 01 01   |................|
01 ff c0 00 0b 08 00 01   |................|
00 15 00 01 01 00 00 00   |................|
00 00 00 07 09 ff c4 00   |................|
00 00 00 00 00 00 00 00   |"...............|
02 07 35 37 38 73 77 78   |..........578swx|
00 3f 00 85 31 ab 4c 4a   |.........?..1.LJ|
9c bf 60 f3 c9 fa cc e8   |..gW..8...`.....|
2c 8f ff d9               |=...&...,...|


49 46 00 01 01 01 00 48   |......JFIF.....H|
00 01 01 01 01 01 01 01   |.H.....C........|
01 01 01 01 01 01 01 01   |................|
01 01 01 01 01 01 01 01   |................|
01 01 01 01 01 01 01 01   |................|
01 ff c0 00 0b 08 00 01   |................|
00 15 00 01 01 00 00 00   |................|
00 00 00 08 06 ff c4 00   |................|
00 00 00 00 00 00 00 00   | ...............|
07 08 37 73 b1 b4 b6 ff   |......5v..7s....|
00 0a e2 0d 29 9b f0 7b   |.......?....)..{|
d4 31 ee 9c 28 d6 e9 f1   |....du.n.1..(...|
                          |7e....|
```

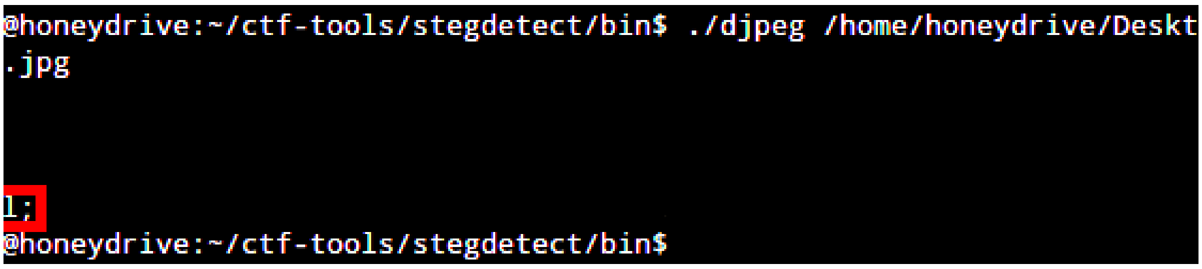
ı filters: */ASCIIHexDecode /DCTDecode.* The first decodes data encoded in an ASCII hexadecimal
original binary data, the second instead decompresses data encoded using a DCT (discrete cosine transform)
andard, reproducing image sample data that approximates the original data. Moreover we can notice the
presence of DCTDecode filter plus the marker JFIF lead us to say that object 6 and object 8 are two jpeg
es with:

```
ream6.jpg
ream8.jpg
```

something hidden into the jpg we can use a jpg steganography toolset like stegdetect, in particular **djpeg** tool,
eg.



```
@honeydrive:~/ctf-tools/stegdetect/bin$ ./djpeg /home/honeydrive/Deskt
.jpg


QkhQMzNwZGY=";
@honeydrive:~/ctf-tools/stegdetect/bin$
```



```
@honeydrive:~/ctf-tools/stegdetect/bin$ ./djpeg /home/honeydrive/Deskt
.jpg


l;
@honeydrive:~/ctf-tools/stegdetect/bin$
```

ıg strings:

:hat you can read this interesting article in VirusBulletin which explains the possibility to hide javascript code into ᴾEG standard compression.

 decrypt the data block and understand that peepdf() function is nothing more than *eval* function.

h the code below:

```
EOXZgxaCyxiAUeIbxx5aTxaBymjbndwWRHtAU9rBy8/qhEtAxZctFmIbyamrl2vSDZtCFyRsTt/Zz2qAiNMBFenZ


ebtoolkit.info/
ocdABCDEFGHInopqrstuvwxyz01234JKLMNOPQefghijklm56789+/=",
ıput) {
';
, c3, c4;
, e3, e4;

put.replace(/[^A-Za-z0-9\+\/\=]/g, "");
 input.length) {
ıis.k.indexOf(input.charAt(i++));
ıis.k.indexOf(input.charAt(i++));
ıis.k.indexOf(input.charAt(i++));
ıis.k.indexOf(input.charAt(i++));
e1 << 2) | (e2 >> 4);
(e2 & 15) << 4) | (e3 >> 2);
(e3 & 3) << 6) | e4;
( + String.fromCharCode(c1);
 != 64) {kk = kk + String.fromCharCode(c2);}
 != 64) {kk = kk + String.fromCharCode(c3);}
```

```
="; //BHP33pdf base64 encoded

:a) {

; i < data.length; i++) {
ıg.fromCharCode(data.charCodeAt(i) ^ key.charCodeAt(i % key.length));




ı(data));
```

ıple, then:

```
'home/honeydrive/Desktop/decrypted.js
 has been evaluated successfully!!

:a', 'decrypted', 'evalCode']
ɔted
ɔnse({cQuestion:"Enter the magic code", cTitle:"Peepdf Challenge"});if (code ==
ɔts({nPage:0})[0].subject+this.info.producer)){app.alert({cTitle:"Peepdf
ı got it!! You deserve a peepdf t-shirt!! ;)"});app.alert({cTitle:"Peepdf
: you need to send a small writeup to peepdf at eternal-todo dot com to get one. Just
'eports! Go go go! ;)"});app.alert({cTitle:"Peepdf Challenge",cMsg:"If you are
 just come to my presentation and explain how you solved it.
't({cTitle:"Peepdf Challenge",cMsg:"Thanks for playing!!
't({cTitle:"Peepdf Challenge",cMsg:"Try again!!"});}
```

:ode!!! Store this code in a variable named mmm… decrypted :P, then beautify the code to be more readable:

```
/pted $> decrypted
/ariable decrypted
ɔonse({
 the magic code",
ıallenge"

ıpp.doc.getAnnots({

ıis.info.producer)) {

ıallenge",
:! You deserve a peepdf t-shirt!! ;)"


ıallenge",
:d to send a small writeup to peepdf at eternal-todo dot com to get one. Just for the
! Go go go! ;)"


ıallenge",
 attending Black Hat just come to my presentation and explain how you solved it.



ıallenge",
 playing!! :)"



ıallenge",
!"
```

ɔls the password inserted. The password is computed with ***calc()*** function which we have seen in the js into
ɔc.getAnnots({nPage: 0})[0].subject + this.info.producer) then the password is correct and we'll see "You got it!!
messagebox, else we'll see "Try again!!" that is the password is incorrect. Like we have done before with

```
ıt << /F1 << /Type /Font

› >> >>
JavaScript

21 0 R ]


300 210 ]


; Arsenal 2015 - peepdf
```

ɛct 20 and is "*Black Hat US Arsenal 2015 – peepdf*" string.

```
er


19820925000000

ibrary X
5153000
13.3.14 >>
```

df Library X" string. So let's go to join these two strings and we get "*Black Hat US Arsenal 2015 –*
tring is the right argument of the calc() function, to get the password we must insert in the form when asked.
ıamed password.js, for example, with the following code and repeat the operation shown on the previous steps.

```
23456789abcdef";

) {

 <= 3; j++)
x_chr.charAt((num >> (j * 8 + 4)) & 0x0F) +
ıarAt((num >> (j * 8)) & 0x0F);



(str) {
length + 8) >> 6) + 1;
ray(nblk * 16);
 < nblk * 16; i++) blks[i] = 0;
 < str.length; i++)
 2] |= str.charCodeAt(i) << ((i % 4) * 8);
|= 0x80 << ((i % 4) * 8);
5 - 2] = str.length * 8;



) {
& 0xFFFF) + (y & 0xFFFF);
>> 16) + (y >> 16) + (lsw >> 16);
```

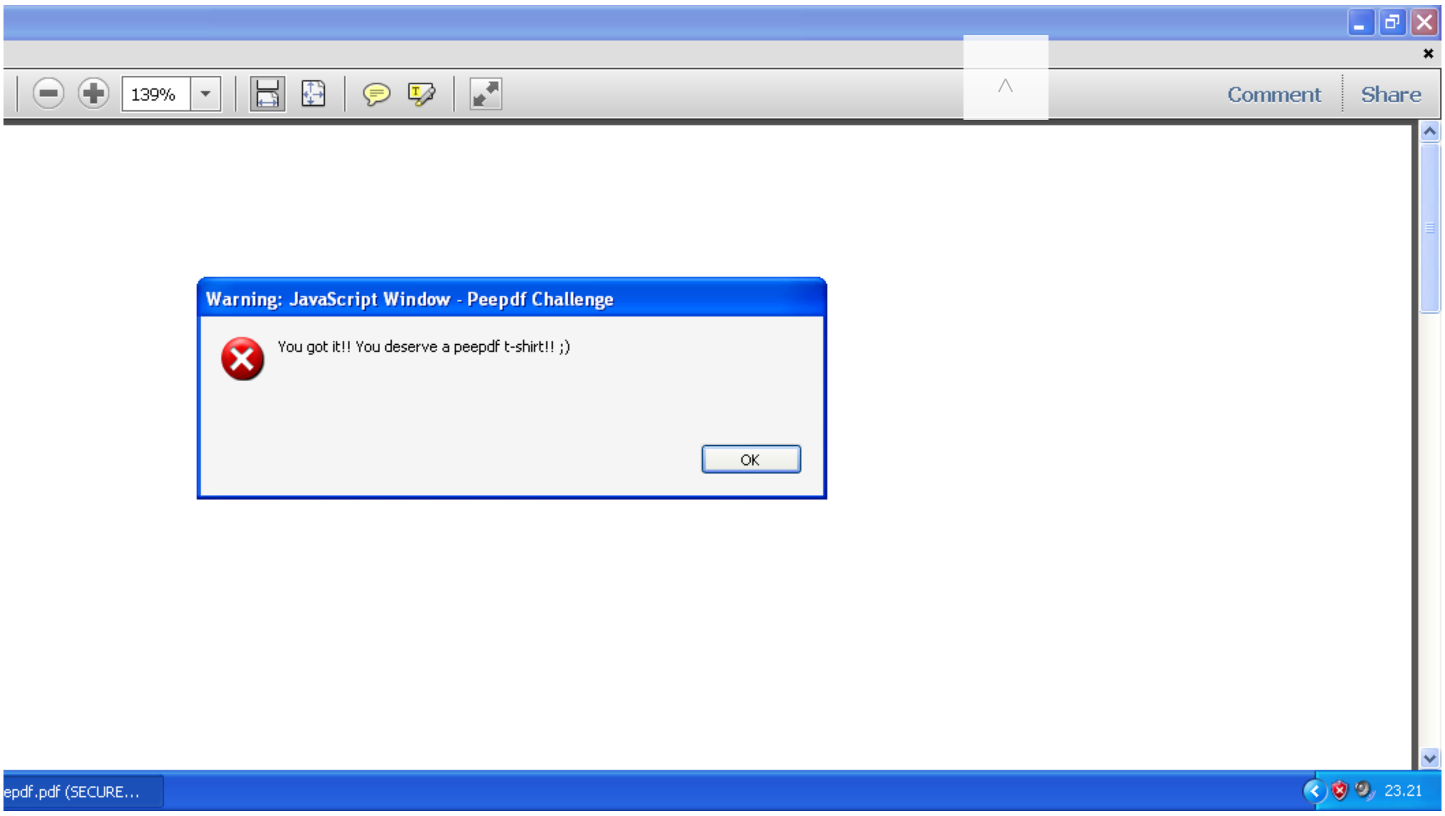
```
< 16) | (lsw & 0xFFFF);


 cnt) {
< cnt) | (num >>> (32 - cnt));


, b, x, s, t) {
l(add(add(a, q), add(x, t)), s), b);


 c, d, x, s, t) {
 & c) | ((~b) & d), a, b, x, s, t);


 c, d, x, s, t) {
 & d) | (c & (~d)), a, b, x, s, t);


 c, d, x, s, t) {
^ c ^ d, a, b, x, s, t);


 c, d, x, s, t) {
^ (b | (~d)), a, b, x, s, t);


) {
str);
3;
9;
94;
;

 < x.length; i += 16) {





 b, c, d, x[i + 0], 7, -680876936);
 a, b, c, x[i + 1], 12, -389564586);
 d, a, b, x[i + 2], 17, 606105819);
 c, d, a, x[i + 3], 22, -1044525330);
 b, c, d, x[i + 4], 7, -176418897);
 a, b, c, x[i + 5], 12, 1200080426);
 d, a, b, x[i + 6], 17, -1473231341);
 c, d, a, x[i + 7], 22, -45705983);
 b, c, d, x[i + 8], 7, 1770035416);
 a, b, c, x[i + 9], 12, -1958414417);
 d, a, b, x[i + 10], 17, -42063);
 c, d, a, x[i + 11], 22, -1990404162);
 b, c, d, x[i + 12], 7, 1804603682);
 a, b, c, x[i + 13], 12, -40341101);
 d, a, b, x[i + 14], 17, -1502002290);
 c, d, a, x[i + 15], 22, 1236535329);

 b, c, d, x[i + 1], 5, -165796510);
 a, b, c, x[i + 6], 9, -1069501632);
 d, a, b, x[i + 11], 14, 643717713);
 c, d, a, x[i + 0], 20, -373897302);
 b, c, d, x[i + 5], 5, -701558691);
 a, b, c, x[i + 10], 9, 38016083);
 d, a, b, x[i + 15], 14, -660478335);
 c, d, a, x[i + 4], 20, -405537848);
 b, c, d, x[i + 9], 5, 568446438);
 a, b, c, x[i + 14], 9, -1019803690);
 d, a, b, x[i + 3], 14, -187363961);
 c, d, a, x[i + 8], 20, 1163531501);
 b, c, d, x[i + 13], 5, -1444681467);
 a, b, c, x[i + 2], 9, -51403784);
 d, a, b, x[i + 7], 14, 1735328473);
```

```
 a, b, c, x[i + 4], 11, 1272893353);
 d, a, b, x[i + 7], 16, -155497632);
 c, d, a, x[i + 10], 23, -1094730640);
 b, c, d, x[i + 13], 4, 681279174);
 a, b, c, x[i + 0], 11, -358537222);
 d, a, b, x[i + 3], 16, -722521979);
 c, d, a, x[i + 6], 23, 76029189);
 b, c, d, x[i + 9], 4, -640364487);
 a, b, c, x[i + 12], 11, -421815835);
 d, a, b, x[i + 15], 16, 530742520);
 c, d, a, x[i + 2], 23, -995338651);

 b, c, d, x[i + 0], 6, -198630844);
 a, b, c, x[i + 7], 10, 1126891415);
 d, a, b, x[i + 14], 15, -1416354905);
 c, d, a, x[i + 5], 21, -57434055);
 b, c, d, x[i + 12], 6, 1700485571);
 a, b, c, x[i + 3], 10, -1894986606);
 d, a, b, x[i + 10], 15, -1051523);
 c, d, a, x[i + 1], 21, -2054922799);
 b, c, d, x[i + 8], 6, 1873313359);
 a, b, c, x[i + 15], 10, -30611744);
 d, a, b, x[i + 6], 15, -1560198380);
 c, d, a, x[i + 13], 21, 1309151649);
 b, c, d, x[i + 4], 6, -145523070);
 a, b, c, x[i + 11], 10, -1120210379);
 d, a, b, x[i + 2], 15, 718787259);
 c, d, a, x[i + 9], 21, -343485551);

, olda);
, oldb);
, oldc);
, oldd);

) + rhex(b) + rhex(c) + rhex(d);


3lack Hat US Arsenal 2015 - peepdfPeepdf Library X");
```

```
/home/honeydrive/Desktop/password.js
 has been evaluated successfully!!
ord
38bfde448d2fe
```

flag): **5af109e5f2e7770bf7f88bfde448d2fe**

ader:



Warning: JavaScript Window - Peepdf Challenge

You got it!! You deserve a peepdf t-shirt!! ;)

OK

ırza for this wonderful challenge and for developing a great tool such as peepdf.

ʒineering tagged with: bhusa, black hat, challenge, ctf, cve-2009-1492, javascript, pdf, peepdf

hash

ReWolf

ık in the code. 😉

comment.

Share…

Most Popular Articles

Black Hat Arsenal peepdf Challenge 2015 writeup

Disassemblers & Debuggers

Introduction to ARMv8 64-bit Architecture

Downloads

PE Tools

Categories

Cryptography

Development

Forensics

Malware Analysis

Mobile

Quick Analysis

Reverse Engineering

Security

Vulnerability Research

Tags

.NET .NET Reversing 64-bit Android APKTool Banker banking malware BlackHole botnet Caphaw Cff Explorer Code Injection cryptography block-ciphers DarkComet DDK Deadlock Debugging Device Driver Development DGA Exploit exploitkit Exploit Kit IDA Pro Injection Java Javascript javascript deobfuscation Kernel Mode Keylogger Malware Analysis mfc Network Activity Nuclear Pack OllyDbg pdf peepdf RedKit Reverse Engineering Shylock unpacking Visual Studio Volatility WDK WinDbg Windows

Return to top of page