Ghiribizzo's Cracking Tutorial

Cracking the PDF format (Part 2)

Cracking the PDF format

Well, you'll notice that my tutorials are in Adobe's Portable Document Format. And why not, it's a nice format, fairly small overhead (compare sizes with HTML) and has some nice features. +ORC has started a project to crack it.

PGP and Signed Tutorials

My tutorials and programs should be signed electronically using PGP. PGP 5 supports DSS/Diffie-Hellman keys. These keys are not supported by previous versions of PGP.
You should check the signature to make sure that the tutorial and especially its program files have not been tampered with. All cracks, tutorials and zip files I release will be signed. This will prevent tampering and will hopefully reduce the chances of viral infection.
My signature will also be the only way you can identify me as my email address will often change.

My Web Site: http://Ghiribizzo.home.ml.org
My Email: Ghiribizzo@geocities.com
My Backup Email: Ghiribizzo@hotmail.com

Having had a quick look around for some information on the PDF format, I became more and more alarmed at the blunders that Adobe had made in implementing security in its PDF format. By security I am not referring to the fact that the restrictions (on printing etc.) can be easily bypassed - Adobe and anybody of reasonable intellect knew this from the start. I began to wonder whether they had employed a secure one-way hash on the password to derive the key - although I use a different password per application, the idea that it would be compromised was still alarming. This is why I began to look into the PDF format and then realised a slightly unusual way of breaking into its permission scheme.

First let's get some information on our target. It uses RC4 to encrypt the data streams. RC4 is a variable-key-size stream cipher developed by Ron Rivest. I've said many times before that we need to cross-train: basic knowledge of cryptography will pay dividends here. For export reasons, the algorithm used will be limited to a 40 bit key - this could be useful for a dead-listing approach. Thankfully, there is a one-way hash scheme employed. MD5 is used (again developed by Ron Rivest). MD5 produces a 128 bit hash, so some manipulation will be involved to use the hash as the key for the RC4 encryption.

The following block is the encryption dictionary (taken from Ghiric7.pdf).

```
29 0 obj
<<
/Filter /Standard
/V 1
/R 2
/O (...)
/U (...)
/P 65476
>>
endobj
```

The P entry in the encryption dictionary is a bit mask; set bits indicate permitted operations. If we number the least significant bit zero:

Bit(s)	Meaning
0-1	Always Clear?
2	Print Document
3	Modify Document
4	Select Text & Graphics
5	Add / Modify Notes
6-15	Always Set
16-31	Always Clear

You will notice that in the PDF file this is stored as a decimal equivalent. e.g. in Ghiric7.pdf we have "/P 65476" which is the decimal equivalent of 'FFC4' the least significant byte being '11000100' i.e. only 'Print Document' is enabled.

/O and /U are the Owner and User password data. I don't know how these are derived, but you should be able to find it if you take a look in the PDF reference manual.

We know that the permissions data cannot be modified in the file as the file key is generated using data from it. The file key is generated using the MD5 algorithm... so let's look for the algorithm and see if that gives us any clues as to where the permissions data is located in memory. We need to find a search string for the MD5 algorithm. The initialising values of the chaining variables is ideal. Having a look through the source code for the MD5 algorithm, we find the following values.

```
void MD5Init(struct MD5Context *ctx)
{
    ctx->buf[0] = 0x67452301;
    ctx->buf[1] = 0xefcdab89;
    ctx->buf[2] = 0x98badcfe;
    ctx->buf[3] = 0x10325476;

    ctx->bits[0] = 0;
    ctx->bits[1] = 0;
}
```

And searching through Hiew we find the corresponding instructions:

```
.0007E3DC: C7400489ABCDEF              mov    d,[eax][00004],0EFCDAB89
.0007E3E3: C74008FEDCBA98              mov    d,[eax][00008],098BADCFE
.0007E3EA: C7400C76543210              mov    d,[eax][0000C],010325476
.0007E3F1: C70001234567                mov    d,[eax],067452301
```

I decided to use the live approach. Firing up Soft-Ice I put a bpx on the above code and loaded Ghiric7.pdf in Acrobat Reader. SoftIce popped up. Did the check come before or after this? I *knew* it was after and I could *feel* that it was close by. So I decided to quickly scan though the code and then stepped though the code quickly. You could have just as easily put a break on memory when you found the locations but when I *felt* that the code was close by I just scanned around for it. There are many places you could patch Reader.

```
Comparing files AC32.BAK and acrord32.exe
0007FE5A: 8B B1
0007FE5B: 4E FC
0007FE5C: 78 90
```

This corresponds to the following Hiew listing: (~ replaced with *)

```
~.00080A5A: 8B4E78                     mov    ecx,[esi][00078]
~.00080A5D: 0BC8                       or     ecx,eax
  00080A5F: F6C101                     test   cl,001
 .00080A62: 894E78                     mov    [esi][00078],ecx

*.00080A5A: B1FC                       mov    cl,0FC     ;forces a value of 7FFC
*.00080A5C: 90                         nop
```

There is a more elegant place to crack this (involving a 1 byte change to an AND instruction). Note that the permission tag has been altered from FF?? to 7F??. There is some confusion as to what to place in the lower byte. As you can't edit PDF files in Reader anyway, there's no need to set some of the flags. The first 2 bits seem to always be clear so it may be a good idea to keep them that way. However, I've tried values from C0-FF and they give results as described in the table above. D4 will give you printing and text selection, you could go for FF, but I don't think it gives you anything more.

If you have Acrobat Exchange then you'll find that searching for the above will yield the same thing. Re-using old cracks works for the vast majority of the time (see ghiric11.pdf). Permissions can be altered to give you editing capability as well. However, this is not a satisfactory crack for Exchange as you are still unable to *remove* the security settings - a password is still required for that. Further alterations must be made to be able to remove the security settings or possibly by altering the permission flag earlier on. However, with this crack you are able to save the modified file, but the old security settings (and passwords) are kept.