

[NEWS] Implementation Flaws in Adobe Document Server for Reader Extensions

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2003-03/0018.html>

From: support@securiteam.com

Date: 03/09/03

From: support@securiteam.com

To: list@securiteam.com

Date: 9 Mar 2003 19:36:14 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

In the US?

Contact Beyond Security at our new California office
housewarming rates on automated network vulnerability
scanning. We also welcome ISPs and other resellers!

Please contact us at: 323-882-8286 or ussales@beyondsecurity.com

Implementation Flaws in Adobe Document Server for Reader Extensions

SUMMARY

A vulnerability in the way Adobe Document Server encrypts parts of the document allows an attacker to break the encryption, or alternatively modify the document in such a way, as the signature (what displays whether the document is in its original form, unmodified) remains valid even though the document has been modified.

DETAILS

Technical info:

Adobe Document Server for Reader Extensions

With this server, customers can assign custom usage rights to specific Adobe Portable Document Format (PDF) forms and documents, so Acrobat Reader 5.1 users can get access to additional features while working with the document. The server can enable four types of usage rights on a PDF form:

- Commenting tools, including sticky notes, highlights, stamps, and strikethroughs;

Securiteam: [NEWS] Implementation Flaws in Adobe Document Server for Reader Extensions

- The ability to save a form to a desktop for offline completion or archiving, without losing any forms data;
- Digital signatures, including support for Public Key Infrastructure systems for third-party validation (VeriSign, Entrust, and others);
- Advanced form features, including the ability to submit a form offline or via email, import or export forms data and attached files.

Description of Adobe Document Server for Reader Extensions features is available at

<http://www.adobe.com/products/server/readerextensions/main.html>
<http://www.adobe.com/products/server/readerextensions/main.html>.

The implementation of Document Server for Reader Extensions does not seem to be very complicated. The Server just gets the PDF file (to be reader-enabled) together with the list of enabling options, and produces new document that contains one additional dictionary – actually, simply by adding an additional block of data.

Note: for details of PDF structure, see Portable Document Format Specification

<http://partners.adobe.com/asn/developer/acrosdk/docs/filefmtspecs/PDFReference.pdf>
<http://partners.adobe.com/asn/developer/acrosdk/docs/filefmtspecs/PDFReference.pdf>

New dictionary is named "ViewerPreferences" and resides within document's "Root" dictionary. For now, only one element is placed inside "ViewerPreferences" dictionary – "Rights" Dictionary. Content of the "Rights" Dictionary can be described as follows (key name, type, and description):

"Version" (number): A number specifying the version of Rights dictionary. Currently only version 1 is supported.

"Document" (array of names): List of flags related to Document operations. Currently only one flag is supported: FullSave.

"Form" (array of names): List of flags related to Form processing. Supported flags: Import, Export, SubmitStandalone, SpawnTemplate.

"Annots" (array of names): List of flags related to Annotations. Supported flags: Create, Delete, Modify, Copy, Import, Export.

"Signature" (array of names): List of flags related to Digital Signature handling. Currently only one flag is supported: Modify.

"Msg" (text string): Arbitrary string to be displayed in "Instructions" box when reader-enabled document is opened in Acrobat Reader 5.1.

"TimeOfUbiquitization" (date): The date and time when document was processed by Document Server for Reader Extensions.

Securiteam: [NEWS] Implementation Flaws in Adobe Document Server for Reader Extensions

"RightsID" or "ADBE_RightsID" (array): List of RSA-based digital signatures for checking integrity of reader-enabling attributes.

Most elements listed in the table above are self-descriptive. Name of the last key could be either RightsID or ADBE_RightsID – they are equivalent in Reader 5.1. Values in RightsID (or ADBE_RightsID) array are 512-bit RSA-encrypted values, and could be decrypted with RSA Public Key, which is hard-coded (in encrypted form) within Reader 5.1 executable. Those values are used as digital signatures of some critical document parts to make sure that document was reader-enabled with Adobe Document Server for Reader Extensions.

Sample form with additional usage rights could be downloaded from
<http://www.adobe.com/products/server/readerextensions/pdfs/sample_docserver_readerext.pdf>
http://www.adobe.com/products/server/readerextensions/pdfs/sample_docserver_readerext.pdf

According to press release from October 21, 2002, available at
<<http://www.adobe.com/aboutadobe/pressroom/pressreleases/pdfs/200210/20021021Ubiquity.pdf>>
<http://www.adobe.com/aboutadobe/pressroom/pressreleases/pdfs/200210/20021021Ubiquity.pdf>, pricing of Adobe Document Server for Reader Extensions starts at US\$75,000.

Adobe Acrobat Reader

Adobe Acrobat Reader is the most popular part of Adobe Acrobat product family. Acrobat Reader is free and can be downloaded from Adobe Systems Incorporated web site

<<http://www.adobe.com/products/acrobat/readermain.html>>
<http://www.adobe.com/products/acrobat/readermain.html>.

Other most interesting (for this statement) products from Adobe Acrobat family are commercial Adobe Acrobat (\$249) and Adobe Acrobat Approval (\$39). Comparison of the free Acrobat Reader 5.1 with Acrobat 5.0 and Acrobat Approval could be found at Planet PDF web site:

<<http://www.planetpdf.com/mainpage.asp?webpageid=2413>>
<http://www.planetpdf.com/mainpage.asp?webpageid=2413>.

For a long time Acrobat Reader was just "the reader" – its ability of editing was limited to form filling, but there was no way to get altered copy of the document on disk. Acrobat Reader 5.1, however, includes some new functions, that extend Reader functionality, and make it very close to commercial Acrobat Approval. In other words, new Acrobat Reader 5.1 is something like a limited version of Acrobat Approval with functional restrictions applied to all documents except reader-enabled ones.

When the document is opened in Acrobat Reader 5.1, the Reader looks inside ViewerPreferences/Rights dictionary, and based on information found there, enables some editing features (or keep them disabled).

The problem

As noted earlier, 512-bit RSA encryption is being used to prevent creating reader-enabled documents without Adobe Document Server for Reader

Securiteam: [NEWS] Implementation Flaws in Adobe Document Server for Reader Extensions

Extensions. 512-bit RSA keys are not considered non-breakable – in 1999, a group of researchers completed the factorization of the 512-bit RSA Challenge Number in 5.2 months (see <http://www.rsasecurity.com/rsalabs/challenges/factoring/rsa155.html> for details). However, it seems that building a system for breaking RSA-512 is more expensive than obtaining Document Server for Reader Extensions from Adobe. Therefore, 512-bit RSA is sufficient for such application. But, as it often happens, implementation corrupts the idea.

"RightsID" key from "Rights" Dictionary usually contains three values. First value is a signature for the "Rights" Dictionary content (excluding "RightsID"/"ADBE_RightsID" key itself), "CreationDate", "Producer" and "Creator" keys from PDF Info Dictionary, and Document ID from PDF Trailer Dictionary. Second and third values from "RightsID" key are signatures of the first and the last pages of PDF document content, respectively.

The document is considered to be enabled properly only when the "Rights" Dictionary signature and at least one Page Content signature are correct.

Let us start the research with page content signature calculation.

At the first stage, every byte of decompressed page content (but not more than 1024 bytes) is passed through the hash function. Hash is not too complicated – some non-printable characters (such as Space, Tab, CR, LF) are ignored, and ASCII-codes of all remaining characters are added to 32-bit counter, initially set to zero.

At the second stage, the 32-bit value calculated as described above (actually, the largest possible value contains only 18 significant bits) is converted to 8-byte hexadecimal representation, and "PaGeS" string is appended to the end. Resulting 13-byte sequence is processed with MD5 hash function, which produces 128-bit (16-byte) page fingerprint.

At the last stage, 512-bit RSA Private Key (stored somewhere in Server) is used to encrypt page fingerprint. Encrypted value is stored in "RightsID" array. To verify Page Content signature, the Reader calculates page fingerprint and compares it with the value decrypted using RSA Public Key.

Obviously, the signature calculation procedure described above is not perfect. Adding "PaGeS" string does not affect the reliability at all. However, it looks like a sort of tradition in Adobe: in undocumented version of key calculation algorithm (V3), used by Standard Security Handler in Adobe Acrobat 5, string "sAIT" has been used in the same way – instead of the real "salt" value, that should be completely random.

However, the most significant problem with Page Content signature calculation is it is very easy to build 1024-byte sequence of characters, treated as comments by PDF interpreter, so the sum of all characters in that sequence will be the same as the sum of first 1024 characters in any reader-enabled document. Therefore, inserting such comment before any page

will produce a new page, interpreted exactly like original one, with fixed (known) signature value. Let us pay some attention to the "Rights" Dictionary signature calculation.

During that operation, some 32-bit hash (not cryptographic one, but a little bit more complicated than just addition) is calculated from all keys of the "Rights" Dictionary excluding "RightsID" array. Hash value is converted to hexadecimal representation and passed through MD5 hash function along with the values from PDF Info Dictionary such as "CreationDate", "Producer" and "Creator", and constant string "Adobe Acrobat". First five bytes of MD5 output are treated as a key for RC4 symmetric stream cipher, which is used to encrypt Document ID from PDF Trailer Dictionary. Encrypted ID is encrypted again with RSA Private Key to produce "Rights" Dictionary signature that should be stored as the first element of the "RightsID" array.

This procedure looks not too bad in comparison with Page Content signature calculation, but not perfect too. Now, the problem is related to Document ID and RC4. It is well known that RC4 (like many other stream ciphers) is just a pseudo-random sequence generator, and generated stream is combined with source data by XOR operation to produce encrypted data. Therefore, if we need to force "Rights" Dictionary signature to be equal to some predefined value, the following approach should work:

- Decrypt signature (from any reader-enabled document) by RSA Public Key to obtain RC4-encrypted Document ID;
- Calculate "Rights" Dictionary and Info Dictionary hash (from document to be reader-enabled) to obtain RC4 key;
- Decrypt Document ID value by RC4 and assign it to the new document.

It is necessary to note that Document ID consists of two parts: permanent identifier and variable identifier. Only permanent identifier is involved in the signature calculation. The Document ID is used to resolve external references by ID, and almost useless in "standalone" documents. So, assigning the new Document ID is not a big issue.

Impact:

As it was shown above, all 512-bit RSA-based signature checks could be easily bypassed. It is very difficult to break RSA-512 cipher itself and calculate signature values for arbitrary document, but it is much easier to modify document to match a fixed (known) signature values without noticeable changes in the document's content, and without breaking, stealing or purchasing RSA private key.

It is looks like the only party negatively affected by such facts is Adobe Systems Incorporated itself. However, if some U.S. government agency will decide to spend \$75K for Adobe Document Server for Reader Extensions, all U.S. taxpayers will be affected by indirection, too.

Conclusion:

It is not too hard to guess why Adobe Document Server for Reader Extensions has been introduced. Generally, there are two main reasons –

Securiteam: [NEWS] Implementation Flaws in Adobe Document Server for Reader Extensions

Government Paperwork Elimination Act (GPEA), U.S. federal agencies must adhere by October 2003 to; and recently announced Microsoft InfoPath (formerly code-named "XDocs"), which promises to become the main competitor for PDF Forms and Adobe Acrobat products family. In addition, efforts to seizure market niche as soon as possible in this case obviously leads to bad design and implementation flaws in cryptography-based operations.

This issue does not deal with security, but deals with cryptography that is widely used to provide security. Using of modern cryptography (RSA, RC4, MD5) cannot guarantee that the information is secured properly. Moreover, even if big companies do not have enough resources to hire competent security/cryptography specialist, what should we expect from the market in whole?

Vendor response:

The problem has been reported to vendor (Adobe Systems Inc) on 02/24/2003; vendor has not replied.

ADDITIONAL INFORMATION

The information has been provided by <mailto:info@elcomsoft.com>
Elcomsoft.com.

=====

This bulletin is sent to members of the SecuriTeam mailing list.
To unsubscribe from the list, send mail with an empty subject line and body to:
list-unsubscribe@securiteam.com
In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind. In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.